Data Analytics with Python

Pau, France

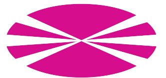
25 - 28 Octobre, 2017

Co-funded by the
Erasmus+ Programme
of the European Union

Python

- General-purpose language, interpreted, object-oriented.
- Portable: an interpreter exist for almost any OS.
- Open-source reference implementation (CPython), but alternative ones exist as well (e.g., iPython, Intel Python).
- Powerful and versatile:
  - Dynamic typing.
  - Complex “basic” types: lists, dictionaries, strings, …
  - Built-in tools: powerful list operations, strings, arrays, ...
  - Libraries: math, statistics, parallelization, …
  - Automatic memory management.

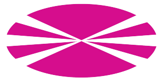
Ecole Internationale des Sciences du Traitement de l'Information



Python

- Popular with the scientific and engineering communities.
- Free and general purpose alternative to Matlab/R.
- “Slice” notation (Matlab-like).
- Multiple extensions: NumPy, Matplotlib, SciPy, pytables, …
- Simple integration: C/C++/Fortran, web services, …
- Web development: Django, Flask, Pyramid, …
- Scientific computing environments: iPython, Anaconda, ...

Ecole
Internationale
des Sciences
du Traitement
de l'Information

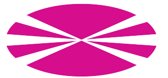


Python - Executing programs

1. Using the interactive interpreter:

```
>>> txt = "Luck, I am your father"
>>> print txt
Luck, I am your father
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Executing programs

2. Interpreting Python code:

```
------------------------------- > script.py
txt = “Luck, I am your father”
print txt
-------------------------------


$ python2 script.py
Luck, I am your father
```

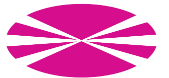
Ecole Internationale des Sciences du Traitement de l'Information

Python - Executing programs

3. Shell script:

```
------------------------------- > script.py
#! /usr/bin/python2
txt = “Luck, I am your father”
print txt
-------------------------------

$ ./script.py
Luck, I am your father
```

Ecole Internationale des Sciences du Traitement de l'Information

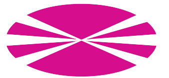


Python - Executing programs

4. Embedding Python into C:

```
------------------------------- > prueba.c
#include <Python.h>
…
Py_Initialize();
PyRun_SimpleString( "txt = 'Luck, I am your father'" );
PyRun_SimpleString( "print txt" );
…
-------------------------------
$ gcc -I /usr/include/python2.7 prueba.c -lpython2.7
$ ./a.out
Luck, I am your father
```

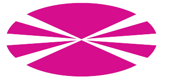
Ecole Internationale des Sciences du Traitement de l'Information



Python - Semantics & Syntax
Indentation, not brackets

```
if x < threshold:
    print “x less than threshold”
else:
    print “x greater than threshold”;
```

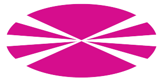
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Semantics & Syntax
Everything is an object

- Including numbers, strings, functions, classes, modules, …
- Each object has a type and its own data.
- This provides flexibility, allowing to deal with any element in a generic way.

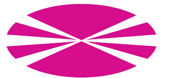
Ecole Internationale des Sciences du Traitement de l'Information



Python - Semantics & Syntax
Comments

```
# This is a dramatic moment.
txt = “Luck, I am your father”
print txt
```

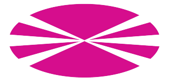
Ecole Internationale des Sciences du Traitement de l'Information



Python - Semantics & Syntax
Function calls

- C syntax:
  - `result = f( x, y, z )`
- Class methods:
  - `result = obj.f( x, y, z )`
- Functions may be passed positional or keyword arguments:
  - `result = f( x, y, z, tol=0.01, method=”fast” )`

Ecole
Internationale
des Sciences
du Traitement
de l'Information

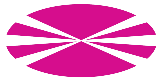


Python - Semantics & Syntax
Variables and passing by reference

- Assignment to a variable generates a new reference to an object.
- The original object remains the same!

```
>>> a = [1,2,3]
>>> b = a
>>> a.append(4)
>>> b
[1,2,3,4]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



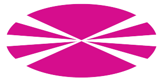
Python - Semantics & Syntax
Variables and passing by reference

Consequently:

- Passing large variables is efficient.
- ¡Functions may permanently modify their parameters!
- Except for **immutable** types. E.g.,

```
>>> a = 10
>>> b = a
>>> a = 20
>>> print b
10
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Semantics & Syntax
Dynamic references, strong typing

References (variables) have no static type:

```
>>> a = 10
>>> type(a)
<type ‘int’>
>>> a = “10”
>>> type(a)
<type ‘str’>
```

Ecole Internationale des Sciences du Traitement de l'Information

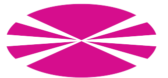
Python - Semantics & Syntax
Dynamic references, strong typing

Python has strong typing, nevertheless:

```
>>> 5 + “5”
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> “5” + 5
TypeError: cannot concatenate 'str' and 'int' objects
>>> 5 + int(“5”)
10
>>> “5” + str(5)
‘55’
```

Each Python object has a specific type (class). Implicit casting is only performed for `int` to floating point conversions.

Ecole Internationale des Sciences du Traitement de l'Information

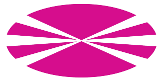


Python - Semantics & Syntax
Dynamic references, strong typing

The `isinstance` operator is used to check whether an object belongs to a particular class:

```
>>> a = 4.5
>>> isinstance(a,int)
False
>>> isinstance(a,float)
True
```

Ecole Internationale des Sciences du Traitement de l'Information

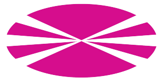


Python - Semantics & Syntax
Methods and Attributes

Python objects have methods and attributes:

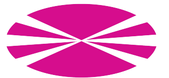
```
>>> a = 4.5
>>> dir(a)
['__abs__', '__add__', '__class__', '__coerce__', '__delattr__', '__div__',
 '__divmod__', '__doc__', '__eq__', '__float__', '__floordiv__', '__format__',
 '__ge__', '__getattribute__', '__getformat__', '__getnewargs__', '__gt__',
 '__hash__', '__init__', '__int__', '__le__', '__long__', '__lt__', '__mod__',
 '__mul__', '__ne__', '__neg__', '__new__', '__nonzero__', '__pos__', '__pow__',
 '__radd__', '__rdiv__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__',
 '__rfloordiv__', '__rmod__', '__rmul__', '__rpow__', '__rsub__', '__rtruediv__',
 '__setattr__', '__setformat__', '__sizeof__', '__str__', '__sub__',
 '__subclasshook__', '__truediv__', '__trunc__', 'as_integer_ratio', 'conjugate',
 'fromhex', 'hex', 'imag', 'is_integer', 'real']
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Semantics & Syntax
Methods and Attributes

```
>>> a.hex()
'0x1.2000000000000p+2'
>>> hasattr( a, “hex” )
True
>>> getattr( a, “hex” )
<built-in method hex of float object at 0x1ed86a0>
>>> getattr( a, “hex” )()
'0x1.2000000000000p+2'
```

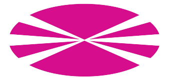



Python - Semantics & Syntax
Importing modules

- A Python module is a `.py` file including definitions and/or code:

```
------------------------------- > modulo1.py
PI = 3.14159

def f(x):
    return x + 2

def g(a,b):
    return a + b
-------------------------------
```

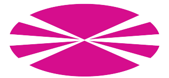
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Semantics & Syntax
Binary operators

`a + b`	Addition	`a - b`	Subtraction
`a * b`	Multiplication	`a / b`	Division (Python 3)
`a // b`	Integer division (Python 3)	`a ** b`	Exponentiation
`a & b`	Bitwise AND	`a | b`	Bitwise OR
`a ^ b`	Bitwise XOR	`a == b`	Equality
`a != b`	Inequality	`a < b, a <= b`	Less/Less or equal than
`a > b, a >= b`	Greater/Greater or equal than	`a is b`	Identity: True if `a` and `b` are the same object

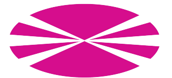
Ecole Internationale des Sciences du Traitement de l'Information



Python - Scalar types

`None`	“null” in Python. Singleton class.
`str`	String type. ASCII in Python 2.x, Unicode in Python 3.
`unicode`	Unicode string.
`float`	Floating point number, double precision (64 bits).
`bool`	Logical, `True` or `False`.
`int`	Signed integer. 32 or 64 bits depending on platform.
`long`	Signed integer, arbitrary precision.

Ecole Internationale des Sciences du Traitement de l'Information



Python - Scalar types
Numerical types

- `int` (32- or 64-bits), `float` (64 bits) and `long` (arbitrary precision).
- Conversion between `int` and `long` is transparent.
- Scientific notation accepted:

```
>>> fval = 6.78e-5
```

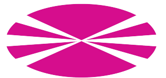
- Division of two integers in Python 2.x yields an integer (truncated, not rounded). For floating point division of integers:

```
>>> fval = a_int / float(b_int)
```

- Complex numbers are written using `j` for the imaginary part:

```
>>> cval = 1 + 2j
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Scalar types
Strings

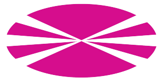
- String literals are written between single or double quotes:

```
>>> a = ‘one way to write a string’
>>> b = “a different way”
```

- To write strings with line breaks, use three single quotes:

```
>>> c = ‘‘‘
This is a larger string
Which spans more than one line
’’’
```

Ecole Internationale des Sciences du Traitement de l'Information

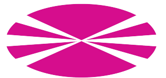


Python - Scalar types
Strings

- Strings are immutable objects: once they are created, they cannot be modified:

```
>>> a = 'this is a string'
>>> a[10] = 'f'
TypeError: 'str' object does not support item assignment
>>> b = a.replace( 'string', 'larger string')
>>> b
'this is a larger string'
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

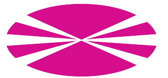


Python - Scalar types
Strings

- Strings can be created from different types using `str()`:

```
>>> a = 5.6
>>> s = str(a)
>>> s
'5.6'
```

Ecole Internationale des Sciences du Traitement de l'Information

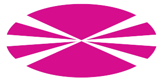


Python - Scalar types
Strings

- A string is just a **collection** of characters (so not really a scalar type):

```
>>> s = ‘python’
>>> list(s)
[ ‘p’, ‘y’, ‘t’, ‘h’, ‘o’, ‘n’ ]
>>> s[:3]
‘pyt’
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

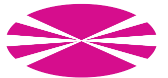


Python - Scalar types
Strings

- The backslash (`\`) is used as the scape character, to encode special characters such as line breaks (`\n`) or Unicode characters.
- A string which should be intepreted literally (raw) can be marked using `r‘string’`:

```
>>> s = r‘no\special\characters\in\this\string’
>>> s
‘no\special\characters\in\this\string’
```

Ecole Internationale des Sciences du Traitement de l'Information

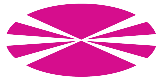


Python - Scalar types
Strings

- Operator + applies string concatenation:

```
>>> a = ‘first half of string ’
>>> b = ‘and second half of string’
>>> a + b
‘First half of string and second half of string’
```

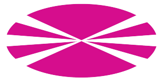
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Scalar types
Strings

- To specify that a string is encoded using Unicode, `u’string’` is used.
- To format a string there’s a C-like template syntax:

```
>>> template = u“%.2f %s are worth %d€”
>>> template % (1.18018, u‘american dollars’, 1)
u'1.18 american dollars are worth 1\u20ac'
>>> print template % (1.18018, u‘american dollars’, 1)
‘1.18 american dollars are worth 1€’
```

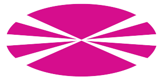



Python - Scalar types
Booleans

- The two boolean values are written `True` and `False`.
- They can be combined using the keywords `and` and `or`.

```
>>> True and True
True
>>> False and True
True
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

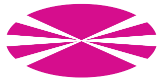


Python - Scalar types
Booleans

- All basic predefined types in Python, as well as any class implementing the `__nonzero__()` method, can be interpreted as `True` or `False` in a conditional clause.

```
>>> a = [1,2,3]
>>> if a: print “True”
‘True’
>>> b = []
>>> if not b: print “False”
‘False’
```

Ecole Internationale des Sciences du Traitement de l'Information



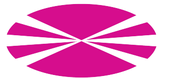
Python - Scalar types
None

- `None` is the null value in Python.
- If a function does not explicitly return a value, it returns `None` implicitly.
- Can be used as a default value for optional parameters to functions:

```
def add_and_maybe_multiply( a, b, c = None ):
    result = a + b
    if c is not None: result *= c
    return result
```

- `None` is not a reserved word, but an instance (singleton) of `NoneType`.

Ecole Internationale des Sciences du Traitement de l'Information



Python - Scalar types
Dates and times

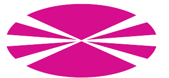
- The `datetime` provides the types `datetime`, `date` and `time`.

```
>>> from datetime import datetime, date, time
dt = datetime( 2015, 04, 24, 12, 25, 32 )
```

```
>>> dt.day
24
```

```
>>> dt.time()
datetime.time( 12, 25, 32 )
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



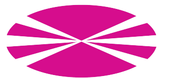
Python - Scalar types
Dates and times

- The `strftime()` method formats a `datetime` object as a string:

```
>>> dt.strftime( '%d/%m/%Y %H:%M' )
'24/04/2015 12:25'
```

- A `datetime` object can be created from a string using `strptime`:

```
>>> datetime.strptime( '20150424', '%Y%m%d' )
datetime.datetime( 2015, 4, 24, 0, 0 )
```

Python - Scalar types
Dates and times

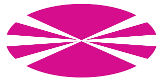
- The subtraction of two `datetime` objects returns a `datetime.timedelta`:

```
>>> dt2 = datetime( 2015, 4, 25, 12, 25, 32 )
>>> dt2 - dt
datetime.timedelta(1)
```

- Operating a `datetime` object with a `timedelta` returns a new `datetime`:

```
>>> from datetime import timedelta
>>> dt + timedelta(days=1)
>>> datetime.datetime(2015, 4, 25, 12, 25, 32)
```

Ecole Internationale des Sciences du Traitement de l'Information



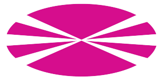
Python - Flow control

`if, elif, else`

These blocks behave like in most other languages:

```
if x < 0:
    print ‘Negative’
elif x == 0:
    print ‘Zero’
elif 0 < x < 5:
    print ‘Positive less than 5’
else:
    print ‘Positive and greater or equal than 5’
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



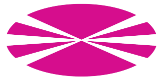
Python - Flow control
`for` loops

- A `for` loop is used to iterate over a collection (like a list or tuple):

```
for val in collection:
    # Do something with val
```

- `continue` skips to the next iteration
- `break` exits the current loop

Ecole Internationale des Sciences du Traitement de l'Information



Python - Flow control
`range()` and `xrange()`

- `range()` returns a list of equally spaced integers:

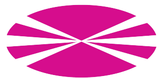
```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- Takes as parameters the start, end, and step of the sequence:

```
>>> range(5,20,3)
[5,8,11,14,17]
```

- Generates values in the interval `[start,end)`.

Ecole Internationale des Sciences du Traitement de l'Information

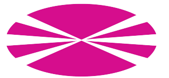


Python - Flow control
`range()` and `xrange()`

- `range()` is useful to code `for` loops with C semantics:

```
>>> for x in range(10):
...     print 2*x
0
2
4
8
16
18
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Flow control

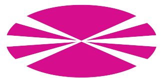
`range()` and `xrange()`

- `xrange()` accepts the same parameters as `range()`, but returns a generator instead of building the full list in memory.
- It features lazy evaluation.

```
>>> xrange(10)
xrange(10)
>>> for x in xrange(10):
...     print 2*x
0
2
…
```

- It is preferable to use `xrange()` when working with large ranges.

Ecole Internationale des Sciences du Traitement de l'Information

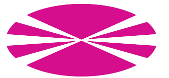


Python - Flow control
`while` loops

- Iterate **while** a condition is met:

```
x = 256
total = 0
while x > 0:
    if total > 500: break
    total += x
    x = x // 2
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



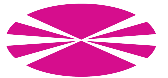
Python - Flow control

`pass`

- `pass` is a no-op instruction in Python.
- It is sometimes required because white spaces in Python delimit execution blocks:

```
if x < 0:
    print “Negativo”
elif x == 0: pass
else:
    print “Positivo”
```

Ecole Internationale des Sciences du Traitement de l'Information



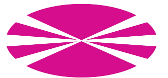
Python - Flow control
Exception handling

- Functions and operations can raise exceptions:

```
>>> 5 / 0
ZeroDivisionError: integer division or modulo by zero
```

- Exceptions can be handled to solve runtime errors.
- This allows to dynamically manage some selected error types.

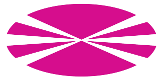
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Flow control
Exception handling

```
def float_division( a, b ):
    x = NaN
    try:
        x = a / float(b)
    except:
        print “Exception during division”
    return x
```

Ecole Internationale des Sciences du Traitement de l'Information

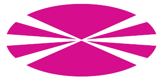


Python - Flow control
Exception handling

- We can explicitly list the exception types managed by a `except` block:

```
def float_division( a, b ):
    x = NaN
    try:
        x = a / float(b)
    except ZeroDivisionError:
        print "Division by zero"
    return x
```

Ecole Internationale des Sciences du Traitement de l'Information

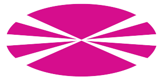


Python - Flow control
Exception handling

- Multiple exception types can be managed by the same block:

```
def float_division( a, b ):
    x = NaN
    try:
        x = a / float(b)
    except ZeroDivisionError, TypeError:
        print "Exception during division"
    return x
```

Ecole Internationale des Sciences du Traitement de l'Information

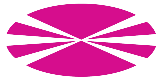


Python - Flow control
Exception handling

- A `finally` block is executed regardless of whether the `try` was successful:

```
f = open( path, 'w' )
try:
    write_to_file( f )
finally:
    f.close()
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

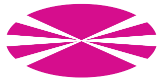


Python - Flow control
Exception handling

- An `else` block will only be executed if the `try` was successful:

```
f = open( path, ‘w’ )
try:
    write_to_file( f )
except: print ‘Failure’
else: print ‘Success’
finally: f.close()
```

Ecole Internationale des Sciences du Traitement de l'Information

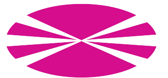


Python - Collections
Tuples

- Tuple: unidimensional sequence of fixed size containing Python objects.

```
>>> tup = (2, 3, 4)
>>> nested_tup = ( (1, 2), (2, 3, 4) )
>>> tup_from_list = tuple( [2,3,4] )
>>> tup_from_iter = tuple( xrange(5) )
>>> tup_from_string = tuple( ‘string’ )
```

Ecole Internationale des Sciences du Traitement de l'Information

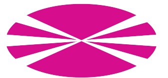


Python - Collections
Tuples

- The elements in a tuple are accessed using the `[]` operator:

```
>>> tup_from_iter[3]
3
>>> tup_from_iter[1:3]
(1,2)
>>> tup_from_string[0]
‘s’
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Collections
Tuples

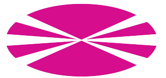
- Tuples are immutable:

```
>>> tup_from_iter[3] = 5
TypeError: ‘tuple’ object does not support item assignment
```

- However, objects inside a tuple can be mutable:

```
>>> tup = ( (1,2), [3,4,5] )
>>> tup[1].append(6)
>>> tup
((1, 2), [3, 4, 5, 6])
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Collections
Tuples

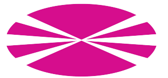
- Tuples are concatenated using the + operator:

```
>>> tup_from_iter + tup_from_string
(0, 1, 2, 3, 4, ‘s’, ‘t’, ‘r’, ‘i’, ‘n’, ‘g’)
```

- The * operator is consistent with the addition/concatenation semantics:

```
>>> tup_from_iter*3
(0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4)
```

Ecole Internationale des Sciences du Traitement de l'Information

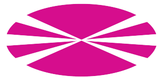


Python - Collections
Tuples

- Note that the objects contained in tuples are not copied, but referenced:

```
>>> tup = ( (1, 2), (3, 4, 5) )
>>> tup2 = tup * 3
>>> tup[1].append(6)
>>> tup2
((1, 2), [3, 4, 5, 6], (1, 2), [3, 4, 5, 6], (1, 2), [3,
4, 5, 6])
```

