Algorithmic and programming

2018 - 2019

Python

- Developed by Guido van Rossum.
- Interpreted language
- Dynamically typed

Python

- Python interpreter

```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15)
 [MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python

- Python interpreter
  - Evaluation of inputs

```
>>> 3+2
5
>>> "Hello "+"world"
'Hello world‘
>>> "you"*3
'youyouyou'
```

Python

- The interpreter
  - To exit python interpreter

Exit() or ctrl-D (Unix or Mac OS) or ctrl-Z (windows)

Python

- Indentation matters to the meaning of the code. It is used to indicate a block of statements
- Variable type do not need to be declared, Python figures out variables types on its own.
- To assign a value to a variable, = is used.

Python

- **Basic datatypes**

Integer

```
x=3
y= x+3
```

Floats

```
x=3.5
y= 5/2
```

Strings

```
x= ’Hello’
y=  "Hello"
z= """ Hello """
```

Booleans x=True

Python

- **Operators:**

Assignment :

Simple x=7

Simultaneous x=y=7

Concurrent x,y=7, 53.2

check the type : type(x)

Test the type : isinstance(x,float)

Python

- **Operators:**

  Arithmetic operators : + , -, *, /, %

Operator	Description
a+b	Addition of a and b
a-b	Subtraction of b from a
a*b	Multiplication of a with b
a/b	Division of a over b
a//b	Floor division of a over b
a**b	a power of b

Python

- **Operators:**

  Logical operators : and, or, not

  Relational operators : >, <, >=, <=, ==, !=

Python

- **Bonus**

  x+=1 (x=x+1)

  x-=1 (x=x-1)

  x*=1 (x=x*1)

Python

- **Input and output statements**

Output : print (expression1, expression2, …)

Example: print(“the value is :”, x)

Input : input ([string])

Examples: x=input(“please enter a value”)
x=input()

Python

- **Control structures**

Conditional : if (condition):
statements

Example:

```
if  x>0:
    x= x+1
    z= x/2
```

Both statements are executed if x>0

```
if x>0:
    x=x+1
z=x/2
```

only the first statement is conditional

Python

- **Control structures**

Conditional : if (condition):
    statements_if
  else:
    statements_else

Python

- **Control structures**

Conditional :
```
if (condition):
    statements_1
elif (condition):
    statements_2
else:
    statements_3
```

Python

- **Control structures**

Loop statement : while (condition):
statements_1

Example

```
i=1
while i<10:
    print(i, “looping”)
    i+=1
```

Python

- **Control structures**

Loop statement : while (condition):
statements_1

Example