Ecole Internationale des Sciences du Traitement de l'Information

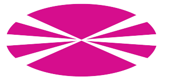


Python - Collections
Tuples

- If the right hand side of an assignment is a tuple, Python tries to unpack it:

```
>>> tup = (1,2,3)
>>> a, b, c = tup
>>> b
2
```

Ecole Internationale des Sciences du Traitement de l'Information



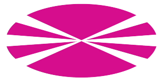
Python - Collections
Tuples

- Nested tuples can be explicitly unpacked:

```
>>> tup = ( 1, 2, (3, 4) )
>>> a, b, (c, d) = tup
>>> c
3
>>> a, b = b, a
>>> a
2
```

- A common use of tuples is to code functions that return multiple values.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

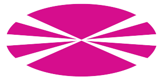


Python - Collections
Lists

- Unlike tuples, lists have variable length and are mutable:

```
>>> list_1 = [2, 3, 7, None]
>>> tup = ( ‘a’, ‘b’, ‘c’ )
>>> list_2 = list( tup )
>>> list_2[1] = ‘d’
>>> list_2
[ ‘a’, ‘d’, ‘c’ ]
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Collections
Lists

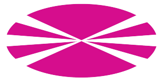
- Elements are added at the end of a list using `append()`:

```
>>> list_2.append( 5 )
>>> list_2
[ ‘a’, ‘d’, ‘c’, 5 ]
```

- Elements can also be added at a specific place using `insert()`:

```
>>> list_2.insert( 3, None )
>>> list_2
[ ‘a’, ‘d’, ‘c’, None, 5 ]
```

Ecole Internationale des Sciences du Traitement de l'Information

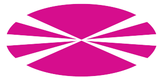


Python - Collections
Lists

- The reverse operation to `insert()` is `pop()`:

```
>>> list_2.pop()
5
>>> list_2
[ ‘a’, ‘d’, ‘c’, None ]
>>> list_2.pop(2)
‘c’
>>> list_2
[ ‘a’, ‘d’, None ]
```

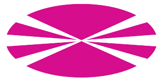
Ecole Internationale des Sciences du Traitement de l'Information



Python - Collections
Lists

- Elements can be deleted from the list using `remove()`:

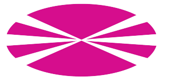
```
>>> list_2.append('a')
>>> list_2
[ 'a', 'd', None, 'a' ]
>>> list_2.remove( 'a' )
>>> list_2
[ 'd', None, 'a' ]
```

Python - Collections
Lists

- The `in` operator checks whether a value is contained in a list:

```
>>> 'a' in list_2
True
>>> 'c' in list_2
False
```

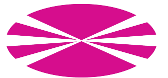
Python - Collections
Lists

- The operator + is used to concatenate lists:

```
>>> list_1 + list_2
[ 2, 3, 7, None, ‘d’, None, ‘a’ ]
```

- `extend()` adds full lists to a given one:

```
>>> list_1.extend( list_2 )
>>> list_1
[ 2, 3, 7, None, ‘d’, None, ‘a’ ]
```

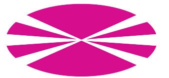



Python - Collections
Lists

- A list can be sorted in place using `sort()`:

```
>>> a = [ 7, 2, 5, 1, 3 ]
>>> a.sort()
>>> a
[1, 2, 3, 5, 7]
>>> b = [‘galicia’, ‘asturias’, ‘cantabria’, ‘euskadi’, ‘navarra’]
>>> b.sort()
>>> b
[ ‘asturias’, ‘cantabria’, ‘euskadi’, ‘galicia’, ‘navarra’ ]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

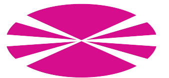


Python - Collections
Lists

- `sort()` accepts a function as sorting key:

```
>>> b = [ ‘caladan’, ‘arrakis’, ‘corrin’, ‘ix’, ‘giedi
prime’ ]
>>> b.sort( key = len )
>>> b
[ ‘ix’, ‘corrin’, ‘caladan’, ‘arrakis’, ‘giedi prime’ ]
```

Ecole Internationale des Sciences du Traitement de l'Information

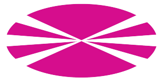


Python - Collections
Lists

- The `bisect` module manipulates sorted lists using binary search.
- It does not check that a list is actually sorted, using it on unsorted lists yields incorrect results.

```
>>> import bisect
>>> c = [1, 2, 2, 2, 3, 4, 7]
>>> bisect.bisect( c, 2 ) # Returns insertion index
4
>>> bisect.insort( c, 6 ) # Sorted insertion
>>> c
[1, 2, 2, 2, 3, 4, 6, 7]
```

Ecole Internationale des Sciences du Traitement de l'Information

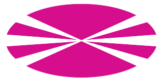


Python - Collections
Slicing

- Sections of indexed collections (such as tuples and lists) can be accessed using slice notation:

```
>>> a = [7, 2, 3, 7, 5, 6, 0, 1]
>>> a[1:5]
[2, 3, 7, 5]

>>> a[3:4] = [6, 3]
>>> a
[7, 2, 3, 6, 3, 5, 6, 0, 1]
```

Ecole Internationale des Sciences du Traitement de l'Information

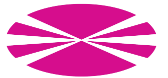


Python - Collections
Slicing

- The slice `[start:stop]` includes the element in the `start` position, but not the element in the `stop` position.
- Any of them may be omitted, in which case the start or ending element of the collection is used by default:

```
>>> a[:5]
[7, 2, 3, 6, 3]
>>> a[3:]
[6, 3, 5, 6, 0, 1]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

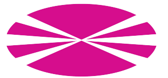


Python - Collections
Slicing

- Negative indices refer to the end of the array:

```
>>> a[-4:]
[5, 6, 0, 1]
>>> a[-6:-2]
[6, 3, 5, 6]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Collections
Slicing

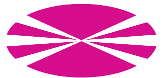
- The step can be modified using `[start:stop:step]`:

```
>>> a[::2]
[7, 3, 3, 6, 1]
```

- A particular use of the step is the reverse a sequence:

```
>>> a[::-1]
[1, 0, 6, 5, 3, 6, 3, 2, 7]
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Collections Manipulation

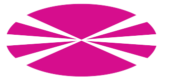
- Oftentimes we want to iterate the elements of a collection and their index at the same time:

```
i = 0
for x in collection:
    # do something with x, i
    i += 1
```

- This is equivalent to

```
for i, x in enumerate(collection):
    # do something with x, i
```

Ecole Internationale des Sciences du Traitement de l'Information

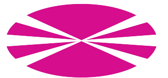


Python - Collections Manipulation

- `sorted()` returns a sorted list containing the elements of a collection:

```
>>> sorted( [7, 1, 2, 6, 0, 3, 2] )
[0, 1, 2, 2, 3, 6, 7]
>>> sorted( ‘test string’ )
[' ', 'e', 'g', 'i', 'n', 'r', 's', 's', 't', 't', 't']
>>> sorted( set( ‘test string’ ) )
[' ', 'e', 'g', 'i', 'n', 'r', 's', 't']
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



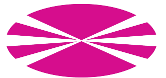
Python - Collections Manipulation

- `zip()` groups the elements in several sequences into a list of tuples:

```
>>> seq1 = [‘caladan’, ‘kaitain’, ‘giedi prime’,
‘arrakis’]
>>> seq2 = [‘atreides’, ‘corrino’, ‘harkonnen’]
>>> zip( seq1, seq2 )
[(‘caladan’, ‘atreides’), (‘kaitain’, ‘corrino’),
(‘giedi prime’, ‘harkonnen’)]
```

- The length of the resulting list is given by the length of the shortest input sequence.

Ecole Internationale des Sciences du Traitement de l'Information

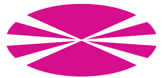


Python - Collections Manipulation

- `zip()` is commonly used to simultaneously iterate several sequences:

```
for i, (a,b) in enumerate( zip( seq1, seq2 ) ):
    # do something with i, a, b
```

Ecole Internationale des Sciences du Traitement de l'Information

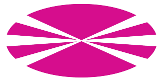


Python - Collections Manipulation

- `zip()` can be used to “unzip”:

```
>>> l = [ (‘caladan’, ‘atreides’), (‘kaitain’, ‘corrino’),
(‘giedi prime’, ‘harkonnen’) ]
>>> unzip1, unzip2 = zip( *l )
>>> unzip1
(‘caladan’, ‘kaitain’, ‘giedi prime’)
>>> unzip2
(‘atreides’, ‘corrino’, ‘harkonnen’)
```

Ecole Internationale des Sciences du Traitement de l'Information

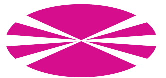


Python - Collections Manipulation

- `reversed()` builds an iterator over the elements of a sequence in reverse order:

```
>>> reversed( range(10) )
<listreverseiterator object at 0x7ff664159510>
>>> list( reversed( range(10) ) )
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

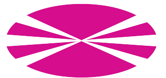
Ecole Internationale des Sciences du Traitement de l'Information



Python - Collections
Dictionaries

- A dictionary (`‘dict’` type) is an associative array (hashtable).
- It is similar to an array, but indexes an object using a key instead of an integer.

```
>>> empty_dict = {}
>>> d1 = { ‘a’: ‘a value’, ‘b’: [1,2,3,4] }
>>> d1
{‘a’: ‘a value’, ‘b’: [1,2,3,4]}
```

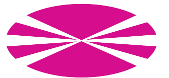



Python - Collections
Dictionaries

- The elements of a dictionary can be accessed and inserted using the same syntax as for lists and tuples:

```
>>> d1[7] = ‘an integer’
>>> d1
{‘a’: ‘a value’, ‘b’: [1,2,3,4], 7: ‘an integer’}
>>> d1[‘b’]
[1,2,3,4]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

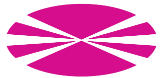


Python - Collections
Dictionaries

- It can be checked whether a key is contained in a dictionary using `in`:

```
>>> ‘b’ in d1
True
>>> 42 in d1
False
```

Ecole Internationale des Sciences du Traitement de l'Information

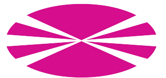


Python - Collections
Dictionaries

- To remove values from a dictionary either `del` or `pop()` can be used:

```
>>> del d1[‘a’]
>>> d1
{‘b’: [1,2,3,4], 7: ‘an integer’}
>>> d1.pop(‘b’)
[1,2,3,4]
>>> d1
{7: ‘an integer’}
```

Ecole Internationale des Sciences du Traitement de l'Information



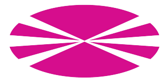
Python - Collections
Dictionaries

- `keys()` and `values()` return the keys and values stored in the dictionary:

```
>>> d1[5] = ‘another integer’
>>> d1.keys()
[5,7]
>>> d1.values()
[‘another integer’, ‘an integer’]
```

- The keys and values are not returned in any particular order, but the orders of both lists are consistent (i.e. `d1[d1.keys()[x]] = d1.values()[x]`).

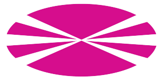
Ecole Internationale des Sciences du Traitement de l'Information



Python - Collections
Dictionaries

- Two dictionaries can be fused using `update()`:

```
>>> d1.update( { 'b': 'caladan', 'c': 'arrakis' } )
>>> d1
{'c': 'arrakis', 'b': 'caladan', 5: 'another integer',
7: 'an integer' }
```

Python - Collections
Dictionaries

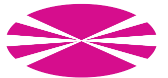
- A dictionary can be created from two lists:

```
mapping = {}
for key, value in zip( key_list, value_list ):
  mapping[key] = value
```

- Or:

```
>>> mapping = dict( zip( key_list, value_list ) )
```

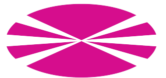
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Collections
Dictionaries

- In order to be usable as a dictionary key, a Python object must be "hashable".
- Basic types in Python are hashable, but not mutable containers (e.g., lists).
- A hashable object implements `__hash__()`, `__eq__()`, and `__cmp__()` such that:
  1. The return value of `__hash__()` does not change during the life of the object.
  2. If two objects are equal according to `__eq__()` they must share the same `__hash__()` value.
  3. `__cmp__()` must compare objects consistently.

Ecole Internationale des Sciences du Traitement de l'Information

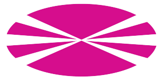


Python - Collections
Set

- A `set` is an unsorted collection of unique elements:

```
>>> set( [2,2,2,1,3,3] )
set([1, 2, 3])
>>> {2, 2, 2, 1, 3, 3}
set([1, 2, 3])
```

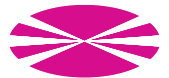
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Collections
Set

Method	Alternate syntax	Description
`a.add(x)`	--	Add `x` to set.
`a.remove(x)`	--	Remove `x` from set.
`a.union(b)`	`a | b`	Union of `a` and `b`.
`a.intersection(b)`	`a & b`	Intersection of `a` and `b`.
`a.difference(b)`	`a - b`	Set difference.
`a.symmetric_difference(b)`	`a ^ b`	Symmetric set difference.
`a.issubset(b)`	--	`True` if `b` is a subset of `a`.
`a.issuperset(b)`	--	`True` if `a` is a superset of `b`.
`a.isdisjoint(b)`	--	`True` if `a` and `b` are disjoint.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Collections

Comprehensions of lists, sets, and dictionaries

- Comprehensions are “syntactic sugar” to generate new collections by operating and filtering preexisting ones.
- The basic syntax is as follows:

```
>>> [expr for val in collection if condition]
```

equals:

```
new_list = []
for val in collection:
    if condition:
        new_list.append( expr )
```

Ecole Internationale des Sciences du Traitement de l'Information

Python - Collections

Comprehensions of lists, sets, and dictionaries

- The filtering condition may be omitted:

```
>>> strings = [‘a’, ‘an’, ‘the, ‘cat’, ‘car’, ‘pigeon’]
>>> [x.upper() for x in strings if len(x) > 2]
[‘THE’, ‘CAT’, ‘CAR’, ‘PIGEON’]
>>> [x.upper() for x in strings]
[‘A’, ‘AN’, ‘THE’, ‘CAT’, ‘CAR’, ‘PIGEON’]
```

Ecole Internationale des Sciences du Traitement de l'Information

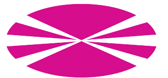
Python - Collections

Comprehensions of lists, sets, and dictionaries

- Using a similar syntax we can write comprehensions of sets and dictionaries:

```
{ key-expr: val-expr for value in collection if condition
}
```

```
{ set-expr for value in collection if condition }
```

Python - Collections

Comprehensions of lists, sets, and dictionaries

- We can write nested loops in a comprehension:

```
>>> tuples = ((1, 2, 3), (4, 5, 6), (7, 8, 9))
>>> [x for tup in tuples for x in tup if x > 3]
[4, 5, 6, 7, 8, 9]
```

- The order of the loops in a comprehension is the same as in an equivalent code:

```
>>> list = []
>>> for tup in tuples:
...     for x in tup:
...         if x > 3: list.append( x )
```

Ecole Internationale des Sciences du Traitement de l'Information

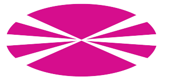
Python - Collections

Comprehensions of lists, sets, and dictionaries

- It is also valid to nest comprehensions:

```
>>> [[x for x in tup] for tup in tuples]
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

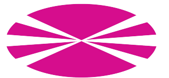


Python - Functions

- Functions are defined using the reserved word `def`.
- `return` is used to return control to the caller and pass the result.

```
def my_func( x, y, z = 1.5 ):
    if z > 1:
        return z * (x+y)
    else:
        return z / (x+y)
```

Ecole Internationale des Sciences du Traitement de l'Information



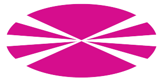
Python - Functions

- If the end of the function code is reached without executing any `return` instruction, `None` is returned automatically.
- Functions receive two types of parameters: positional and keyword.
- Keyword arguments are commonly used to provide default values to optional parameters.
- In the example, `x` and `y` are positional parameters, while `z` is a keyword parameter. The function can be called in different ways:

```
>>> my_func( 5, 6, z=0.7 )
6.14
```

```
>>> my_func( 3.14, 7, 3.5 )
17.990000000000002
```

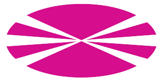
Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions

Name spaces, scopes, and local functions

- A function can access variables in two different scopes: global and local.
- Alternatively, variables can be explicitly defined inside a namespace.
- By default, variables assigned inside a function belong to the local scope.
- The local scope of a function is created when it is called, and initially contains its parameters.
- The local scope is destroyed when the function returns (except for closures, which we will briefly cover later).



Python - Functions

Name spaces, scopes, and local functions

```
def func():
    a = []
    for i in range(5):
        a.append( i )
```

- When `func()` is called, `a` is created. Then, the loop is executed and 5 integers are appended to `a`. Finally, the end of the function body is reached and `a` is destroyed.

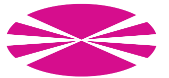
Ecole Internationale des Sciences du Traitement de l'Information

Python - Functions

Name spaces, scopes, and local functions

```
>>> a = []
>>> def func():
...     for i in range(5):
...         a.append( i )
>>> func()
>>> a
[0, 1, 2, 3, 4]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

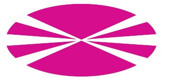


Python - Functions

Name spaces, scopes, and local functions

```
>>> a = []
>>> def func():
...     a = range(5)
>>> func()
>>> a
[]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

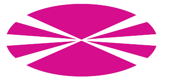


Python - Functions

Name spaces, scopes, and local functions

```
>>> a = []
>>> def func():
...     global a
...     a = range(5)
>>> func()
>>> a
[0, 1, 2, 3, 4]
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions

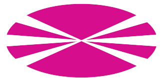
Name spaces, scopes, and local functions

- New functions can be declared anywhere in the code.
- In particular, it is legal to declare a function nested inside another function. These are called local functions, and are created when the enclosing function is called:

```
def outer_f( x, y, z ):
    def inner_f( a, b, c ):
        pass
    pass
```

- `inner_f()` does not exist until the call to `outer_f()`. As soon as `outer_f()` returns, `inner_f()` is destroyed.
- `inner_f()` can access the variables and functions in the local scope of `outer_f()`, but it cannot add anything to it.

Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions
Returning multiple values

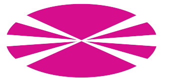
- A notable difference w.r.t. other languages such as C/C++/Java is the ability of a function to return multiple values.

```
def f():
    a = 5; b = 6; c = 7;
    return a, b, c
x, y, z = f()
```

- The implementation is very simple: `f()` is actually returning a single value, which happens to be a tuple.

```
>>> f()
(5, 6, 7)
```

Ecole Internationale des Sciences du Traitement de l'Information



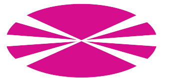
Python - Functions
Functions as objects

- A Python function is just a special type of object which defines the `()` operator.
- As such, it is possible to use functors (pointers to function objects) to code complex operations in a simple way.
- E.g., we want to perform cleanup operations on the strings in the following array:

```
>>> planets = [ ‘    Caladan ’, ‘Ix!’, ‘Ix’, ‘ix’,
‘aRraKIs’, ‘giedi  prime##’, ‘Salusa secundus?’ ]
```

- We want to build a list of uniform strings for its analysis. We need to apply removal of unnecessary spaces and symbols, and to fix capitalization.