```
i=1
while i<10:
    print(i, “looping”)
    i+=1
```

Python

- **Control structures**

Loop statement : for element in sequence:
statements

Examples

- for i in range (0,3):
  print(i, “looping”)

- str= “abcd”
  for i in str:
  print i

Python

- **Control structures**

Loop statement :

Break stops the loop

Continue stops the current iteration and immediately go to the next one

Functions

Python

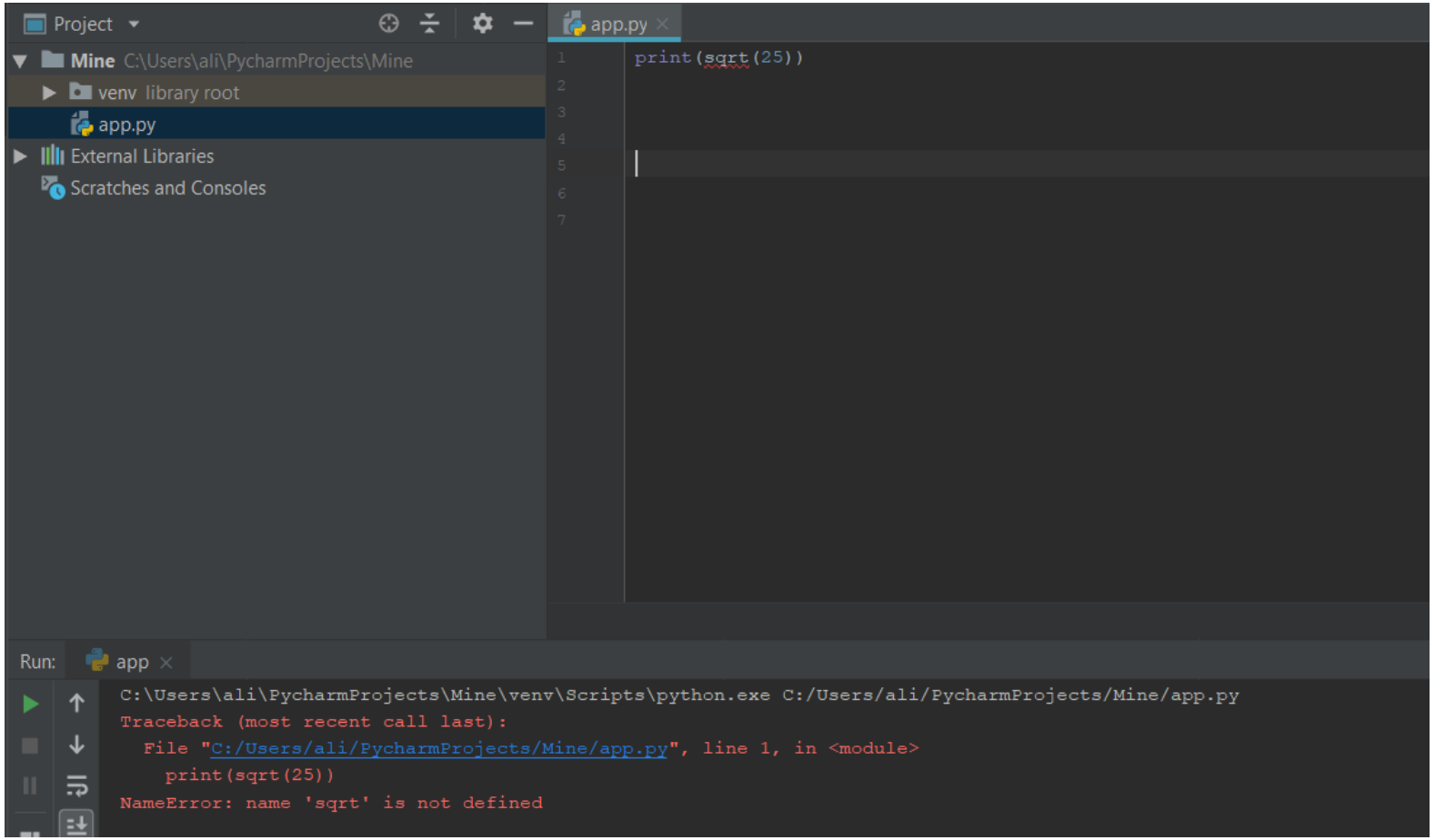
- **Library functions**

  Examples: print(), range(), sqrt()

  You just need to import appropriate libraries.

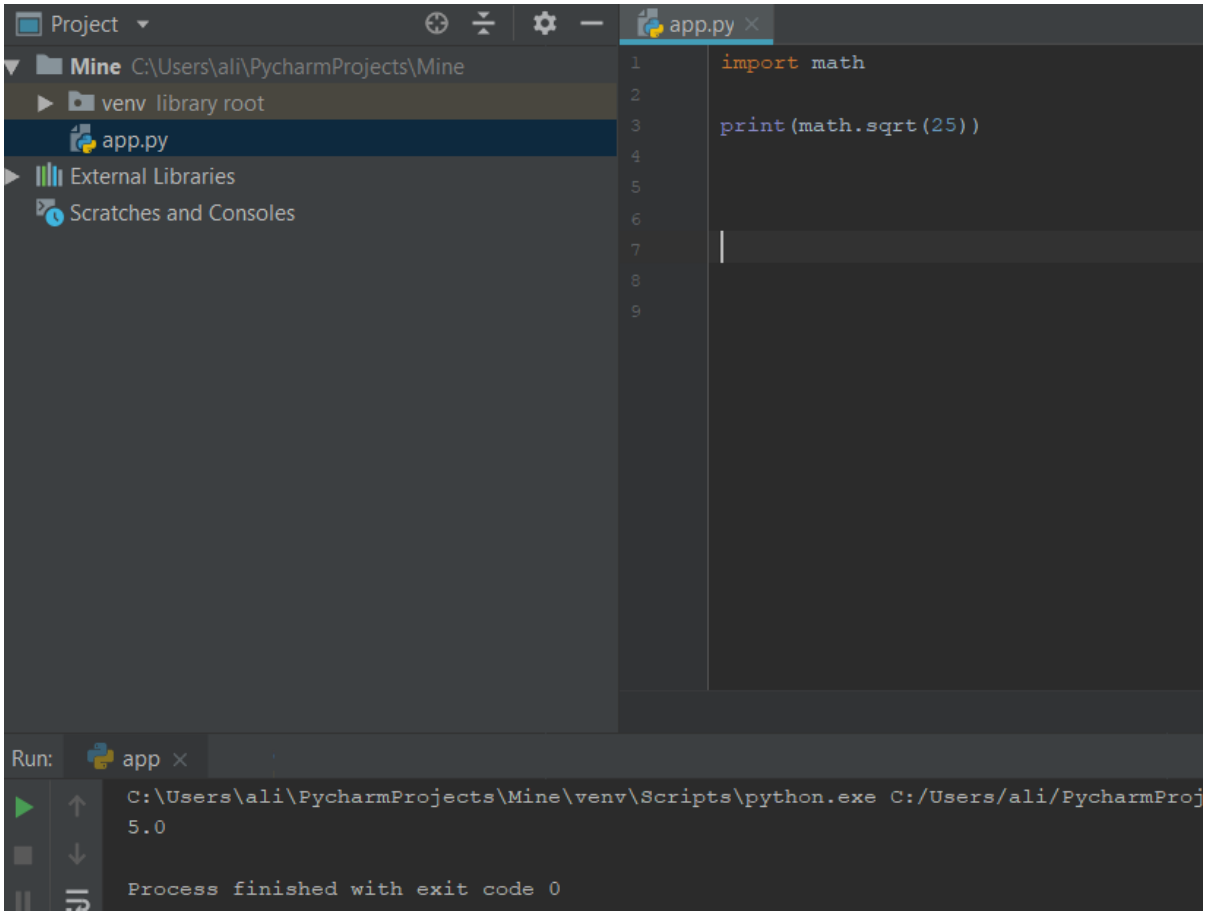
Python

- **Library functions**



Python

- **Library functions**



Python

- **User defined functions**

Python

- **User defined functions**

  Syntax:

  def function_name (parameter1, parameter2, …):

  body of the function

- ✓ The types of the parameters are determined implicitly by their use.
- ✓ Parameters are optional
- ✓ If a function does not return a value explicitly, it returns NONE.

Python

- **Example:**