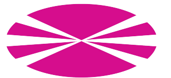
Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions
Functions as objects

```
import re # Regular Expression module

def clean_strings( strings ):
    result = []
    for value in strings:
        value = value.strip()
        value = re.sub(‘[!#?]’, ‘’, value) # removes !#? symbols
        value = value.title()
        result.append( value )
    return result
```

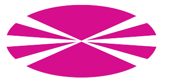
Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions
Functions as objects

```
def clean_strings_v2( strings, ops ):
    result = []
    for value in strings:
        for function in ops:
            value = function( value )
        result.append( value )
    return result

def remove_punctuation( value ):
    return re.sub( “[!#?]”, “”, value )

clean_ops = [ str.strip, remove_punctuation, str.title ]
```

Ecole Internationale des Sciences du Traitement de l'Information

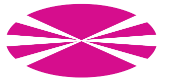


Python - Functions
Functions as objects

- It is possible to pass functions as arguments to other functions.
- E.g., Python provides mechanisms to apply a function to a list of objects:

```
>>> map( str.strip, planets )
[‘Caladan’, ‘Ix!’, ‘Ix’, ‘ix’, ‘aRraKIs’, ‘giedi prime##’,
‘Salusa secundus?’]
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

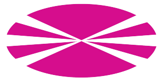


Python - Functions
Anonymous (λ) functions

- An anonymous function, or lambda function, is a single-instruction functional expression the result of which is its return value:

```
def short_function(x):
    return x*2

equiv_anon = lambda x: x*2
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

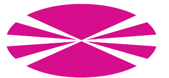


Python - Functions
Anonymous (λ) functions

- Oftentimes it is simpler to use a reference to a lambda function than to write an ad-hoc, named one:

```
>>> a = [4, 0, 1, 5, 6]
>>> map( lambda x: x*2, a )
[8, 0, 2, 10, 12]


>>> strings = [‘caladan’, ‘ix’, ‘corrin’, ‘giedi prime’]
>>> strings.sort( key = lambda x: len(set(list(x))) )
# Sort by number of different letters in the string
>>> strings
[‘ix’, ‘caladan’, ‘corrin’, ‘giedi prime’]
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions
Closures

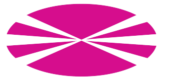
- A closure is a dynamically-generated function which is returned by another function.

- The distinguishing characteristic of a closure is that it is capable of accessing the local scope of its generating function after the latter returns.

Python - Functions
Closures

```
>>> def make_closure( a ):
...     def closure():
...         print( “Variable in the local scope: %d” % a )
...     return closure

>>> closure = make_closure(5)
>>> closure()
‘Variable in the local scope: 5’
```

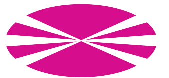
Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions
Closures

- In the example, a closure was created with an immutable internal state (the integer `a`).

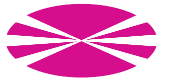
- Mutable variables can also be used in a closure. These can be modified, dynamically altering the behavior of the closure.



Python - Functions
Closures

```
>>> def make_watcher():
...     have_seen = set([])
...     def has_been_seen( x ):
...         if x in have_seen: return True
...         else: have_seen.add(x)
...         return False
...     return has_been_seen
>>> watcher = make_watcher()
>>> vals = [5, 6, 1, 5, 1, 6, 3, 5]
>>> [watcher(x) for x in vals]
[False, False, False, True, True, True, False, True]
```

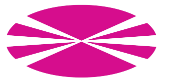
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Functions
Closures

- The local variables of a closure can be **modified**.
- No new variables can be **added** to the scope of a closure. A workaround is to add key/value pairs to a dictionary in the scope.
- Closures allow to build generic functions with plenty of options, that can be dynamically instantiated into specialized, efficient and simple variants.

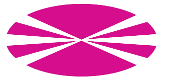
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Functions
Closures

```
>>> def format_and_pad( template, space ):
...     def formatter( x ):
...         return (template % x).rjust( space )
...     return formatter
>>> fmt = format_and_pad( “%.4f”, 15 )
>>> fmt(1.756)
'         1.7560'
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Functions

Extended syntax: *args, *kwargs

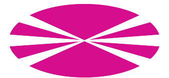
- Functions are called using a mix of positional and keyword parameters:

```
>>> func( a, b, c, d = d_value, e = e_value )
```

- Internally, this function:
  1. Receives an `args` tuple containing its positional parameters.
  2. Receives a `kwargs` dictionary containing its keyword parameters:
  3. Performs the following assignment:

```
>>> (a, b, c) = args
>>> d = kwargs.get( ‘d’, d_default_value )
>>> e = kwargs.get( ‘e’, e_default_value )
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions

Extended syntax: *args, *kwargs

```
def g( x, y, z=1 ): return (x+y) / z
def hello_world_then_call( f, *args, *kwargs ):
    print ‘args is’, args
    print ‘kwargs is’, kwargs
    print “Hello world! Now I’m going to call %s” % f
    return f( *args, *kwargs )
>>> hello_world_then_call( g, 1, 2, z=5 )
args is (1, 2)
kwargs is {‘z’: 5.0}
Hello world! Now I’m going to call <function g at 0x2dd5cf8>
0.6
```

Ecole Internationale des Sciences du Traitement de l'Information

Python - Functions

Partial function application

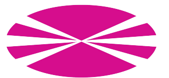
- Partial function application consists in creating new functions from preexisting ones by fixing some of their parameters:

```
>>> def add(x, y): return x+y
>>> add_5 = lambda y: add(5,y)
```

- Alternatively:

```
>>> from functools import partial
>>> add_5 = partial( add, 5 )
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions
Generators

- A generator is a function which returns a sequence of values in a lazy way, stopping its execution after each value in the sequence.

- Generators are useful to generate large, iterable sequences in a memory-efficient way (e.g., `range()` vs `xrange()`).

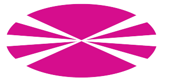
- Generators are declared as a function which returns a value using `yield` instead of `return`.

Ecole Internationale des Sciences du Traitement de l'Information

Python - Functions
Generators

```
>>> def squares( n = 10 ):
...     for i in xrange( 1, n+1 ):
...         print “Generating squares from 1 to %d”%(n**2)
...         yield i ** 2
>>> gen = squares()
>>> gen
<generator object squares at 0x7fd9e3e796e0>
```

Ecole Internationale des Sciences du Traitement de l'Information

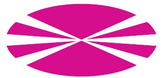


Python - Functions
Generators

- When the generator function is called, no code is executed.
- Each element must be explicitly requested:

```
>>> for x in gen:
...     print x
Generating squares from 1 to 100
1
Generating squares from 1 to 100
4
.
.
.
```

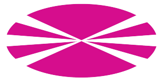
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python - Functions
Generators

```
def make_change( amount, coins=[1, 2, 5, 10, 20, 50], hand=[] ):
    if amount == 0: yield hand
    for coin in coins:
        if coin > amount or (len(hand) > 0 and hand[-1] < coin): continue
        for result in make_change(amount-coin, coins=coins, hand=hand+[coin]):
            yield result

>>> len(list(make_change(53)))
530
```

Ecole Internationale des Sciences du Traitement de l'Information



Python - Functions
Generator expressions

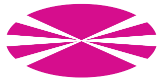
- A compact way to declare a simple generator is to use a generator expression:

```
>>> gen = ( x ** 2 for x in xrange(100) )
>>> gen
<generator object <genexpr> at 0x7fd9e3e798c0>
```

- This expression is equivalent to:

```
def gen():
    for x in xrange(100):
        yield x**2
```

Ecole Internationale des Sciences du Traitement de l'Information



Python
Files and Operating System

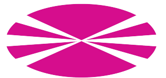
- To open a file, `open()` is called passing a relative or absolute path:

```
>>> f = open( ‘folder/file.txt’ )
```

- By default, the file is opened in read-only mode (`‘r’`). The file can be seen as a collection of lines, and iterated using a `for` loop:

```
>>> for line in f:
...       # do something with line
```

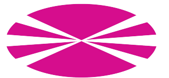
Ecole Internationale des Sciences du Traitement de l'Information



Python
Files and Operating System

Opening mode	Description
`r`	Read only.
`w`	Write only. Creates a new file, overwriting any previous one.
`a`	Concatenate to a file (created if it does not exist).
`r+`	Read-write.
`b`	Binary mode (usage example: `'rb'`).
`U`	Use universal end-of-line mode. Translates any end-of-line marker in the file to `'\n'`.

Ecole Internationale des Sciences du Traitement de l'Information



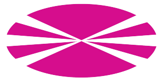
Python
Files and Operating System

- To write to a file we can use the methods `write()` or `writelines()`:

- E.g., to remove white lines from a file:

```
>>> f_in = open( path, ‘r’)
>>> f_out = open( ‘tmp.txt’, ‘w’ )
>>> f_out.writelines( [x for x in f_in if len(x) > 1] )
```

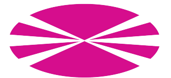
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Python
Files and Operating System

Method	Description
`read( [size] )`	Reads data from the file as a string. The optional argument `[size]` is the number of bytes to read. Without it, the entire file is read.
`readlines( [size] )`	Same as `read()`, but returns a list of strings (one per line in the file). Without `[size]` the entire file is read.
`write( str )`	Writes `str` to the file.
`writelines( str )`	Writes a list of strings to the file, one per line.

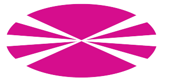
Ecole Internationale des Sciences du Traitement de l'Information



Python
Files and Operating System

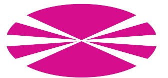
Method	Description
`close()`	Closes the file.
`flush()`	Synchronizes the I/O buffer to disk.
`seek( pos )`	Moves the file pointer to `pos`.
`tell()`	Returns the current position of the file pointer.
`closed`	`True` if the file is closed.

Ecole Internationale des Sciences du Traitement de l'Information



iPython

- Alternative interactive Python interpreter.
- Does not provide any analysis or computation tool beyond those in cPython, but greatly simplifies some repetitive tools and the general development process.
- It includes GUI environments to simplify data visualization, e.g., it provides a web interface to Python, allowing to save HTML sessions which can then be shared (called Notebooks, we will use them extensively in this course).



iPython

```
$ ipython2

Python 2.7.13 (default, Jul 21 2017, 03:24:34)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

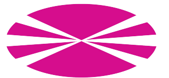
Ecole
Internationale
des Sciences
du Traitement
de l'Information

iPython - Features
Pretty printing

```
In [1]: a = 5

In [2]: a
Out[2]: 5

In[3]: from numpy.random import randn

In[4]: data = { i: randn() for i in range(7)}
```

```
In[4]: data
{0: -2.171772092191186,
 1: 0.2947739898361525,
 2: -0.3602586072105093,
 3: 1.300393521182511,
 4: -0.0003341635712392460 7,
 5: -0.572174661598875,
 6: 0.8141305611469151}
```

Ecole Internationale des Sciences du Traitement de l'Information

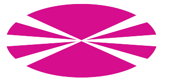


iPython - Features
Enumerated prompt

- iPython provides a prompt in which inputs and outputs are identified by an integer. This integer can be used to recover both:

```
In[5]: _i3
Out[5]: u'data = { i: randn() for i in range(7) }'

In[6]: _4
{0: -2.171772092191186,
 1: 0.2947739898361525,
...
```

Ecole Internationale des Sciences du Traitement de l'Information

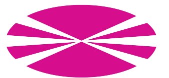


iPython - Features
Autocompletion

- One of the improvements in usability w.r.t. cPython is the tab completion mechanism.
- When typing a command, pressing <Tab> activates a search of the current namespace, trying to identify any variable (objects, functions, etc.) whose name matches what has been typed so far:

```
In[1]: a_number = 1
In[2]: a_letter = 'a'
In[3]: a<Tab>
a_letter   a_number    abs           %alias       %alias_magic
all        and         any           apply        as
assert     %autocall   %autoindent   %automagic
```

Ecole Internationale des Sciences du Traitement de l'Information

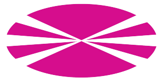


iPython - Features
Autocompletion

- Besides finding the variables `a_letter` and `a_number`, in the previous example there are other valid completions, such as the magic commands `%alias` or `%autoindent`, as well as the operator `and` or the function `any()`.
- The autocompletion mechanism also allows to explore the methods and attributes of an object:

```
In[3]: b = [1,2,3]

In[4]: b.<Tab>

b.append    b.count     b.extend    b.index     b.insert
b.pop       b.remove    b.reverse   b.sort
```

Ecole Internationale des Sciences du Traitement de l'Information

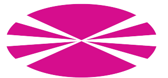


iPython - Features
Autocompletion

- It is also possible to use autocompletion with the scope of a module:

```
In[5]: import datetime
In[6]: datetime.<Tab>
datetime.MAXYEAR     datetime.MINYEAR          datetime.date
datetime.datetime    datetime.datetime_CAPI    datetime.time
datetime.timedelta   datetime.tzinfo
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information

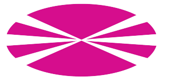


iPython - Features
Autocompletion

- By default, methods and attributes starting with an underscore are hidden.
- To show them, we must include the underscore explicitly:

```
In[7]: datetime._<Tab>

datetime.__class__              datetime.__dict__
datetime.__file__               datetime.__getattribute__
datetime.__init__               datetime.__new__              ...
```

Ecole Internationale des Sciences du Traitement de l'Information

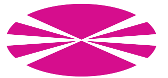


iPython - Features
Autocompletion

- Besides looking for variables and functions, the autocompletion mechanism also searches file names when the interpreter deems it appropriate.

- It is also useful to explore keyword parameter names.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

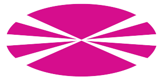


iPython - Features
Introspection

- Typing a question mark before or after a name returns information about it:

```
In[1]: b = [1,2,3]
In[2]: b?
Type:         list
String form: [1, 2, 3]
Length:       3
Docstring:
list() -> new empty list
list(iterable) -> new list initialized from iterable's items
```

Ecole Internationale des Sciences du Traitement de l'Information



iPython - Features
Introspection

- If the name to be inspected is a function, it prints its docstring if available:

```
In[1]: import numpy as np
In[2]: np.linalg.det?
Type:         function
String form: <function det at 0x7f27c44b4ed8>
File:         /usr/lib/python2.7/site-packages/numpy/linalg/linalg.py
Definition:   numpy.linalg.det(a)
Docstring:
Compute the determinant of an array.
```

Ecole Internationale des Sciences du Traitement de l'Information

iPython - Features
Introspection

```
Parameters
----------
a : (..., M, M) array_like
    Input array to compute determinants for.

Returns
-------
det : (...) array_like
    Determinant of `a`.

...
```

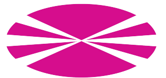
Ecole
Internationale
des Sciences
du Traitement
de l'Information

iPython - Features
Introspection

- If two question marks are used (e.g., `numpy.linalg.det??`), the source code of the function is shown, if available.
- The question mark character also allows to search a namespace using regular expressions:

```
In[3]: np.linalg.*rank*?
numpy.linalg.matrix_rank
```

Ecole Internationale des Sciences du Traitement de l'Information



iPython - Magic commands

- iPython defines many so-called “magic commands”, designed to simplify frequent tasks.
- The syntax of magic commands is `%magicname`.
- They behave like programs which are interpreted by iPython.
- Some magic commands receive arguments.

```
In[1]: %reset?
```

…

```
Resets the namespace by removing all names defined by the user
...
```

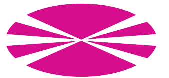
Ecole Internationale des Sciences du Traitement de l'Information

iPython - Magic commands
%run

- `%run` executes Python code in a given file:

```
-------------------------------------> test.py
print "Hello world!"
-------------------------------------


In[1]: %run test.py
Hello world!
```

Ecole
Internationale
des Sciences
du Traitement
de l'Information



iPython - Magic commands

%run

- The file is executed inside an empty namespace. As such, the execution is equivalent to a command-line invocation.
- However, all the names defined in the script (imports, functions, global variables) will be accessible by the iPython environment after the `%run`.
- If the script expects arguments, this can be provided as usual:

```
In[1]: %run script.py <args>
```

Ecole Internationale des Sciences du Traitement de l'Information

iPython - Magic commands

`%paste / %cpaste`

- A quick way to execute code is to copy-paste from a web page or other sources.
- Using <Ctrl-C> and <Ctrl-V> is tricky with Python, due to the special indentation semantics.:

```
x=5
y=7
if x > 5:
    x += 1

    y += 8
```

- The white line in the previous code prevents correct execution: some interpreters throw an `IndentationError`, while others execute `y += 8` outside the conditional block.

Ecole Internationale des Sciences du Traitement de l'Information

iPython - Magic commands

`%paste / %cpaste`

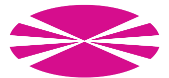
- `%paste` and `%cpaste` allow to copy-paste code without worrying about indentation.
- `%paste` executes the text in the clipboard automatically.
- `%cpaste` is similar, but allows to edit the clipboard text before its execution.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

iPython - Magic commands

Command	Description
`%quickref`	Show a quick reference sheet.
`%magic`	Print information about the magic function system.
`%debug`	Activate the interactive debugger using the top of the stack from the last error.
`%hist`	Print input history.

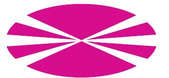
Ecole
Internationale
des Sciences
du Traitement
de l'Information



iPython - Magic commands

Command	Description
%pdb	Control the automatic calling of the pdb interactive debugger.
%paste	Paste & execute a pre-formatted code block from clipboard.
%cpaste	Opens a special prompt to copy and format code manually.
%reset	Resets the namespace by removing all names defined by the user.
%page <object>	Pretty print an object and display it through a pager.

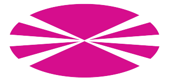
Ecole Internationale des Sciences du Traitement de l'Information



iPython - Magic commands

Command	**Description**
%run script.py	Executes the script in file script.py.
%prun <instruction>	Executes <instruction> using cProfile (Python profiler).
%time <instruction>	Measures the execution time of <instruction>.
%timeit <instruction>	Executes <instruction> multiple times and measures its execution time. Useful to time commands with short executions.

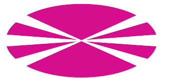
Ecole Internationale des Sciences du Traitement de l'Information



iPython - Magic commands

Command	Description
`%who / %who_ls / %whos`	Print all interactive variables with different verbosity levels.
`%xdel <variable>`	Deletes `<variable>`.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



iPython - Shortcuts

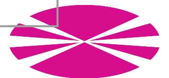
Command	Description
<Ctrl+P> / <cursor up>	Search backwards in the command history, taking into account the current prompt.
<Ctrl+N> / <cursor down>	Search forward in the command history, taking into account the current prompt.
<Ctrl+R>	Search backwards in the command history in the readline style.
<Ctrl+Shift+V>	Paste text from the clipboard.
<Ctrl+C>	Interrupt current interactive execution.

Ecole Internationale des Sciences du Traitement de l'Information

iPython - Shortcuts

Command	Description
<Ctrl+A>	Moves cursor to the beginning of the current line.
<Ctrl+E>	Moves cursor to the end of the current line.
<Ctrl+K>	Removes text from the cursor up to the current line end.
<Ctrl+U>	Removes all text from the current line.
<Ctrl+F> / <cursor derecho>	Move cursor forward one character.
<Ctrl+B> / <cursor izquierdo>	Move cursor backward one character.
<Ctrl+L>	Clear screen.

Ecole Internationale des Sciences du Traitement de l'Information



iPython - Command history
Search and reuse

- Using <cursor up> and <cursor down> it is possible to explore the command history. Contrary to the cPython interpreter, iPython takes into account the current prompt to restrict the search.
- Using <Ctrl+R> the current prompt is searched backwards in the command history. To discard a result and continue searching, press <Ctrl+R> again.

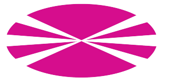
```
In[1]: a_command = f( x, y, z )
(reverse-i-search)'com': a_command = f( x, y, z )
```

Ecole Internationale des Sciences du Traitement de l'Information

iPython - Command history
Input and output

- iPython stores both the input and output command history.
- The last two outputs are stored in variables _ (an underscore) and __ (two underscores), respectively.
- Input lines are stored in a set of variables named `_iX`, where `X` is the sequence number of the input. In a similar way, a set of variables named `_X` stores the corresponding outputs.
- **Warning**: these logs reference all the outputs of the interactive session, and so they will never be garbage collected. This may result in memory issues, particularly in long sessions, which must be manually solved through `%xdel` and/or `%reset`.

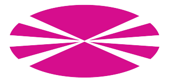
Ecole Internationale des Sciences du Traitement de l'Information



iPython - Command history
Input and output logs

- iPython allows to record an interactive session, including inputs and outputs.
- The log is activated using `%logstart`.
- It is deactivated using `%logoff`.
- The log can be temporarily paused and reactivated using `%logstop` and `%logon`, respectively.
- `%logstate` shows the current log state.

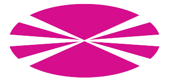
Ecole Internationale des Sciences du Traitement de l'Information



iPython - Interacting with the OS

Command	Description
`!cmd`	Executes `cmd` in an OS shell.
`output = !cmd`	Executes `cmd` and stores its standard output into variable `output`.
`%alias alias_name cmd`	Defines an alias for command `cmd` with the name `alias_name`.
`%bookmark`	Folder bookmarking system in iPython.
`%cd <dir>`	Changes the working directory to `<dir>`.

Ecole Internationale des Sciences du Traitement de l'Information



iPython - Interacting with the OS

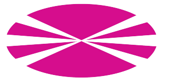
Command	Description
%pwd	Prints the current working directory.
%pushd <dir>	Stacks the current working directory and changes it to <dir>.
%popd	Changes the current working directory to the top of the stack and pops it.
%dirs	Shows the current working directory stack.
%dhist	Shows the list of visited directories.
%env	Returns a dictionary with the environment.

Ecole Internationale des Sciences du Traitement de l'Information

iPython - Software tools
ipdb debugger

- The ipdb debugger included with iPython improves over the pdb standard by including autocompletion, syntax highlighting, and more verbose exception info.
- There are several ways to invoke ipdb:
  1. %debug enters the last error location.
  2. Activating the automatic debugging using %pdb.
  3. Executing %debug <command>.
  4. %run -d script.py launches the script through ipdb.
  5. Executing import ipdb; ipdb.run(“<command>”)
  6. Inserting a call to ipdb.set_trace() at any point in a Python script.

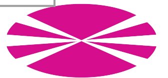
Ecole
Internationale
des Sciences
du Traitement
de l'Information



iPython - Software tools
ipdb commands

Command	Description
h(elp)	Shows the available commands.
help <command>	Shows the documentation available for <command>.
c(ontinue)	Resumes program execution.
q(uit)	Finishes the debugging session.
b(reak) number	Sets a breakpoint at line number of the current file.
b path:number	Sets a breakpoint at line number of file path.

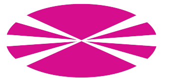
Ecole Internationale des Sciences du Traitement de l'Information



iPython - Software tools
ipdb commands

Command	Description
`s(tep)`	Execute the current line, stop at the first possible occasion (either in a function that is called or in the current function).
`n(ext)`	Continue execution until the next line in the current function is reached or it returns.
`u(p) / d(own)`	Move the current frame one level up / down in the stack trace (to an older / newer frame).
`a(rgs)`	Print the arguments of the current function.

Ecole Internationale des Sciences du Traitement de l'Information



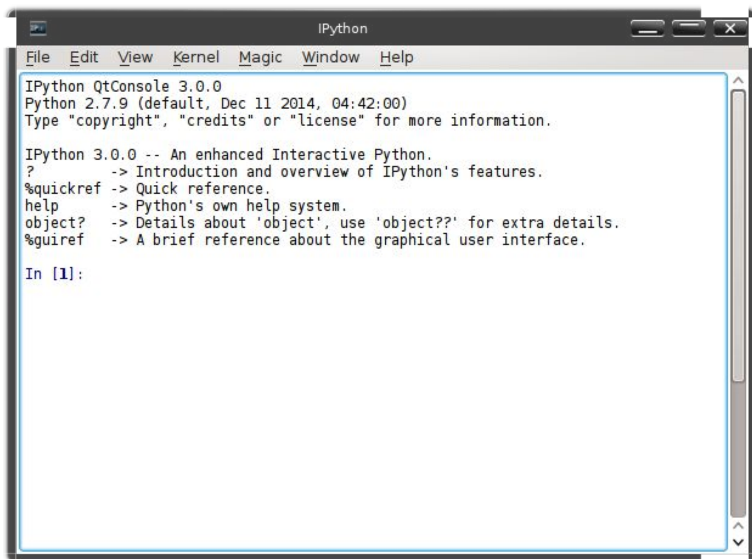
iPython - Software tools
ipdb commands

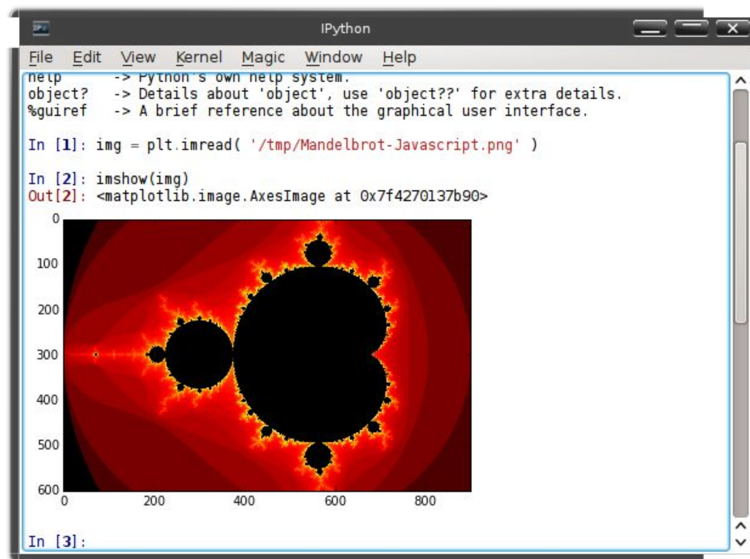
Command	Description
`debug instruction`	Enter a recursive debugger that steps through the code argument (which is an arbitrary expression or statement to be executed in the current environment).
`l(ist) [first [,last]]`	List source code for the current file, either around the current line or between lines `first` and `last`.
`w(here)`	Print the current stack trace.

Ecole Internationale des Sciences du Traitement de l'Information

iPython - Software tools
QT console

```
$ ipython2 qtconsole --pylab=inline
```





Ecole Internationale des Sciences du Traitement de l'Information

iPython - Software tools
QT console

- The QT console simplifies some tasks:
    - It allows to open several windows attached to a single iPython kernel.
    - Automatically shows the documentation of functions when interactively writing code.
- The `--pylab` option enables several environment changes. The underlying idea is to automatically get an analysis and development GUI similar to Matlab:
    - It loads `matplotlib` to draw plots.
    - Automatically imports `numpy`.
    - Enables support for embedded plots with `--inline`.

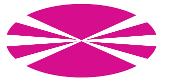
Ecole
Internationale
des Sciences
du Traitement
de l'Information

iPython - Software tools
HTML Notebook

- iPython also includes an HTML environment known as Notebook.
- It uses a JSON-based format to share code, outputs and plots.
- It runs on a web server which can be accessed using a regular browser.
- Allows to use a remote iPython interpreter easily through the web.
- To launch a server/client pair locally, we run:

```
$ ipython2 notebook
```

Ecole Internationale des Sciences du Traitement de l'Information



iPython - Software tools
HTML Notebook

Home - Konqueror

Ficheiro Editar Vista Ir a Marcadores Ferramentas Configuración Xanela Axuda

http://localhost:8889/tree

jupyter

Files Running Clusters

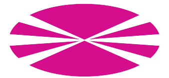
To import a notebook, drag the file onto the listing below or **click here.**

New

- Downloads
- HiPEAC.pdf

Páxina cargada.

Ecole Internationale des Sciences du Traitement de l'Information



iPython - Software tools
HTML Notebook

Notebook-001.ipynb

Untitled - Chromium

localhost:8888/notebooks/Untitled.ipynb?kernel_name=python2

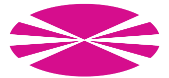
jupyter Untitled Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Help | Python 2

Cell Toolbar: None

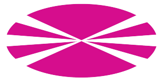
```
In [1]: %pylab inline
        Populating the interactive namespace from numpy and matplotlib

In [2]: img=plt.imread( '/tmp/Mandelbrot-Javascript.png' )

In [3]: imshow(img)
Out[3]: <matplotlib.image.AxesImage at 0x7f847692b410>

In [ ]:
```

Ecole Internationale des Sciences du Traitement de l'Information



NumPy

- NumPy (***Num**erical **Py**thon*) is a fundamental package that enables efficient array and vector processing.
- It provides:
  - An `ndarray` class, which represents a multidimensional array and provides *vectorized* arithmetic operations and *broadcasting* capabilities.
  - *Vectorized* operations are applied to whole arrays (without using Python loops).
  - Operations for lineal algebra, randomness, signal processing, …
  - Tools for the effective integration of Python and C/C++/Fortran.

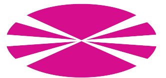


NumPy - `ndarray`

`Notebook-002.ipynb`

- n-dimensional array object.
- Allows to perform operations on large data blocks with scalar syntax.
- Unlike lists, a NumPy array is usually homogeneous: it holds objects of a specific, predefined type.
- An `ndarray` contains, among others, the following two properties:
  - `shape`: tuple containing the array dimensions.
  - `dtype`: instance of the `dtype` class which specifies element datatype.

Ecole Internationale des Sciences du Traitement de l'Information

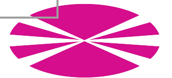


NumPy - `ndarray` Creation

Notebook-002.ipynb

Function	Description
`array()`	Builds a new array from the input sequence (e.g., a list). Copies input data to the new array.
`asarray()`	Converts the input sequence to an array, but it does not copy data if the input is already an `ndarray`.
`arange()`	Same as `range()` but returning an `ndarray` instead of a list.
`ones()` `ones_like()`	Build an array containing all `1s` from a tuple specifying the desired dimensions, or copying `shape` and `dtype` from another array.

Ecole Internationale des Sciences du Traitement de l'Information

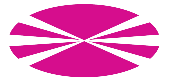


NumPy - `ndarray` Creation

`Notebook-002.ipynb`

Función	Descripción
`zeros()` `zeros_like()`	Build an array containing all `0s` from a tuple specifying the desired dimensions, or copying `shape` and `dtype` from another array.
`empty()` `empty_like()`	Build an array without initializing its data from a tuple specifying the desired dimensions, or copying `shape` and `dtype` from another array.
`eye()`, `identity()`	Build an array with `1s` in its main diagonal and `0s` elsewhere. `identity()` requires square dimensions.

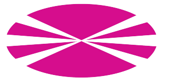
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - `ndarray` Datatypes

Tipo	Código	Descripción
`int8, uint8`	`i8, u8`	Signed/unsigned integer, 8 bits (1 byte).
`int16, uint16`	`i16, u16`	Signed/unsigned integer, 16 bits (2 bytes).
`int32, uint32`	`i32, u32`	Signed/unsigned integer, 32 bits (4 bytes).
`int64, uint64`	`i64, u64`	Signed/unsigned integer, 64 bits (8 bytes).

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - `ndarray`
Datatypes

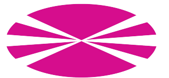
Tipo	Código	Descripción
`float16`	`f2`	Floating point, half precision (2 bytes).
`float32`	`f4` o `f`	Floating point, single precision (4 bytes). Compatible with a C `float`.
`float64`	`f64` o `d`	Floating point, double precision (8 bytes). Compatible with a C `double` and with a Python `float`.
`float128`	`f16` o `g`	Floating point, extended precision (16 bytes).

Ecole Internationale des Sciences du Traitement de l'Information

NumPy - `ndarray`
Datatypes

Tipo	Código	Descripción
`complex64`	`c8`	Complex number represented as 2 x `float32`.
`complex128`	`c16`	Complex number represented as 2 x `float64`.
`complex256`	`c32`	Complex number represented as 2 x `float128`.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - ndarray
Datatypes

Tipo	Código	Descripción
bool	?	Boolean type (True or False).
object	O	Python object (reference).
string_	S_	String type with fixed length (1 byte per character). E.g., S10 represents a 10-character string.
unicode_	U_	Unicode type with fixed length (the number of bytes per character varies per platform). Similar to string_.

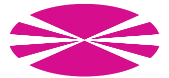
Ecole Internationale des Sciences du Traitement de l'Information

NumPy - `ndarray`
Datatypes

Notebook-002.ipynb

- `ndarray` can be cast to a different datatype using `astype()`.
- The call to `astype()` always creates a new copy of the array memory, even if the data is cast to its current type.
- If the cast fails for any reason, a `TypeError` exception is raised.

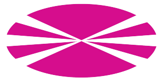
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - `ndarray`
Broadcasting

- When two arrays are operated, NumPy must decide what operation to perform in the first place.
- If both arrays have the same dimensionality, the operation is performed in an element-wise fashion.
- Otherwise, NumPy tries to *broadcast* (replicate) the "smallest" array to match it to the largest one.

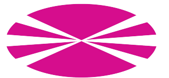
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - `ndarray`
Broadcasting

- The broadcasting process begins by the innermost dimension (the rightmost ones), and moves towards the outer ones (to the left).
- Two dimensions are compatible if:
  1. Both have the same number of elements, or
  2. One of them has a single element.
- If none of the previous conditions is met, the operation raises a `ValueError`.
- If the second condition is met, the array with a single element in that particular dimension is copied `n` times to match both.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - `ndarray`
Broadcasting

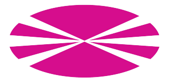
`Notebook-002.ipynb`

- The arrays are not required to have the same number of dimensions. Non-existing dimensions are assumed to have a single element.
- For example:

```
A  : 256 x 256 x 3
B  :             3
A*B: 256 x 256 x 3
```

```
A  : 256 x 256 x 3
B  : 256 x   1 x 3
A*B: 256 x 256 x 3
```

Ecole Internationale des Sciences du Traitement de l'Information

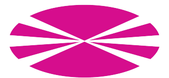


NumPy - `ndarray`
Basic indexing

`Notebook-002.ipynb`

- 1-dimensional arrays behave in a similar way to lists.
- An important difference is that in NumPy a slice is not a copy, but a *view* over the array data. This improves performance, but also means that slice operations may have collateral effects on the original array.
- It is possible to obtain a copy of a slice (or any array) by calling the `copy()` method of `ndarray`.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - `ndarray`

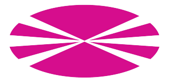
Basic indexing

`Notebook-002.ipynb`

The elements in each index of an n-dimensional array are not scalars, but (n-1)-dimensional arrays.

axis 1

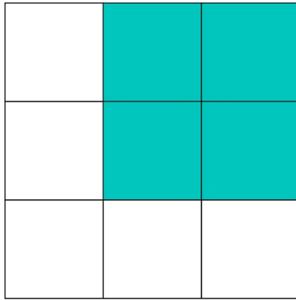
axis 0	0	1	2
0	0, 0	0, 1	0, 2
1	1, 0	1, 1	1, 2
2	2, 0	2, 1	2, 2



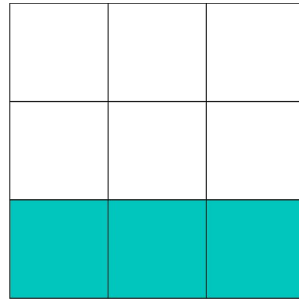
NumPy - `ndarray`
Indexing with slices

`Notebook-002.ipynb`

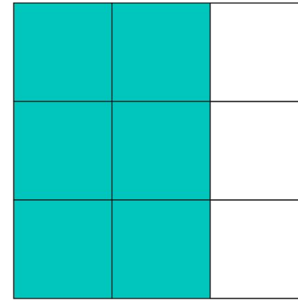
- The slice syntax is also valid for indexing arrays.
- Working with n-dimensional arrays, it is possible to use any combination of basic indexing and slicing.



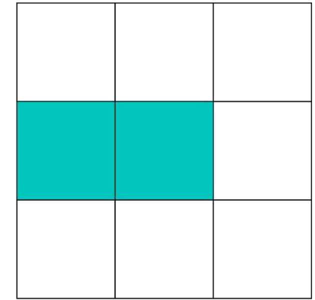
`arr[:2, 1:]`
`shape = (2,2)`



`arr[2]`
`shape = (3,)`

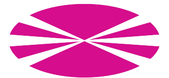


`arr[:,:2]`
`shape = (3,2)`



`arr[1:2,:2]`
`shape = (1,2)`

Ecole Internationale des Sciences du Traitement de l'Information

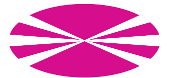


NumPy - `ndarray` Boolean indexing

`Notebook-002.ipynb`

- It is possible to index an array using a boolean array.
- The boolean array must have the same dimensionality as the axis it must index.
- Boolean indexing can be used to select elements in an array using data semantics (e.g., the elements which are greater than a particular number). The conditions do not need to refer to the array being indexed.
- Boolean indexing always returns a copy of the data (not a view).
- When accessing an n-dimensional array, boolean indexing can be mixed together with basic and sliced indexing.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - `ndarray`
Fancy indexing

`Notebook-002.ipynb`

- The term *fancy indexing* describes the indexing of an array using an integer collection containing the indices of the elements to select.
- If a list of lists is provided, the behavior changes: a 1D array is returned, containing the data indexed by the tuples resulting from applying `zip()` to the input lists.
- Fancy indexing always returns a copy of the data, not a view.
- When accessing an n-dimensional array, fancy indexing can be mixed together with basic, sliced, and boolean indexing.

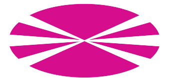
Ecole
Internationale
des Sciences
du Traitement
de l'Information

NumPy - `ndarray` Transposition

`Notebook-002.ipynb`

- `ndarray` provides the `transpose()` method and the `T` attribute.
- For n-dimensional arrays, `transpose()` accepts as input a tuple specifying an arbitrary dimensional permutation.
- The T attribute is a quick way of accessing “clasic” transposition, in which a matrix is flipped over its main diagonal.
- The `swapaxes()` is used to swap any two axes.
- All these operations return a view over the original array.

Ecole Internationale des Sciences du Traitement de l'Information

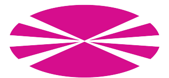


NumPy - Universal functions

`Notebook-002.ipynb`

- A universal function, or `ufunc`, performs element-wise operations over an `ndarray`.
- The term describes a wrapper over a simple function which reads one or more scalars and returns a single scalar.
- Many `ufunc`s take a single element as input (e.g., `sqrt()` or `exp()`). These are *unary* `ufunc`s.
- Other `ufunc`s take two elements and return one (*binary* `ufunc`s, e.g., `add()` or `maximum()`).
- Some `ufunc`s return more than one array (e.g., `modf()` returns the integral and fractional parts of a floating point array).

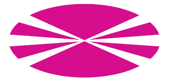
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Universal functions
Unary `ufunc`s

Function	Operation
`abs, fabs`	Absolute value of integers, floating point or complex numbers. `fabs()` is a faster version for non-complex numbers.
`sqrt`	Square root: `arr ** 0.5`.
`square`	Square: `arr ** 2`.
`exp`	Exponentiation: `np.e ** arr`.
`log, log10, log2, log1p`	Natural, base 10, and base 2 logarithms. `log1p(x)` returns `log(1+x)`.

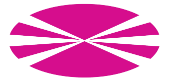
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Universal functions
Unary `ufunc`s

Function	Return value
`sign`	The sign of each element in the array: 1 (positive), 0 (zero), -1 (negative).
`ceil / floor / rint`	Round up / down / to closer integer.
`modf`	Integral and fractional parts of an array.
`isnan`	`True` if an element is `np.nan` (NaN).
`isfinite, isinf`	Boolean array indicating if an element is finite / infinite.

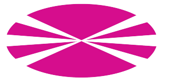
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Universal functions
Unary `ufunc`s

Function	Return value
`cos, cosh, sin, sinh, tan, tanh`	Trigonometric and hyperbolic functions.
`arccos, arccosh, arcsin, arcsinh, arctan, arctanh`	Inverse trigonometric and hyperbolic functions.
`logical_not`	Element-wise boolean NOT.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Universal functions
Binary `ufunc`s

Function	Return value
`add`	Addition of two input arrays.
`subtract`	Subtracts the second parameter from the first.
`multiply`	Multiplication of two input arrays.
`divide / floor_divide`	Division / integral division of the input.
`power`	Exponentiation of the bases in the first array to the exponents in the second array.

Ecole Internationale des Sciences du Traitement de l'Information

NumPy - Universal functions
Binary `ufunc`s

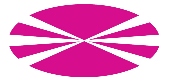
Function	Return value
`maximum / fmax`	Element-wise maximum. `fmax()` ignores NaN values.
`minimum / fmin`	Element-wise minimum. `fmin()` ignores NaN values.
`mod`	Element-wise modulo.
`copysign`	Copies the signs of the element in the second array to the values in the first.

Ecole Internationale des Sciences du Traitement de l'Information

NumPy - Universal functions
Binary `ufunc`s

Function	Return value
`greater / greater_equal / less / less_equal / equal / not_equal`	Boolean operations comparing wether each element in the first array is greater / greater or equal / less / less or equal / equal / different than each element in the second array.
`logical_and / logical_or / logical_xor`	Element-wise boolean operations AND / OR / XOR.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



NumPy - Data processing

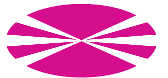
Notebook-002.ipynb

- Using NumPy we can express data processing algorithms that would otherwise require loops using array operations.