```
def add(a, b):
    result = a + b
    return result
```

Python

- **Example:**

```
def message():
    print('Welcome to EISTI')
```

Python

- **Calling a function**
  - ✓ Passing arguments to a function

Example:

```
print(add(5, 3))
```

Python

- **Calling a function**
  - ✓ Call by value
    - ➢ Example:

```
message()
```

Python

- **Calling a function**
  - ✓ Always done by value except for some mutable types.
    - ➢ The value of the variable is passed to the function as a parameter.
    - ➢ Different memory is allocated to the parameters.

Python

- **Calling a function**
  - ✓ Call by value
    - ➢ Example:

```
def add(a, b):
    a=a+1
    result = a + b
    return result


a=3
print(add(a, 3), a)
```

Python

- **Exercises:**
  - ✓ Write a function to compute the maximum of three given values.
  - ✓ Write a function to compute the perimeter along with the area of a given.
  - ✓ Write a function that return the mirror of a given string.

Tuples

Python

- **Tuples**
  - ✓ Immutable ordered sequence of heterogeneous type items.

    Immutable: cannot be modified once created.

    Heterogeneous: mixed types.

Python

- **Tuples**
  - ✓ Immutable ordered sequence of heterogeneous type items.

```
x= (23, "test", (3.14, "ouch"),5)
```

x[0]

x[2][1]

Python

- **Tuples**
  - ✓ Immutable ordered sequence of heterogeneous type items.

```
x= (23, "test", (3.14, "ouch"),5)
```

Count from the end

x[-1]

Python

- **Exercise**

  Write a function that returns the name and the age of the user

Arrays (lists)

Python

- **List**
  - ✓ Sequence of values
  - ✓ Each value identified by its position

Python

- **List**
  - ✓ Sequence of values
  - ✓ Each value identified by its position
  - ✓ Python counts from 0.
  - ✓ Can count from the end.
  - ✓ Use function len to get the length of a list

Python

- **List**

  - ✓ Printing each element of the list

```
primes=[2,3,5,7,11,13]
for i in primes:
    print(i)
```

Python

- **List**
  - ✓ Adding one to each element of the list

```
primes=[2,3,5,7,11,13]
i=0
while i<len(primes):
    primes[i]=primes[i]*2
    i= i + 1
```

Python

- **List**
  - ✓ Compute the sum

```
primes=[2,3,5,7,11,13]
sum=0
for i in primes:
    sum+=i
```

Python

- **Add an element to the list**
  - ✓ Append()

Python

- **Lists functions**
  - ✓ Reverse()
  - ✓ Sort()
  - ✓ Insert (index, element)
  - ✓ Remove(element)