- This *vectorization* process usually comes together with significant performance improvements.

- We will explore different ways of using NumPy to process data.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Data processing
Conditional logic

Notebook-002.ipynb

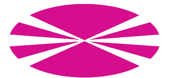
- `numpy.where()` allows to write a vectorized version of the expression `x if c else y`.
- Using pure Python, this can be written as:

```
>>> result = [(x if c else y) for x, y, c in zip(xarr, yarr, cond)]
```

- However, this version...
  - Is very slow.
  - works with 1-dimensional arrays only.
- The equivalent `where()` version is written:

```
>>> result = np.where( cond, xarr, yarr )
```

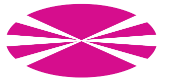
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Data processing
Mathematics and statistics

Method	Description
`sum`	Summation of all the elements of an array along a given optional axis.
`mean`	Arithmetic mean.
`std, var`	Standard deviation and variance.
`min, max`	Minimum and maximum.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

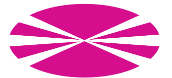


NumPy - Data processing
Mathematics and statistics

`Notebook-002.ipynb`

Method	Description
`argmin, argmax`	Indices of minimum and maximum elements.
`cumsum`	Cumulative sum.
`cumprod`	Cumulative product.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Data processing
Boolean arrays

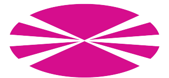
Notebook-002.ipynb

- When working with boolean arrays, the previous methods use `True == 1` and `False == 0`. This allows to reduce boolean values using `cumsum()` / `cumprod()` instead of AND / OR.

- There are however ad-hoc methods for reducing boolean arrays: `any()` and `all()`.

- These methods work with non-boolean arrays by interpreting `non-zero == True`, `zero == False`.

Ecole Internationale des Sciences du Traitement de l'Information

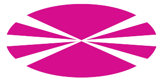


NumPy - Data processing
Sorting

`Notebook-002.ipynb`

- As with lists, NumPy arrays can be sorted in place using `sort()`.
- `sort()` accepts an optional `axis` parameter which indicates the axis to sort over in multidimensional arrays.
- `numpy.sort()` returns a sorted copy of the array.

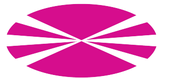
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Data processing
Set logic

Function	Description
`unique(x)`	Computes the unique elements in array `x` (1D).
`intersect1d(x,y)`	Common elements in `x` and `y`.
`union1d(x,y)`	Union of `x` and `y`.
`in1d(x,y)`	Boolean array indicating whether each element of `x` is in `y`.
`setdiff1d(x,y)`	Set difference, `x`-`y`.
`setxor1d(x,y)`	Symmetric set difference.

Ecole Internationale des Sciences du Traitement de l'Information

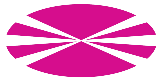


NumPy - Array I/O
Storing arrays to disk

- `np.save()` and `np.load()` store arrays to disk in binary format.
- By default, the data is stored uncompressed with extension `.npy`.

```
>>> np.save( path_string, arr )
.
.
.
>>> arr = np.load( path_string )
```

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Array I/O
Storing arrays to disk

- We can store several arrays to the same file compressed using ZIP and with `.npz` extension using `np.savez()`:

```
>>> np.savez( path_string, a = arr1, b = arr2 )
```

- Reading a `.npz` file NumPy returns a dictionary object that loads individual arrays in a lazy way:

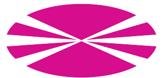
```
>>> arch = np.load( path_string )
>>> arch[‘b’]
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Ecole Internationale des Sciences du Traitement de l'Information

NumPy - Array I/O
Storing arrays to disk

- It is often useful to read arrays stored in text format.
- NumPy provides `np.loadtxt()` and `np.genfromtxt()`.
- Both can be adapted to a specific input format, varying comments format, delimiters, conversion functions, rows to be skipped, or columns to parse.
- `np.savetxt()` performs the reverse operation: it writes an array to a CSV-like file.
- `np.genfromtxt()` is similar to `np.loadtxt()`, but it supports treatment of missing values and structuring of the output array.

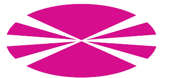
Ecole
Internationale
des Sciences
du Traitement
de l'Information



NumPy - Linear algebra (`numpy.linalg`)

Function	Description
`diag`	Returns the elements in the diagonal of a square matrix as a 1D array, or builds a square matrix from a 1D array.
`dot`	Matrix product.
`trace`	Sum of the elements in the matrix diagonal.
`det`	Determinant.
`eig`	Eigenvalues and eigenvectors.

Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Linear algebra (`numpy.linalg`)

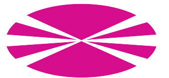
Function	Description
`inv`	Inverse of an square matrix.
`pinv`	Moore-Penrose pseudo-inverse.
`qr`	QR decomposition.
`svd`	Singular-value decomposition.
`solve`	Solves `Ax = b`, with `A` a square matrix.
`lstsq`	Least squares solution to `y = Xb`.

Ecole Internationale des Sciences du Traitement de l'Information

NumPy - Random numbers (`numpy.random`)

Function	Description
`seed`	Changes the generator seed.
`permutation`	Random permutation of a sequence.
`shuffle`	Random permutation of a sequence (in-place).
`rand`	Samples a uniform distribution.
`randint`	Samples integers from a uniform distribution.

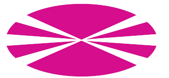
Ecole Internationale des Sciences du Traitement de l'Information



NumPy - Random numbers (`numpy.random`)

Function	Description
`randn`	Samples a standard normal distribution.
`binomial`	Samples a binomial distribution.
`normal`	Samples a normal distribution.
`beta`	Samples a beta distribution.

Ecole Internationale des Sciences du Traitement de l'Information

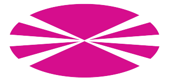


NumPy - Random numbers (`numpy.random`)

`Notebook-002.ipynb`

Function	Description
`chisquare`	Samples a χ^2 distribution.
`gamma`	Samples a gamma distribution.
`uniform`	Samples a uniform `[0,1)` distribution.

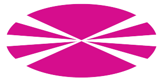
Ecole Internationale des Sciences du Traitement de l'Information



Matplotlib

- The previous examples included some plotting, but without going into details.
- Matplotlib provides mostly 2D plotting capabilities, although it includes limited 3D functionalities.
- The basic supported plot types are lines, bar charts, histograms, pie charts, and variations of them.
- This section briefly introduces Matplotlib. It has a vast amount of options that cannot be covered in detail. We will later focus on higher level libraries instead.

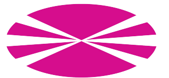
Ecole Internationale des Sciences du Traitement de l'Information



Matplotlib - Introduction

- The `%matplotlib` magic command automatically configures iPython to show plots.
- By default, iPython detects the proper backend for the current window manager. Using `%matplotlib inline` the plots will be shown embedded inside a QT or Notebook environment.
- A basic Matplotlib plot includes the following elements:
  - `x` and `y` axes: horizontal and vertical axes, respectively.
  - Tick marks in the `x` and `y` axes.
  - Tick labels, showing axis values.
  - Drawing area, called canvas.

Ecole Internationale des Sciences du Traitement de l'Information



Matplotlib - Introduction

Notebook-003.ipynb

- `plot()` is used to draw lines and marks.
- It accepts pairs of `x` and `y` sequences, which must have the same lengths, together with a string indicating how to plot the data.
- Adjacent points are joined using straight lines.
- Points and lines are drawn following the requested style.
- The function returns a list of the lines which have been added to the current figure.
- If only one sequence is passed to the function, it is assumed to contain the values of the `y` axis. Values for the horizontal axis will be automatically generated as `x = range( len(y) )`.

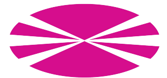
Ecole Internationale des Sciences du Traitement de l'Information

Matplotlib - Introduction

Notebook-003.ipynb

Function	Description
`figure`	Creates a new figure. Accepts an integer that acts as a unique identifier for this figure. This integer can be used to programatically change the active figure.
`subplot( x, y, z )`	Divides a figure into a mesh of subfigures with `x` rows and `y` columns. Besides, it activates subfigure number `z` (`1 < z < x*y`, row major order).

Ecole Internationale des Sciences du Traitement de l'Information

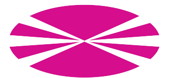


Matplotlib - Introduction

`Notebook-003.ipynb`

Function	Description
`bar / barh`	Create vertical / horizontal bar charts. To plot stacked bars the `bottom` parameters is used, indicating the starting point for the new bars.
`boxplot`	Box plots.
`scatter`	Draws points, but it does not join them with lines unlike `plot()`.
`hist`	Histograms.
`pie`	Pie charts.

Ecole Internationale des Sciences du Traitement de l'Information

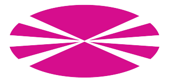


Matplotlib - Design

Notebook-003.ipynb

Function	Description
`title`	Plot title. Like all other text-related Matplotlib functions, it accepts LaTeX syntax.
`xlim / ylim`	Configures the limits of the `x` / `y` axes.
`autoscale`	Automatic axes limits.
`xticks / yticks`	Configures the ticks for each axis and, optionally, the tick labels.

Ecole Internationale des Sciences du Traitement de l'Information

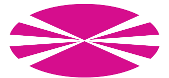


Matplotlib - Design

`Notebook-003.ipynb`

Function	Description
`axis`	Configures both axes’ limits.
`axhline / axvline`	Draws a horizontal / vertical line,
`axhspan / axvspan`	Draws rectangles that cover the entire width / height of the plot.

Ecole Internationale des Sciences du Traitement de l'Information

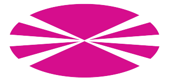


Matplotlib - Design
Axes

Notebook-003.ipynb

- A reference to an instance of `matplotlib.axes.Axes` allows to configure plots with a high level of detail.
- Such reference can be obtained by calling `matplotlib.pyplot.gca()` (`G`et `C`urrent `A`xes).
- E.g., calling `ax.xaxis.set_major_locator( matplotlib.ticker.MultipleLocator(10))` sets ticks in the `x` axis of the `ax` axes in values multiple of 10.
- E.g., calling `ax.xaxis.set_major_formatter()` allows to specify functions that will be used to format the tick labels in the `x` axis of the `ax` axes.

Ecole Internationale des Sciences du Traitement de l'Information



Matplotlib - Design
Legends and annotations

`Notebook-003.ipynb`

- All plotting functions (`plot()`, `hist()`, etc.) accept a `label` parameter indicating the name to use in the legend.
- `matplotlib.pyplot.legend()` automatically builds a legend from a list of handles. If no such list is provided, all the elements in the figure with a label not starting with “_” are listed.
- Otherwise, we can provide the lists of lines, etc. returned by each plotting function to specify which elements in the figure should be added to the legend.
- `matplotlib.pyplot.annotate()` adds a textual annotation and the specified place in the figure.

Ecole Internationale des Sciences du Traitement de l'Information

Matplotlib - Styles

Notebook-003.ipynb

Property	Value type	Description
`alpha`	`float`	Transparency used for the element.
`color`	Matplotlib color	Line / marker color.
`dashes`	Sequence	Line pattern.
`label`	`string`	Text to label this element in the plot legend.
`linestyle`	See docs	Line type.
`linewidth`	`float`	Line width, in points.

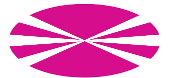
Ecole Internationale des Sciences du Traitement de l'Information

Matplotlib - Styles

`Notebook-003.ipynb`

Property	Value type	Description
`marker`	See docs	Marker used for points in a line.
`mec`	Matplotlib color	`M`arker `E`dge `C`olor.
`mew`	`float`	`M`arker `E`dge `W`idth.
`mfc`	Matplotlib color	`M`arker `F`ace `C`olor.
`markersize`	`float`	Marker size, in points.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Matplotlib - Styles

Notebook-003.ipynb

Property	Value type	Description
solid_capstyle	[‘butt’|‘round’|‘projecting’]	End of line style.
solid_joinstyle	[‘miter’|‘round’|‘bevel’]	Line union style.
visible	[True | False]	Visibility.
xdata	numpy.ndarray	X axis data.
ydata	numpy.ndarray	Y axis data.
Zorder	Número	Stacking order in the Z axis.

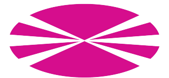
Ecole Internationale des Sciences du Traitement de l'Information

Matplotlib - Colors

Notebook-003.ipynb

- The `matplotlib.colors` module includes utilities for defining and converting colors.
- Basic predefined colors can be referenced using a single letter: b (*blue*), g (*green*), r (*red*), c (*cyan*), m (*magenta*), y (*yellow*), k (*black*), w (*white*).
- Shades of gray can be codified using a floating point number in `[0, 1]`.
- Other colors can be specified using different formats:
  - HTML hex string: `"#eeefff"`
  - `(R, G, B)` tuple, with `R`, `G` and `B` in `[0, 1]`.
  - HTML string: `"red"`, `"burlywood"`, `"chartreuse"`, ...

Ecole
Internationale
des Sciences
du Traitement
de l'Information



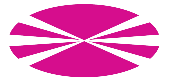
Matplotlib - Colors
Colormaps

`Notebook-003.ipynb`

- The `matplotlib.cm` module includes a set of colormaps to use with plots.

- Colormaps are useful when using `imshow()`, which is similar to `plot()` but interprets the values in the input array as indices to a colormap. If the input array is 2-dimensional, an image will be plotted.

Ecole Internationale des Sciences du Traitement de l'Information

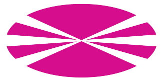


Matplotlib - Saving to file

Notebook-003.ipynb

- Figures can be saved to a file using `matplotlib.pyplot.savefig()`.
- The file format to use is inferred from the extension in the provided path.
- The most relevant parameters that control the quality of the stored figure are:
  - `dpi`: dots per inch.
  - `bbox_inches`: inches of whitespace surrounding the figure.
- The figure does not need to be stored to a file: it can be written to any object which supports I/O, such as `StringIO`.

Ecole Internationale des Sciences du Traitement de l'Information



Matplotlib - 2D histograms

Parameter to `hist()`	**Description**
`bins`	Number of classes to use in the histogram, or sequence containing the boundaries between classes.
`range`	Range for each class. If `bins` is not used, this can be provided as a sequence.
`normed`	If `True`, values are normalized and the result is a probability density function.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Matplotlib - 2D histograms

`Notebook-003.ipynb`

Parameter to `hist()`	**Description**
`histtype`	By default, bar chart. Other values: ● `barstacked`: stacks bars when working with multiple data series. ● `step`: line without filling. ● `stepfilled`: line with filling.
`align`	How to align bars for each class: `mid`, `left`, o `right`.
`color`	Colors to be used.
`orientation`	`horizontal` or `vertical`.

Ecole Internationale des Sciences du Traitement de l'Information

Matplotlib - Pie charts

Notebook-003.ipynb

Parameter to `pie()`	Description
`explode`	Fraction of the pie ratio to use as offset for each slice.
`autopct`	String or function indicating how to label each slice with a numeric value.
`pctdistance`	Ratio between the center of each slice and the start of the `autopct` text.
`labeldistance`	Radial distance to draw labels.
`startangle`	Rotation angle of the origin.
`wedgeprops`	Dictionary containing slice properties.

Ecole Internationale des Sciences du Traitement de l'Information

Matplotlib - 3D plots

`Notebook-003.ipynb`

- Although Matplotlib is focused in 2D plots, there are several toolkits provided limited 3D capabilities.

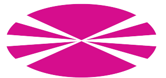
- The `mpl_toolkits.mplot3d` module provides methods to create 3D scatter plots, surfaces, lines and meshes. Its interface is very similar to the original Matplotlib.

- Main difference: axes are instances of `mpl_toolkits.mplot3d.Axes3D`. Projections are performed by specialized 3D classes, while other parts of the figure (labels, ticks, etc.) are managed by vanilla Matplotlib.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Text formats
JSON

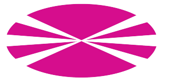
- JSON (*JavaScript Object Notation*) is an open standard and text format targetted at transmitting data in the form of key-value pairs.
- It is used as an alternative to HTML for transmitting data in client-server applications.
- Although originally derived from JavaScript, JSON is independent and many languages and interpreters include mechanisms to automate reading and writing JSON data.
- It is much more flexible than other text-based alternatives such as CSV.



Text formats
JSON

```
obj = “““
{ “name”: “John”,
  “residences”: [“USA”, “Spain”, “France”],
  “pet”: null,
  “relatives”: [ {“name”: “Andrew”, “age”: 25,
                  “pet”: “Zuko” },
                 {“name”: “Helen”, “age”: 33,
                  “pet”: “Cisco”} ]
}
”””
```

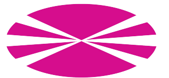
Ecole Internationale des Sciences du Traitement de l'Information



Text formats
JSON

- Almost valid Python code.
- Basic types: objects (dictionaries), arrays (lists), strings, numbers, booleans, and nulls.
- Keys must be strings.
- The `json` module, included in the Python standard library, allows to read and write Python objects in JSON using its `loads()` and `dumps()` methods, respectively.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Text formats
HTML y XML

`Notebook-004.ipynb`

- Many web services communicate through HTML/XML.
- The `urllib2` module allows to open HTTP connections.
- The `lxml` allows to parse XML documents, e.g. HTML.
- Parsing non-semantic web pages requires detailed knowledge of their structure.
- Many HTML elements exist only for formatting purposes. In order to extract the underlying semantics it is necessary to process formats, syntax, etc. This process is called data wrangling, or data munging.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Pandas

- Pandas provides data structures and methods to improve the structured data processing capabilities of native Python.
- The basic data structure in Pandas is the `DataFrame` (similar to `data.frame` in R). It is a 2D table, conceptually similar to an Excel spreadsheet.
- Pandas combines the array processing capabilities of NumPy with the flexibility of spreadsheets and relational databases.
- Provides indexing, reshaping, splitting, aggregation, and selection of subsets of data.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Pandas

- Pandas design objective is to provide new data management capabilities to the Python ecosystem:
  - Data structures with indexed axes supporting explicit or implicit data alignment.
  - Seamless processing of time series (timestamp-indexed data).
  - Arithmetic operations and reductions along axes.
  - Flexible management of unknown (null) data.
  - Merges and similar operations typical of relational databases.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

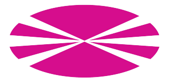
Pandas - Data structures

`Series`

`Notebook-005.ipynb`

- A `Series` object represents an object conceptually similar to a 1D array.
- It actually contains two different arrays: a data array and an index array, which labels data.
- Both arrays are esentially NumPy arrays with their own datatypes.
- Series are similar to dictionaries, as they can be seen as key-value pairs. In fact, a constructor is provided to build a `Series` object from a `dict`.

Ecole Internationale des Sciences du Traitement de l'Information



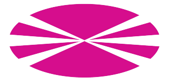
Pandas - Data structures

DataFrame

Notebook-005.ipynb

- A DataFrame object represents a tabular structure, similar to a spreadsheet (or a data.frame in R).
- It contains an ordered collection of colums, each of which may have a different datatype (numeric, string, etc.).
- Includes an index on its rows and another one on its columns. In this sense, it can be seen as an aggregation of series, all of them sharing the same index.
- Internally, the data is stored in a 2D format (bidimensional NumPy array), although higher dimensional data may be represented using hierarchical indices.

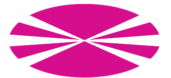
Ecole Internationale des Sciences du Traitement de l'Information



Pandas - Data structures
`DataFrame()` constructor

Parameter (type)	Description
`2D ndarray`	Data array, with optional row and column labels.
`dict of arrays, lists, or tuples`	Each sequence becomes a column of the `DataFrame`. All of them might have the same length.
`dict of Series`	Each value becomes a column. The indices in each series are unified if an explicit index is not provided.

Ecole Internationale des Sciences du Traitement de l'Information

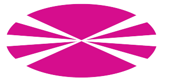


Pandas - Data structures

`DataFrame()` constructor

Parameter (type)	Description
`dict of dicts`	Each internal dictionary becomes a column. The keys in the different dictionaries are unified as in a `dict` of `Series`.
`list of dicts or Series`	Each item becomes a row in the `DataFrame`. The union of the keys of the dictionaries or the indices of the series is used for the column labels.
`list of lists or tuples`	Same as providing an `2D ndarray`.

Ecole Internationale des Sciences du Traitement de l'Information

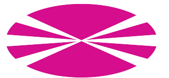


Pandas - Data structures

`DataFrame()` constructor

Parameter (type)	Description
`DataFrame`	The indices already in the old DataFrame are used, unless different ones are explicitly provided.

Ecole Internationale des Sciences du Traitement de l'Information

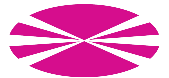


Pandas - Data structures

`Index`

`Notebook-005.ipynb`

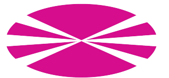
- `Index` objects are responsible for storing axis labels and names.
- Any other array or sequence type provided as an index when building a `Series` or `DataFrame` object is internally converted to an `Index`.
- `Index` objects are immutable. This guarantees referential integrity when shared by different structures.
- `Index` provides methods and attributes to support set logic and value inspection.



Pandas - Data structures
`Index` subclasses

Class	Description
`Index`	Array of generic Python objects.
`Int64Index`	Integers.
`MultiIndex`	Hierarchical index, representing multiple indexing levels in a single axis. Similar to a tuple array.
`DatetimeIndex`	Timestamp with nanosecond resolution.
`PeriodIndex`	Time periods.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas - Data structures
`Index` methods

Method	Description
`append`	Concatenates additional `Index` objects, producing a new object.
`diff`	Set difference.
`intersection`	Set intersection.
`union`	Set union.
`isin`	Computes a boolean array marking whether each of the entries in an `Index` is included in another collection.
`delete`	Creates a new `Index` object by removing an element from the original one.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Data structures
`Index` methods

Method	Description
`drop`	Creates a new `Index` object by removing a set of elements from the original one.
`insert`	Creates a new `Index` object by inserting a new element into the original one.
`is_monotonic`	`True` if each element is greater or equal than the previous one. Alias for `is_monotonic_increasing()` . An `is_monotonic_decreasing()` method is also provided.
`is_unique`	`True` if the index has no duplicate elements.
`unique`	Computes an array containing the unique elements in the index.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Essential functions

`Notebook-005.ipynb`

- Reindexing: `reindex()` method.
- Removing entries: `drop()` method.
- Indexing, selection, and filtering: operator `[]` and attributes `DataFrame.loc` and `DataFrame.iloc`.
- Arithmetic operations: operators `+`, `-`, `*`, and `/`, and methods `add()`, `sub()`, `mul()` y `div()`.
- Functional application and mapping: basic methods (`mean()`, `sum()`, …), method `Series.map()`, methods `DataFrame.apply()`, and `DataFrame.applymap()`.
- Sorting and classification: methods `sort_values()`, `sort_index()`, and `rank()`.
- Managing indices with duplicates: method `Index.is_unique()`.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Essential functions
`reindex()` parameters

Parameter	Description
`index`	New sequence to use as index.
`method`	Interpolation method (`ffill` or `bfill`).
`fill_value`	Value to use as a placeholder for null data.
`limit`	Maximum number of elements to fill using interpolation.
`level`	Hierarchical level to reindex.
`copy`	Marks whether data should be copied in case the new and the old indices are equivalent (`True` by default).

Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Essential functions
`DataFrame` indexing

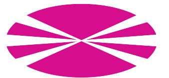
Syntax	Description
`obj[val]`	Selects a column or subset of columns, except if `val` is an array or the `DataFrame` is boolean, in which case it filters colums.
`obj.loc[val]`	Selects a row or subset of rows by label.
`obj.loc[:,val]`	Selects a column or subset of columns, by label.
`obj.loc[val1, val2]`	Selects both rows and columns, by label.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Essential functions
`DataFrame` indexing

Syntax	Description
`reindex()`	Reorganizes one or more axes according to new indices.
`xs()`	Returns a cross-section of the dataframe attending to its labels.
`icol() / irow()`	Selects a single row / column attending to its location.
`get_value() / set_value()`	Selects a single value attending to row/column labels.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas - Descriptive statistics
Reduction parameters

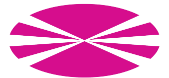
Parameter	Description
`axis`	The axis on which to perform the reduction (rows=`0`, colums=`1`).
`skipna`	Whether to exclude null values. `True` by default.
`level`	Reduction grouping by `level` in hierarchical indices.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Pandas - Descriptive statistics Methods

Method	Description
`count`	Number of non-null values.
`describe`	Computes several statistics of a `Series` or the columns of a `DataFrame`.
`min / max`	Minimum / maximum value.
`argmin / argmax`	Location of the minimum / maximum value.
`idxmin / idxmax`	Index (label) of the minimum / maximum value.
`quantile`	Returns the specified p-quantile.
`sum`	Summation.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas - Descriptive statistics Methods

Method	Description
`mean`	Arithmetic mean.
`median`	Median.
`mad`	Mean absolute deviation.
`var`	Variance.
`std`	Standard deviation.
`skew`	Skewness (third standardized moment).
`kurt`	Kurtosis (fourth standardized moment).

Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Descriptive statistics Methods

Method	Description
`cumsum`	Cumulative sum.
`cummin, cummax`	Cumulative minimum / maximum.
`cumprod`	Cumulative product.
`diff`	First order differences.
`pct_change`	Percentage change.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Unknown data

Notebook-005.ipynb

- Unknown or null data are common in most data analysis applications.

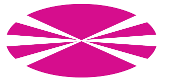
- Pandas simplifies the management of null data, e.g., all descriptive statistics functions automatically omit unknown values.

- `numpy.nan` is used as the default placeholder for unknown data. `None` will also be treated as a null value by Pandas.

Pandas - Unknown data
`fillna()` parameters

Parameter	Description
`value`	Escalar value or dictionary to use for filling.
`method`	Interpolation type (“`ffill`” or “`bfill`”).
`axis`	Axis to fill (by default `0`, i.e., rows).
`inplace`	Modifies the object inplace, instead of creating a new copy.
`limit`	Maximum number of consecutive values to fill using interpolation.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas - Hierarchical indices

`Notebook-005.ipynb`

- Hierarchical indexing allows to works (at a conceptual level) using `DataFrame` objects with more than 2 dimensions.

- It is implemented through the addition of different *levels* to the row and/or column indices of the table.

- Hierarchical indices are implemented by the `MultiIndex` class.

- It is possible to apply reduction operations to the different levels in the index hierarchy, obtaining a `DataFrame` (instead of a `Series`) after the reduction.

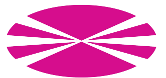
Ecole Internationale des Sciences du Traitement de l'Information

Pandas - Other considerations
Integer indexing

- Pandas objects with integer indices can be confusing, as the semantics of position- and label-based indexation vary.

```
obj = Series( range(5), index=['a', 'b', 'c', 'd', 'e'] )
obj[-1]
??


obj = Series( range(5), index=range(5) )
obj[-1]
??
```

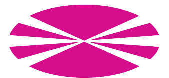
Ecole Internationale des Sciences du Traitement de l'Information



Pandas - Other considerations
Integer indexing

- For objects with an integer index, Pandas cannot decide whether the user wants to apply location- or label-based indexing.
- It solves the ambiguity by always using label-based indexing.
- Provides methods `Series.iget_value()` and `DataFrame.irow()` for positional-based indexing in integer-indexed data.

Ecole Internationale des Sciences du Traitement de l'Information



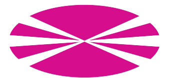
Pandas: Data I/O
Reading / writing text

Notebook-006.ipynb

- Pandas provides a set of methods to create `DataFrame` objects from tabular data stored in text format. The most useful ones are `read_csv()` and `read_table()` (which are nowadays mostly equivalent), which include options for:
  - Using one or more columns in the text file as `DataFrame` indices.
  - Naming columns using parameters or extracting names from the file.
  - Inferring types and performing data conversions.
  - Parsing dates, including combining several colums into a single one.
  - Iterating over chunks of large files (to fit data into memory).
  - Cleaning data: ignoring some rows or columns, comments, etc.

- Automatic type inference implies that it is not required to specify column types. Management of data and other non-basic types takes extra effort.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Data I/O

`read_csv` / `read_table` parameters

Parameter	Description
`path`	URL of the file to open.
`sep / delimiter`	Regular expression to use for separating fields.
`header`	Row number containing the colum names (`0` by default), or `None` if correlative integers should be used.
`index_col`	Columns to build the `DataFrame` index.
`names`	Names for the `DataFrame` columns. Combine with `header=None`.
`skiprows`	Number of rows to ignore at the beginning of the file, or list of row numbers to ignore.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data I/O

`read_csv` / `read_table` parameters

Parameter	Description
`na_values`	Values that should be considered marks of unknown data.
`comment`	Regular expression to mark the beginning of a comment.
`parse_dates`	Tries to parse dates to a `datetime` object. `False` by default. If `True`, Pandas tries to parse all columns as dates. Alternatively, it can be a list of specific columns to parse as dates. If an element in the list is a tuple/list, it will try to combine the specified columns to parse a single date.
`keep_date_col`	If several columns are used to build a single date column, do not keep the joined columns. `True` by default.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data I/O

`read_csv` / `read_table` parameters

Parameter	Description
`converters`	Dictionary containing a mapping of columns to parsing functions. Each corresponding function will be applied to all the elements of a given column and the result will be inserted into the `DataFrame`.
`dayfirst`	When parsing potentially ambiguous dates, assume dates in international format (DD/MM/YYYY). `False` by default.
`date_parser`	Function to use to parse dates.
`nrows`	Number of rows to read, starting from the beginning of the file.
`iterator`	If `True`, the return value will be a `TextFileReader` object, to process the file in a chunk-by-chunk manner.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data I/O

`read_csv` / `read_table` parameters

Parameter	Description
`chunksize`	If chunk-by-chunk processing is active, size of each chunk.
`skip_footer`	Number of rows to ignore at the end of the file.
`verbose`	If `True`, print information about the parsing process.
`encoding`	Character encoding, e.g., ‘utf-8’.
`squeeze`	If `True`, in case the result contains a single column, return a `Series`.
`thousands`	Thousands separator, e.g., ‘,’ or ‘.’

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data I/O
Binary formats

Notebook-006.ipynb

- The `pickle` module in the Python standard library provides a convenient method to serialize (marshall) and deserialize (unmarshall) objects to binary format.
- Pandas provides the methods `to_pickle()`/`read_pickle()` which store/read data to/from `pickle` files.
- Using `pickle` is discouraged for long term storage, since `pickle` does not guarantee backwards compatibility.
- Pandas also provides methods to read and write Excel, HDF5, Stata, and HTML files, among others.

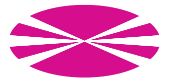
Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data I/O
HTML and web services

`Notebook-006.ipynb`

- Many web sites include APIs that provide access to data sources in JSON format.
- There are several ways to access these services from Python.
- A simple way is through the `requests` package.
- The responses to requests will be converted to a JSON object.
- It is trivial to build a Pandas object from JSON, as we have seen.

Ecole Internationale des Sciences du Traitement de l'Information

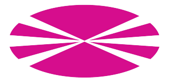


Pandas: Data I/O
Databases

`Notebook-006.ipynb`

- Text and binary files are ultimately inefficient to store large amounts of data.
- Databases, both relational and non-relational, are one of the most common data sources in computer science.
- Pandas provides methods to load data from SQL queries. The `pandas.io.sql` module allows to execute SQL sentences and process the results.
- Other types of non-SQL databases, such as MongoDB, store objects in different formats: JSON, text, etc. Loading data mechanisms vary on a per-case basis.

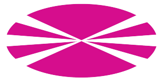
Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Data wrangling

- Much of the programming work in data analysis is spent on data preparation: loading, cleaning, transforming, and rearranging.
- Pandas and Python provide a set of flexible, high level data manipulation tools suitable for these preparation tasks:
  - Merging and combination of data: `merge()`, `concat()`, `combine_first()`.
  - Reshaping and pivoting: `stack()`, `unstack()`, `pivot()`.
  - Data transformation: removing duplicates, functional application, replacing values, renaming axes, discretization and binning, detecting and filtering outliers, permutation, and random sampling.
  - String manipulation: normalizing, cleaning, use of regular expressions...

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Data wrangling
Combining and merging datasets

`Notebook-007.ipynb`

- `pandas.merge()` connects rows in `DataFrame` based on one or more keys. It is similar to a `join` operation in a relational database.

- `pandas.concat()` glues or stacks together objects along an axis:
  - How are the new axes labeled? (union, intersection, …)
  - Are the original groups identifiable in the resulting object?
  - Which axis should be used for concatenating?

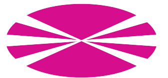
- `combine_first()` enables splicing together overlapping data to fill in missing values in one object with values from the other.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
`merge()` parameters

Parameter	Description
`left`	Left hand side operand of the merge.
`right`	Right hand side operand of the merge.
`how`	`[‘inner’|’outer’|’left’|’right’].‘inner’` by default.
`on`	Column names to join on. Must be found in both `DataFrame` objects. If not specified, will use all columns with matching names.

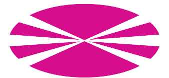
Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Data wrangling
`merge()` parameters

Parameter	Description
`left_on`	Columns in *left* operand to use as join keys.
`right_on`	Columns in *right* operand to use as join keys.
`left_index`	`[True|False]`. Use row index in *left* as its join key.
`right_index`	`[True|False]`. Use row index in *right* as its join key.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Data wrangling
`merge()` parameters

Parameter	Description
`sort`	`[True|False]`. Sort merged data lexicographically by join keys. `True` by default. Disable to get better performance in some cases on large datasets.
`suffixes`	Tuple of string values to append to column names in case of overlap. `(‘_x’, ‘_y’)` by default.
`copy`	If `False`, avoid copying data into resulting data structure in some exceptional cases. By default always copies.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Pandas: Data wrangling
`concat()` parameters

Parameter	Description
`objs`	List or dictionary of Pandas objects to be concatenated. The only required argument.
`axis`	Axis to concatenate along. `0` by default (rows).
`join`	`[‘inner’|’outer’]`. `‘outer’` by default. Whether to intersect (`inner`) or union (`outer`) together indices along the other axes.
`join_axes`	Specific indices to use for the other `(n-1)` axes instead of performing union/intersection logic.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
`concat()` parameters

Parameter	Description
`keys`	Values to associate with objects being concatenated, forming a hierarchical index along the concatenation axis.
`levels`	Specific indices to use as hierarchical index level or levels if keys passed.
`names`	Names for created hierarchical levels if `keys` and/or `levels` passed.
`verify_integrity`	Check new axis in concatenated objects for duplicates and raise exception if so. `False` by default.
`ignore_index`	Do not preserve indices along concatenation axis.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
Reshaping and pivoting

`Notebook-007.ipynb`

- A different kind of transformations is its reshaping, sometimes called pivoting.

- It consists in transposing rows and columns, modifying data dimensionality.

- The basic reshaping and pivoting operations provided by Pandas are:
  - `stack()`: pivots from the columns in the data to the rows.
  - `unstack()`: pivots from the rows into the columns.
  - `pivot()`: reshapes rows and columns, allowing to transform tables in “long” format to “wide” format in a single step.

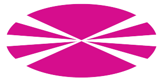
Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
Data transformations

`Notebook-007.ipynb`

- We have focused on structural modifications. There are other types of transformations which focus on the data:
  - Removing duplicates: `duplicated()`, `drop_duplicates()`.
  - Functional application: `apply()`, `map()`, `applymap()`.
  - Value substitution: `replace()`.
  - Index renaming: `rename()`.
  - Discretization and binning: `cut()`, `qcut()`.
  - Detecting and filtering outliers.
  - Permutation and random sampling: `permutation()`, `take()`.
  - Computing indicators/dummy variables: `get_dummies()`.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Data wrangling
String manipulation

`Notebook-007.ipynb`

- One of Python's most popular characteristics is its string manipulation routines.

- The `str` class provides methods for conveniently performing many string operations, such as searching, substitions, splitting, etc.

- Many of these operations accept regular expressions as parameters.

- Pandas adds string functionality, as it allows to apply string operations over data tables automatically handling unknown values.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
`str` methods

Method	Description
`count`	Returns the number of non-overlapping occurrences of substring in the string.
`endswidth / startswidth`	Returns `True` if a string ends with suffix / starts with prefix.
`join`	Use string as a delimiter for concatenating a sequence of other strings.
`index`	Return position of first character in substring if found in the string. Raises `ValueError` if not found.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
`str` methods

Method	Description
`find`	Like `index()`, but returns `-1` if not found.
`rfind`	Like `find()`, but returns position of last occurrence.
`replace`	Replace occurrences of string with another string.
`strip / rstrip / lstrip`	Trim whitespace, including newlines.
`split`	Break string into list of substrings using passed delimiter.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Pandas: Data wrangling
`str` methods

Method	Description
`lower / upper`	Convert alphabet characters to lowercase or uppercase, respectively.
`ljust / rjust`	Left / right justify. Pad opposite side of string with spaces (or some other fill character) to return a string with a minimum width.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Pandas: Data wrangling
Regular expressions

`Notebook-007.ipynb`

- Regular expressions provide a flexible way to search or match string patterns in text.

- A regular expression is a string containing certain elements, called “special characters”, with a specific semantic.

- A regular expression (*regex*) describes a pattern to match in the text, or a way to manipulate it.

- The `re` module includes functions for regular expression application. It has three types of functions: pattern matching, substitutions, and splitting.

Ecole Internationale des Sciences du Traitement de l'Information

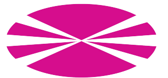
Pandas: Data wrangling
`regex` methods

Method	Description
`findall / finditer`	Return all non-overlapping matching patterns in a string as a list / iterator.
`match`	Match pattern at start of string and optionally segment pattern components into groups. Returns a `match` object, or `None`.
`search`	Scan string for match to pattern; returning a match object if so. The match can be anywhere in the string, as opposed to `match()`.
`split`	Break string into pieces at each occurrence of pattern.
`sub, subn`	Replace all / first n occurrences of pattern with replacement expression.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
Vectorized string functions in Pandas

Method	Description
`cat`	Concatenate strings element-wise with optional delimiter.
`contains`	Return boolean array if each string contains pattern.
`count`	Count occurrences of pattern.
`endswith` / `startswith`	Applies homonym functions in `str` element-wise.
`findall`	Compute list of all occurrences of pattern for each string.



Pandas: Data wrangling
Vectorized string functions in Pandas

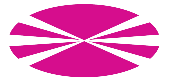
Method	Description
`get`	Index into each element (retrieve i-th element).
`join`	Join strings in each element of the Series with passed separator.
`len`	Compute length of each string.
`lower / upper`	Convert cases.
`match`	Use `re.match` with passed regex on each element.
`pad`	Add whitespace to left, right, or both sides of string.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data wrangling
Vectorized string functions in Pandas

Method	Description
`center`	Equivalent to `pad(side="both" )`.
`repeat`	Duplicate values.
`replace`	Replace occurrences of pattern with some other regex.
`slice`	Slice each string in the Series.
`split`	Split strings on delimiter.
`strip / rstrip / lstrip`	Trim whitespace, including newlines, element-wise.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Plotting and visualization

`Notebook-008.ipynb`

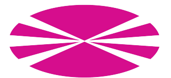
- Matplotlib provides a powerful framework for plotting and visualization, but it is a low-level tool.

- Building a plot involves configuring several Python objects without implicit semantics.

- Pandas objects are a centralized storage of data, with semantics at least partially known.

- Pandas objects provide a `plot()` method which builds complex plots in a convenient manner.

Ecole Internationale des Sciences du Traitement de l'Information



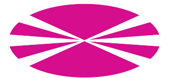
Pandas: Plotting and visualization
`plot()` parameters

Parameter	Description
`ax`	Matplotlib subplot object to plot on. If nothing passed, uses active subplot.
`kind`	`[‘line’|‘bar’|‘barh’|‘hist’|‘box’|‘kde’|‘density’|‘area’|‘pie’|‘scatter’|‘hexbin’].` Type of chart.
`logx / logy`	Use logarithmic scale on the X / Y axis.
`use_index`	Use the object index for tick labels.
`rot`	Rotation angle of tick labels.
`xticks / yticks`	Values to use for X / Y axis ticks.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Plotting and visualization
`plot()` parameters

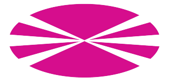
Parameter	Description
`xlim / ylim`	X / Y axis limits (specified as `[min,max]`).
`grid`	`[True|False]`. Display axis grid.
`subplots`	`[True|False]`. Plot each `DataFrame` column in a separate subplot.
`sharex / sharey`	`[True|False]`. If `subplots=True`, share the same X / Y axis, linking ticks and limits.
`layout`	Tuple indicating the geometry of subplots to use.
`figsize`	Figure size.



Pandas: Plotting and visualization
`plot()` parameters

Parameter	Description
`title`	Plot title.
`legend`	`[True|False|'reverse']`. Add a subplot legend.
`style`	Dictionary matching each column with the style to use for plotting it.
`loglog`	Use log scale for both axes.
`fontsize`	Font size to use for ticks.
`colormap`	Color map to index.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Plotting and visualization
`plot()` parameters

Parameter	Description
`colorbar`	`[True|False]`. Whether to draw the value legend for ‘`scatter`’ and ‘`hexbin`’ plot types).
`table`	If a `Series` or `DataFrame` object is provided, includes it in the plot. Useful to combine plots and tables.
`stacked`	Create stacked plot.
`sort_columns`	`[True|False]`. Whether to sort columns lexicographically before plotting.
`secondary_y`	List of columns that should be referenced to a secondary Y axis.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Plotting and visualization
`plot()` parameters

Parameter	Description
`mark_right`	`[True|False]`. When a secondary axis is used, automatically add the suffix "`(right)`" to the legends of the series referenced to it.
`**kwds`	Parameters not processed by Pandas will be passed to Matplotlib.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Pandas: Data aggregation

- One of the reasons for the popularity of relational databases and SQL is the ease with which data can be joined, filtered, transformed, and aggregated.
- However, query languages like SQL have limited expressiveness. Pandas allows to implement *split-apply-combine* operations conveniently:
  - Split a Pandas object into pieces using one or more keys.
  - Compute group summary statistics.
  - Apply a varying set of functions to each column of a `DataFrame`.
  - Compute pivot tables and cross-tabulations.
  - Perform quantile analysis and other data-derived group analyses.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data aggregation

key	data
A	0
B	5
C	10
A	5
B	10
C	15
A	10
B	15
C	20

split

A	0
A	5
A	10

B	5
B	10
B	15

C	10
C	15
C	20

apply

sum

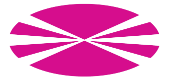
sum

sum

combine

A	15
B	30
C	45

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Data aggregation

`Notebook-009.ipynb`

- A critical aspect of these transformations is how to categorize data.
- A Pandas object is *split* into groups based on one or more keys, applied on a particular axis.
- The Pandas mechanism for this operation is `groupby()`.
- A grouping key can take many forms, e.g.:
    - A list or array of values that is the same length as the axis being grouped.
    - A value indicating a column name in a `DataFrame`.
    - A `dict` or `Series` giving correspondences between the values on the axis being grouped and the group names.
    - A function to be invoked on the axis index or the individual labels of the index.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data aggregation
Aggregation functions

`Notebook-009.ipynb`

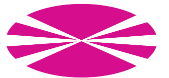
- An *aggregation functions* is any transformation which produces a scalar value from an array (also called *reduction*).
- Aggregation functions as implemented by the `GroupBy` class have been optimized and are computed on the original data of the `DataFrame` or `Series`.
- Applicable aggregation functions are not limited to this subset: any function, including user-defined functions, can be applied to a grouped dataset.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Data aggregation
`GroupBy` methods

Method	Description
`count`	Number of non-NA values in the group.
`sum`	Sum of non-NA values.
`mean`	Mean of non-NA values.
`median`	Arithmetic median of non-NA values.

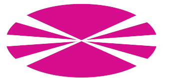
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Pandas: Agregación
Métodos en `GroupBy`

Método	Descripción
`std / var`	Unbiased standard deviation / variance.
`min / max`	Minimum / maximum of non-NA values.
`prod`	Product of non-NA values.
`first / last`	First / last non-NA value.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

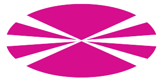


Pandas: Group-wise operations

Notebook-009.ipynb

- Aggregation is only one kind of group operation: accepts functions that reduce a one-dimensional array to a scalar value.
- In the general case, we want to apply any kind of operation to grouped data.
- This is done using `transform()` and `apply()`:
    - `transform()` broadcasts the result of an aggregation over the original members of the group.
    - `apply()` applies a function to each group and combines the results using `pandas.concat()`.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

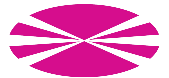


Pandas: Pivot tables and cross-tabulation

`Notebook-009.ipynb`

- A pivot table is a data summarization tool which aggregates a table by one or more keys, arranging the data in a rectangle with some groups along rows and some along columns.
- It can be built using `groupby()`, but `pivot_table()` provides a more convenient high-level interface.
- A cross-tabulation is a special case of a pivot table that computes group frequencies.
- Could also be built manually using several functions, but `crosstab()` simplifies the process.

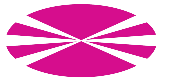
Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Time series

`Notebook-010.ipynb`

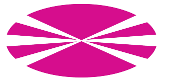
- Any dataset which includes observations at many points in time forms a time series.
- Many time series are *fixed frequency*: data points occur at regular intervals.
- Others are *irregular*: without a fixed offset between data points.
- How time series are referred depends on the application. Among others:
    - *Timestamps*: specific instants in time.
    - Fixed *periods*, such as the month of January 2007 or the full year 2010.
    - *Intervals* of time, indicated by start an end timestamps.
    - Elapsed time relative to a particular fixed start time.
- Pandas provides a standard set of time series tools and data algorithms to slice and dice, aggregate, resample, etc.



Pandas: Time series
Base frequencies

Alias	*Offset* type	Description
`D`	`Day`	Calendar daily.
`B`	`BusinessDay`	Business daily.
`H`	`Hour`	Hourly.
`T / min`	`Minute`	Minutely.
`S`	`Second`	Secondly.
`L / ms`	`Milli`	Millisecond.
`U`	`Micro`	Microsecond.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Time series
Base frequencies

Alias	*Offset* type	Description
`M`	`MonthEnd`	Last calendar day of month.
`BM`	`BusinessMonthEnd`	Last business day of month.
`MS`	`MonthBegin`	First calendar day of month.
`BMS`	`BusinessMonthBegin`	First business day of month.
`W-MON`, `W-TUE`, `...`	`Week`	Weekly on given day of week: `MON`, `TUE`, `WED`, `THU`, `FRI`, `SAT`, or `SUN`.
`WOM-1MON`, `WOM-1TUE`, `...`	`WeekOfMonth`	Generate weekly dates on the first, second, third, or fourth week of the month. For example, `WOM-3FRI` for the 3rd Friday of each month.

Pandas: Time series
Base frequencies

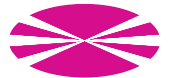
Alias	*Offset* type	Description
`Q-JAN, ...`	`QuarterEnd`	Quarterly dates anchored on last calendar day of each month, for year ending in indicated month.
`BQ-JAN, ...`	`BusinessQuarterEnd`	Quarterly dates anchored on last business day of each month, for year ending in indicated month.
`QS-JAN, ...`	`QuarterBegin`	Quarterly dates anchored on first calendar day of each month, for year ending in indicated month.
`BQS-JAN, ...`	`BusinessQuarterBegin`	Quarterly dates anchored on first business day of each month, for year ending in indicated month.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Time series
Base frequencies

Alias	***Offset* type**	**Description**
`A-JAN, ...`	`YearEnd`	Annual dates anchored on last calendar day of given month.
`BA-JAN, ...`	`BusinessYearEnd`	Annual dates anchored on last business day of given month.
`AS-JAN, ...`	`YearBegin`	Annual dates anchored on first calendar day of given month.
`BAS-JAN, ...`	`BusinessYearBegin`	Annual dates anchored on first business day of given month.

Ecole Internationale des Sciences du Traitement de l'Information



Pandas: Time series
`resample()` parameters

Parameters	Description
`freq`	String or `DateOffset` indicated desired resample frequency.
`how`	Function name or array function producing aggregated value.
`axis`	Axis to resample on, default to `0` (rows).
`fill_method`	How to interpolate when upsampling (`‘ffill’` or `‘bfill’`).
`closed`	In downsampling, which end of each interval is closed (inclusive). Defaults to `‘right’`.
`label`	In downsampling, how to label the aggregated result, with the right or left bin edge. Defaults to `‘right’`.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Time series
`resample()` parameters

Parameters	Description
`loffset`	Time adjustment to the bin labels, such as `‘-1s’` / `Second(-1)` to shift the aggregate labels one second earlier.
`limit`	When forward or backward filling, the maximum number of periods to fill.
`kind`	Aggregate to periods (`‘period’`) or timestamps (`‘timestamp’`); defaults to kind of index the time series has.
`convention`	When resampling periods, the convention (`‘start’` or `‘end’`) for converting the low frequency period to high frequency. Defaults to `‘end’`.

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Time series
Moving window functions

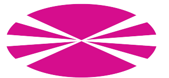
Función	Descripción
`rolling_count`	Returns number of non-NA observations in each trailing window.
`rolling_sum`	Moving window sum.
`rolling_mean`	Moving window mean.
`rolling_median`	Moving window meadian.
`rolling_std` / `rolling_var`	Moving window variance / standard deviation. Uses (n-1) denominator.
`rolling_skew` / `rolling_kurt`	Moving window skewness (3rd moment) / kurtosis (4th moment).

Ecole Internationale des Sciences du Traitement de l'Information

Pandas: Time series
Moving window functions

Function	Description
`rolling_min / rolling_max`	Moving window minimum / maximum.
`rolling_quantile`	Moving window score at percentile / sample quantile.
`rolling_corr / rolling_cov`	Moving window correlation / covariance.
`rolling_apply`	Apply generic array function over a moving window.

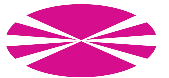
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Pandas: Time series
Moving window functions

Function	Description
`ewma`	Exponentially-weighted moving average.
`ewmstd / ewmvar`	Exponentially-weighted moving standard deviation / variance.
`ewmcorr / ewmcov`	Exponentially-weighted moving correlation / covariance.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

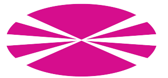


Scientific data libraries

- A computer uses multiple I/O technologies:
  - Magnetic tapes for backups.
  - Magnetic disks for HDDs.
  - SSDs for fast access to system functions.
  - Flash (USB pen drives) for portability.
  - Optical disks (DVD) for software distribution.

- POSIX defines functions to access data independently from the **physical** storage medium (`read()`, `write()`, `fread()`, etc.)

Ecole Internationale des Sciences du Traitement de l'Information

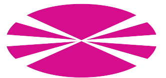


Scientific data libraries

- However, under POSIX the programmer is responsible for:
  - Using the OS-provided primitives (not all OS support POSIX!)
  - Organizing data according to a sequence-of-bytes model.
  - Performing the required conversions among different storage formats (e.g., when reading files generated by a *big-endian* architecture on a *little-endian* one).

- Basic aim: decouple applications from the storage **logic**.

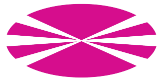
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Scientific data libraries
NetCDF

- In 1982, NASA designs CDF.
- Goal: to show a the pilot of a space shuttle climate data coming from different sources in a unified way in a single data screen through the *Pilot Climate Data System* (PCDS).
- Based on two fundamental ideas:
    - Including into the data file a **metadata** section, describing what the file contains.
    - An API between the application and the OS which decouples the data semantics and their storage logic.

Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
NetCDF

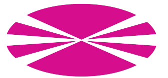
- NetCDF appears in 1990 as a spin-off of the CDF project.
- It maintains the fundamental CDF goals, plus:
    - Portability and platform-independent format.
    - C interface (CDF supported Fortran only).
    - UNIX implementation.
    - Mechanism to access multiple data points in a single step (hyperslab, same as using *slice* notation in Python).
- Common API used by every client: decouples data producers and consumers.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

Scientific data libraries
NetCDF

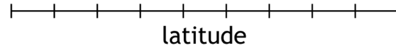
- A NetCDF file models a dataset using the following concepts:
  - Multidimensional variables:
    - Data types.
    - Shape (dimensionality).
    - Properties (key/value pairs).
  - Dimensions (coordinates):
    - Size.
    - Properties (key/value pairs).
  - File properties (key/value pairs).

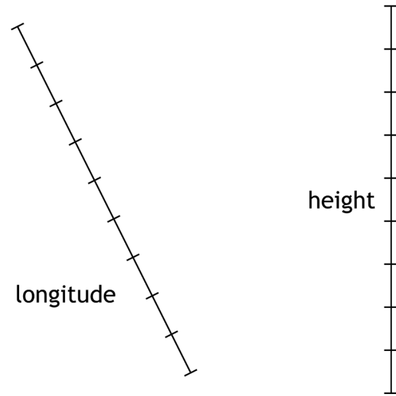
Ecole
Internationale
des Sciences
du Traitement
de l'Information



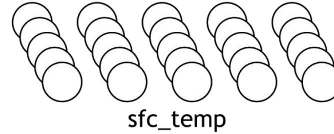
Scientific data libraries
NetCDF

Dimensions



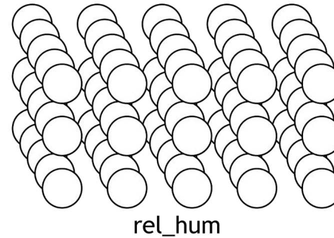


Variables



sfc_temp properties

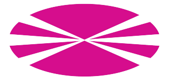
long_name: "surface temperature"
units: "degC"
valid_range: -100, 70



rel_hum properties

long_name: "relative humidity"
units: "percent"
missing_value: -99

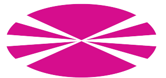
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
NetCDF

- Along with the data, the NetCDF file includes:
  - The names of dimensions, variables, and properties.
  - The coordinate systems associated to variables.
  - The magnitudes of coordinates and variables.
  - The properties which capture any other descriptive information about the data.
- High-level interface: data are accessed using their names and the slice to access (instead of strings of bytes).
- Supports the basic scientific data types: strings and numbers, both integer and floating point.

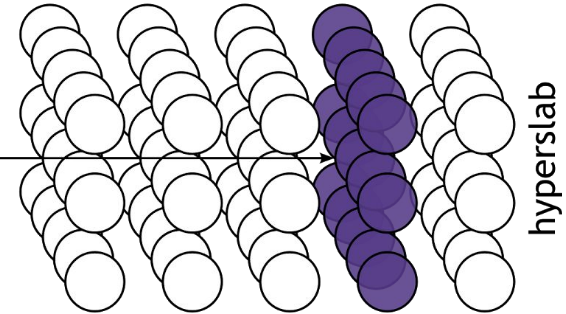
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
NetCDF

file id. variable id. corner vector edge length vector

```
nc_get_vara_<type>( netcdf_id, var_id, start, count )
nc_set_vara_<type>( netcdf_id, var_id, start, count )
```




```
<type> : text, uchar, schar, short, int, long,
         float, double, ushort, uint, longlong,
         ulonglong, string, (empty)
```

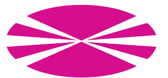
Ecole Internationale des Sciences du Traitement de l'Information

Scientific data libraries
NetCDF

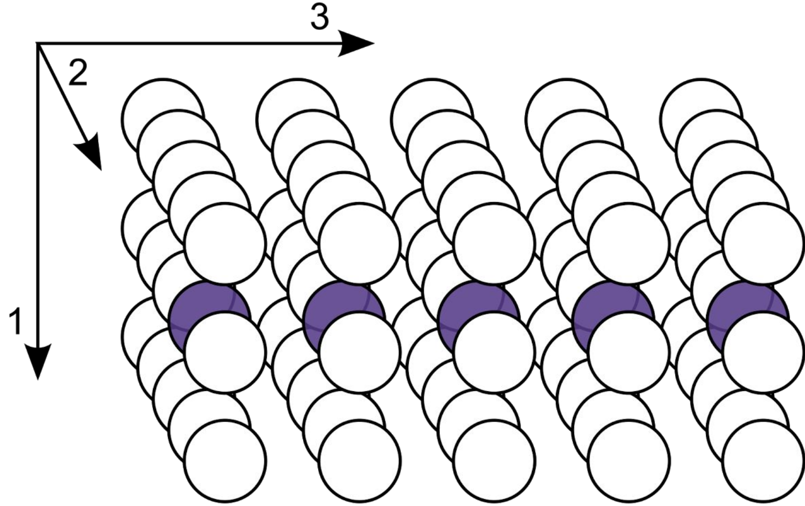
- We will use the Python interface.
- Instead of C-style functions (`nc_get_var(), nc_put_var()`) NumPy-style syntax is used, together with functions for creating and reading files.
- The NumPy-style slice syntax is readily translatable to hyperslab syntax:

$$[\ \ldots,\ e_k + a_k,\ \ldots\ ] = [\ \ldots,\ e_k{:}e_k{+}a_k,\ \ldots\ ]$$

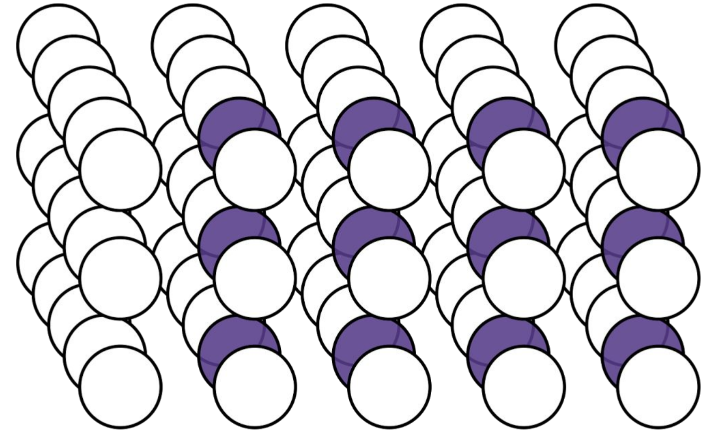
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
NetCDF

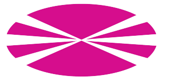



```
corner      : (1, 3, 0)
edge length: (1, 1, 5)
slices      : (1, 3, :5)
```

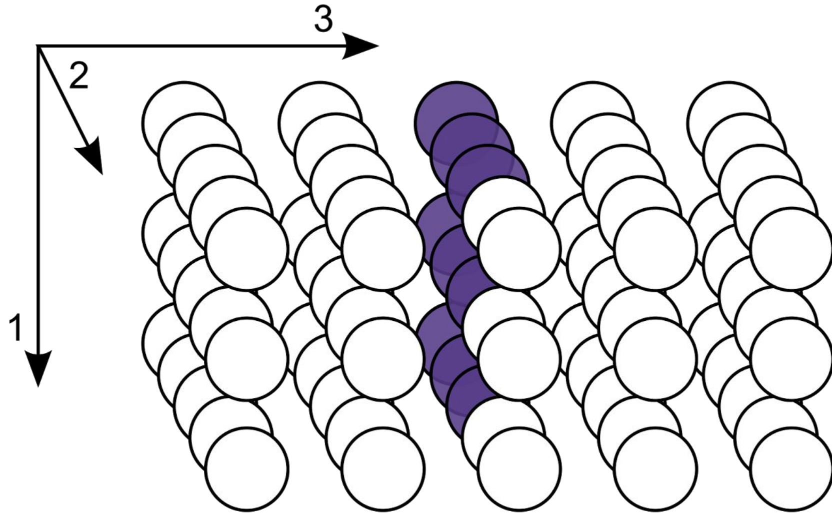


```
corner      : (0, 3, 1)
edge length: (3, 1, 4)
slices      : (:3, 3, 1:5)
```

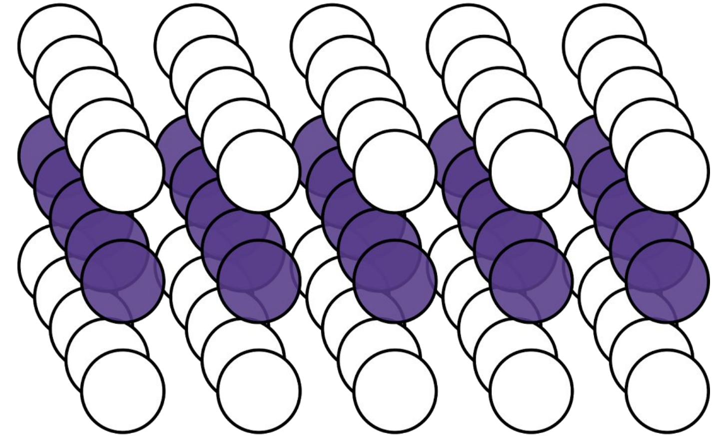
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Scientific data libraries
NetCDF

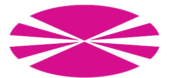



```
corner     : (0, 0, 2)
edge length: (3, 3, 1)
slices     : (:3, :3, 2)
```

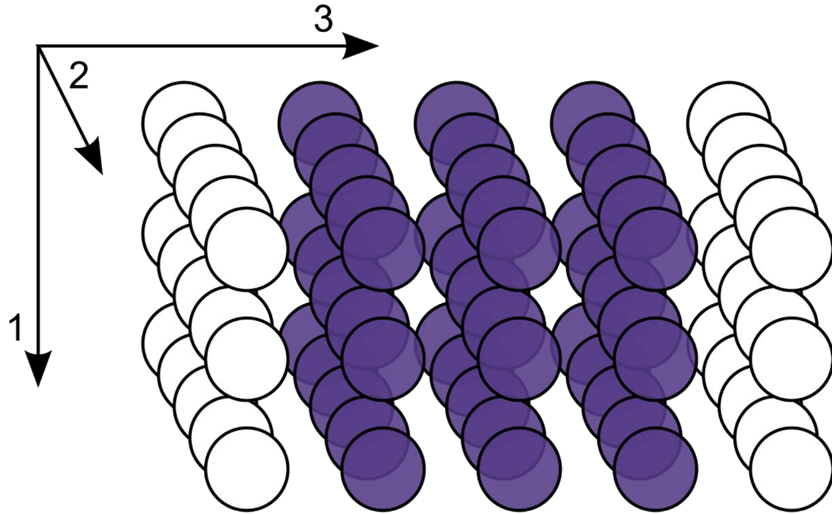


```
corner     : (1, 0, 0)
edge length: (1, 5, 5)
slices     : (1,:,:)
```

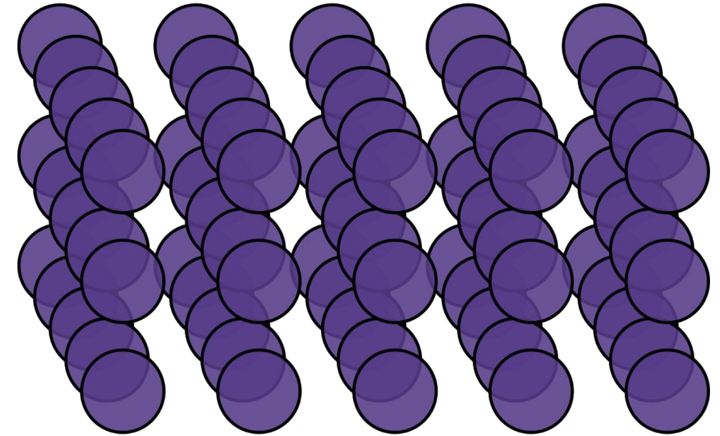
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
NetCDF

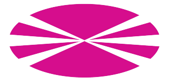



```
corner     : (0, 0, 1)
edge length: (3, 5, 3)
slices     : (:, :, 1:4)
```



```
corner     : (0, 0, 0)
edge length: (3, 5, 5)
slices     : (:, :, :)
```

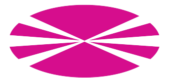
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
NetCDF

- Hyperslab accesses in NetCDF always return contiguous memory, even if the original data were not contiguous on disk.
- It is not possible to use a “step” different from 1 in hyperslab access (less versatile than slices).
- NetCDF APIs are consistent with the memory conventions of the host language: e.g., the C interface is row-major, while the Fortran interface is column-major.
- A Fortran array defined as `[N][M]` will be read in C/Python as `[M][N]`.
- A single dimension can be unlimited (i.e., allowed to grow indefinitely after being created). It must be the slowest-changing dimension (outermost).

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Scientific data libraries
NetCDF

- Internally, NetCDF (except version 4) uses the XDR format (*eXternal Data Representation*):
  - Non-proprietary format by Sun Microsystems to represent data in a platform-independent way.
  - Although XDR supports representing arbitrary C structs, NetCDF only supports multidimensional arrays of basic data types.
  - All conversions between machine-dependent formats are internally handled by XDR, in a programmer-transparent way. The application always receives data in the format native to the execution platform.

Ecole Internationale des Sciences du Traitement de l'Information

Scientific data libraries
NetCDF

`Notebook-011.ipynb`

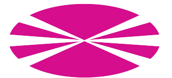
- NetCDF-3 limitations:
  - Single unlimited dimension, due to internal representation limitations.
  - Not possible to perform sequential accesses to data: impossible to read/write from/to standard input/output. Intermediate files must be used.
  - Not a DB system. Impossible to access data except through names and coordinates (e.g., not possible to search all points of data with value greater than a given threshold).
  - Not possible to represent anything but multidimensional arrays (e.g., lists of structs).

Ecole Internationale des Sciences du Traitement de l'Information

Scientific data libraries
COARDS conventions

Property	Description
`long_name`	Descriptive name of a variable.
`scale_factor`	If present, the program reading data must multiply it by a scaling factor upon read.
`add_offset`	If present, data must be shifted by a specific amount upon read.

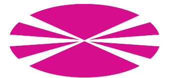
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
COARDS conventions

Property	Description
`valid_range`	Valid range of values.
`_FillValue`	If present, the value will be used to initialize data on disk. It will be considered as a mark of NA values, so it should be outside of the valid range of values.
`missing_value`	Same as `_FillValue`, but with no special treatment (i.e., data are not automatically initialized on disk).

Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
HDF5

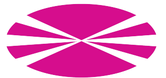
- HDF: Hierarchical Data Format.
- Created in 1987 in parallel to NetCDF.
- Developed by the University of Illinois at Urbana-Champaign and the National Center for Supercomputing Applications.
- Different objectives from NetCDF:
  - Platform-independent representation.
  - Access and store large objects efficiently.
  - Store different types of objects into a single container.
  - **Define a flexible format capable of accomodating new types of objects and metadata.**
  - Provide C and Fortran interfaces.

Ecole Internationale des Sciences du Traitement de l'Information

Scientific data libraries
HDF5

- During the 90s, and through NSF funding, HDF improves its end-user support: documentation, support, etc. and introduces compatibility with NetCDF files.
- In 1996 HDF5 is created on top of HDF4, providing parallel I/O for use in the Sandia, Los Alamos, and LLNL supercomputers.
- An HDF5 file is conceptually similar to a NetCDF file: it holds data and metadata, but working only with two types of objects:
  - Group: structure which aggregates cero or more HDF5 objects, plus metadata.
  - Dataset: Multidimensional data array, plus metadata.

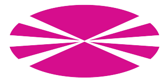
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
HDF5

- The hierarchy formed by groups and datasets is similar to a directory tree.
- Objects are referenced through paths:
  - "`/`" is the root group in the file.
  - "`/foo`" is object *foo* inside the root group in the file.
  - "`/foo/bar`" is object *bar* inside group *foo* inside the root group in the file.
- An HDF5 dataset aggregates the concepts of variable and dimension in NetCDF: dimensions do not exist independently from a dataset.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Scientific data libraries
HDF5

- A dataset is created from:
  - Its path: a group to contain the object.
  - A name.
  - A datatype.
  - A *dataspace*, which encodes dimensionality.

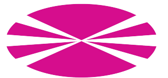
- The `h5py` module allows to store in HDF5 any Python object, except for Unicode strings and generic objects.

Ecole Internationale des Sciences du Traitement de l'Information

Scientific data libraries
HDF5

- A dataspace defines the dimensionality of a dataset:
  - Two types: simple and complex:
    - Simple dataspace: multidimensional array.
    - Complex dataspace: different, more general organization.
  - Each dimension can have fixed or unlimited size. HDF5 allows multiple unlimited dimensions.
- When an HDF5 is created, its properties can be specified. Properties may refer to:
  - The HDF5 file.
  - Each dataset.
  - Data access.

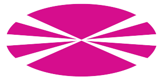
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
HDF5

- File creation properties:
  - User block size: a block at the beginning of the file which has free format, and can be written by the user at will (by default `0` bytes).
  - Offset and length pointer sizes (used in indices, by default `sizeof(size_t)` = `8` bytes).
  - Size of pointers in symbol table (by default `16` bits).
  - Size of pointers in B-trees to index chunked datasets (by default `32` bits).

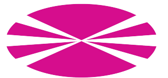
Ecole Internationale des Sciences du Traitement de l'Information



Scientific data libraries
HDF5

- Dataset creation properties:
  - Storage type: contiguous, compact, chunked.
  - Data filters: compression, checksums, user-defined.
  - Memory allocation: early, incremental, late.
  - Fill values: ifset, alloc, never.
- Data access properties:
  - Caches and buffer sizes.
  - Metadata block sizes.
  - Parallel I/O mode.
  - etc.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Scientific data libraries
HDF5

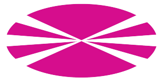
Notebook-011.ipynb

- HDF5 associates an *attribute* list to groups and datasets.

- The attribute is implemented as a small dataset with its own dataspace.

- Differences with a regular dataset:
    - Does not accept partial I/O (hyperslabs).
    - Its datatype must mutch the memory type.

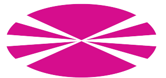
Ecole
Internationale
des Sciences
du Traitement
de l'Information



Scientific data libraries
Integration of NetCDF and HDF5

- NetCDF-4 is a joint project by Unidata and the HDF Group.
- It integrates both formats, using HDF-5 as the persistence layer of a NetCDF interface:
  - Allows to represent hierarchical data.
  - Preserves the simplicity of the NetCDF interface (this is only noticeable using the C and Fortran interfaces, not the Python one).
- NetCDF-4 is still compatible with previous NetCDF versions.
- It is capable of creating complex dataspaces, as well as groups, multiple unlimited dimensions, compression, and parallel I/O.

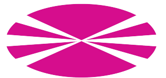
Ecole
Internationale
des Sciences
du Traitement
de l'Information



PyTables

- PyTables is a hierarchical database engine based in HDF5.
- Designed to efficiently managed large volumes of data.
- Optimizes memory and disk usage so that final size is smaller (particularly if compression is enabled) that using relational or object databases.
- Allows to define **tables** similar to those in a relational database.
- But each column can be a scalar or a NumPy array (i.e., multidimensional).
- A PyTables file stores both tables and NumPy arrays.

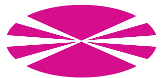
Ecole Internationale des Sciences du Traitement de l'Information



PyTables
Characteristics

- Supports up to 2^{63} rows per table.
- Unlike relational databases, each cell in a hierarchical table can have any dimensionality.
- Supports indexing any column of the table.
- Users can define metadata the give semantical meaning to their data.
- Supports data compression through Zlib, LZO, bzip2, and Blosc.
- Inherits HDF5 characteristics: supports files larger than 2GB, high-performance I/O, platform-independent files, ...

Ecole
Internationale
des Sciences
du Traitement
de l'Information

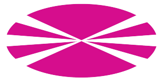


PyTables
Characteristics

`Notebook-012.ipynb`

- All the objects of a PyTables tree are instances of the `Node` class.
- The `Group` and `Leaf` classes inherit from `Node`: a group contains other groups and leaf objects (similar to HDF5).
- A leaf object is a data container, an can be an instance of the following classes:
  - `Table`.
  - `Array`.
  - `CArray`: *chunked*, allows data compression.
  - `EArray`: *extensible*, allows to enlarge one of its dimensions.
  - `VLArray`: *variable length*, its rows have variable number of columns.
  - `UnImplemented`: HDF5 data unsupported by PyTables.

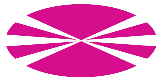
Ecole Internationale des Sciences du Traitement de l'Information



Blaze

- NumPy arrays and Pandas `DataFrames` are the foundational structures of Python as a data analysis tool.
- However, processing capabilities are restricted by physical memory.
- Besides, loading data into Pandas is not always trivial (e.g., parsing JSON, XML, SQL, etc.).
- Blaze has two main goals:
  - To integrate different computational backends and data storage technologies transparently.
  - To trivialize out-of-core computing.

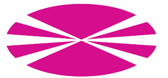
Ecole Internationale des Sciences du Traitement de l'Information



Blaze
Data sources

- Native Python objects (lists, dictionaries, …).
- NumPy.
- Pandas.
- SQL.
- HDF5.
- MongoDB.
- Spark.
- Impala.

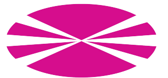
Ecole Internationale des Sciences du Traitement de l'Information



Blaze

- Blaze does not compute anything.
- It “simply” performs data transformations and bridges software interfaces to make different data sources and computational backends interact an a cohesive way.
- It employs functionalities which already exist in the Python ecosystem.
- Allows to conveniently transform data from one format to another (e.g., CSV to SQL).
- Blaze is not as mature as Pandas. Its documentation is often incomplete, and its APIs change frequently.

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Blaze

`Notebook-013.ipynb`

- A Blaze *server* provides a web-services API to data.

- Any data source accepted by Blaze can be exposed through this API.

- The server can be accessed in different ways:
  - Using programs written in any language, e.g., using the `requests` module, to build JSON calls.
  - Using the `blaze` module, defining a data source with the server adress through the `blaze://` protocol.

Ecole Internationale des Sciences du Traitement de l'Information

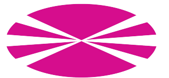
Seaborn

`Notebook-014.ipynb`

- Seaborn is a graphics library which improves the design and appearance of Matplotlib plots and includes additional statistical methods:
  - It allows to use themes to unify plot aesthetics.
  - Adds functions to visualize and compare distributions on one and two variables.
  - Linear regression tools.
  - Functions for visualizing data matrices and clustering.
  - Plotting statistical time series.

- Similar to `ggplot` in R.

Ecole Internationale des Sciences du Traitement de l'Information

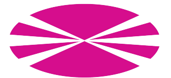


Bokeh

Notebook-015.ipynb

- Bokeh is another graphical library with different objectives.
- Part of Continuum's (Anaconda) analysis tools, together with Blaze.
- Focuses on building plots that can be visualized as HTML with Javascript.
- Creates *interactive* plots.
- Provides output in different formats, such as HTML, inlined into an iPython Notebook, or PNG.
- Does not currently allow to create vector versions (e.g., SVG).

Ecole
Internationale
des Sciences
du Traitement
de l'Information



Performance analysis

- Python is intrinsically inefficient when executing array codes (i.e., scientific/engineering codes).

- (Except if said codes can be “vectorized” using NumPy).

- However, it is very good at I/O-bound codes such as data analytics.

- Ideally, we could use Python as an universal language, including:
  - Data generation (main computation).
  - Data analytics (statistics, *crunching*).
  - Report generation (plots).

Ecole Internationale des Sciences du Traitement de l'Information

Performance analysis

- We focus now on implementing efficient scientific codes on Python.
- The development cycle for scientific codes is generally incremental: analysis, implementation, debugging, evaluation, rinse and repeat.
- During performance analysis it is fundamental to understand the bottlenecks and their causes to correct them in a future iteration.
- Profiling is the process which investigates the behavior of the code, developing a model which allows to correct the observed problems in successive development phases.

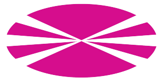
Ecole Internationale des Sciences du Traitement de l'Information

Performance analysis
Measuring runtime

`Notebook-016.ipynb`

- The execution time of a section of code is a useful performance metric.
- The `time` module allows to measure the execution time of a block of code.
- iPython also provides the `%time` and `%timeit` magic commands.
- However, we need to investigate how execution time is distributed across the code.
- It is not convenient to include a great deal of profiling code including calls to `time` and/or `print`.
- Python provides powerful tools to analyze code performance in an automatic way, avoiding (most) changes to the code.

Ecole Internationale des Sciences du Traitement de l'Information



Performance analysis
`cProfile`

`Notebook-016.ipynb`

- The `cProfile` module, part of the Python standard library, interacts with the Python VM to measure the time spent inside each function in a code.
- Its output is basically identical to that of GNU prof: a list of functions including number of times each one was called, cumulative execution time, and time per call.
- Due to the required instrumentation, executing code through `cProfile` is much slower than a regular execution.
- Information is provided **at the function level** only, i.e., no information can be inferred about which parts of a function are more costly.

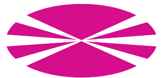
Ecole Internationale des Sciences du Traitement de l'Information

Performance analysis
`line_profiler`

Notebook-016.ipynb

- The `line_profiler` module analyzes the time spent executing each line of a function.
- Its output is similar to that of `cProfile`, but instead of offering data at the function level, it does so at the line-of-code level.
- As any other profiling tool, it introduces execution overheads. In this case, much larger to those of `cProfile`.
- For this reason, the ideal chain of tools is: analyze the full code with `cProfile`, and then do a fine-grained analysis of the critical functions in the code with `line_profiler`.

Ecole Internationale des Sciences du Traitement de l'Information



Compiling to C: Cython

`Notebook-016.ipynb`

- Cython is a framework allowing to compile Python code into C.
- A Cython code is similar to OpenCL: there is a main program written in Python which calls computational kernels written in annotated Python (`.pyx` files).
- This `.pyx` file is translated to C by a source-to-source compiler.
- The C code obtained from an un-annotated `.pyx` is very similar to the original Python code: it calls the Python interpreter to gain dynamic knowledge about data types and function calls.
- If the `.pyx` with the types of the variables in the code it is possible to save runtime checks, and therefore interpreter calls, getting dramatic performance returns with minimal effort.

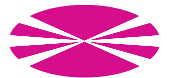
Ecole Internationale des Sciences du Traitement de l'Information

Compiling to C
Parallelism in Python

`Notebook-016.ipynb`

- One of the most controversial characteristics of Python is the *Global Interpreter Lock* (GIL), which prevents two different threads to execute Python bytecode at the same time.
- The GIL has several advantages (single-threaded execution is more efficient, integration with C libraries, etc.), but it clearly does not benefit parallelism.
- **It does not affect I/O-bound operations**. A process releases the GIL before waiting for I/O.
- Cython allows automatic parallelization through OpenMP.
- The limitations imposed by the GIL are worked-around by working with native C types, avoiding calls to the interpreter.

Ecole
Internationale
des Sciences
du Traitement
de l'Information

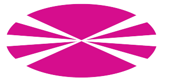


Compiling to C
Numba

`Notebook-017.ipynb`

- Numba is a tool similar to Cython: it generates C code from Python code using *Just-In-Time compilation* (JIT).
- Besides their interfaces, the main difference between Numba and Cython is the framework used for source-to-source compiling. It uses its own LLVM implementation.
- Numba provides the `@ji` decorator, as well as `numba.jit()`, to compile particular functions.
- The paid NumbaPro version also allows to automatically generate GPGPU code.

Ecole Internationale des Sciences du Traitement de l'Information

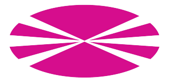


Parallel Python
`threading`

`Notebook-018.ipynb`

- The `threading` module allows to write multithreaded programs.
- It works well for I/O-bound codes, but not in CPU-bound ones except when the computation uses C libraries intensively (such as Cython or NumPy).
- Parallelism model similar to POSIX threads (`pthreads`).

Ecole Internationale des Sciences du Traitement de l'Information



Parallel Python
`multiprocessing`

`Notebook-018.ipynb`

- The `multiprocessing` module imitates the `threading` interface, but creates processes instead of threads.

- Each process executes on its own Python interpreter, therefore circumventing the single-GIL limitation.

- However, different processes do not share memory. The `Manager` class allows to share different types of Python objects (lists, dictionaries, mutexes, queues, arrays, …).

Ecole Internationale des Sciences du Traitement de l'Information