Algorithmic and programming

2019 - 2020

Course Objective

- Learn about the fundamentals of algorithms.
- Get started with programming (Python)

Course Objective

- Learn about the fundamentals of algorithms.
- Get started with programming (Python)

Everyone must and will write programs

“Practice does make perfect"

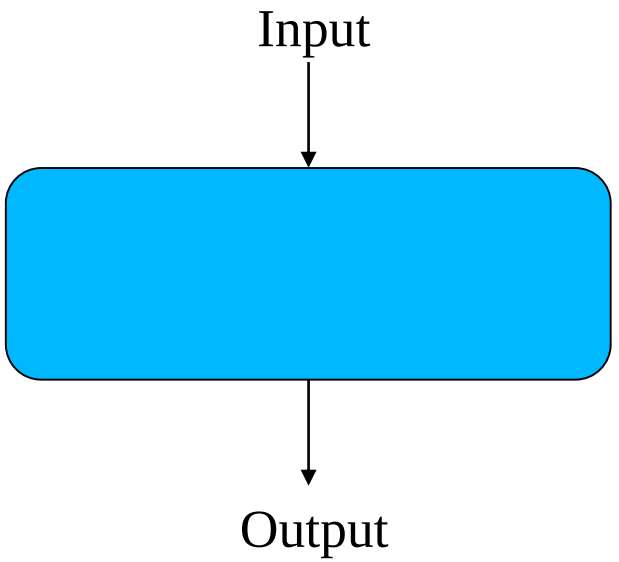
Algorithmic vs. Programming

- *Algorithmic*
  - Named after a Persian mathematician known as: Al-Khowarizmi
  - Design a solution (an algorithm) for a given problem
- *Programming*
  - Tell the machine what action to perform
    - *"This machine has no brain. Use your own"*
  - Transferring the content of an algorithm into a language the computer understands.

Algorithm

"A well defined computational procedure that takes some value, or set of values as inputs and produces some value or a set of values, as outputs" (Cormen *et al.*)

- **In other words**: An algorithm is a sequence of computational steps that transform the input into the output.



How to describe an algorithm?

- Natural language?

How to describe an algorithm?

- Natural language?

  Good for daily life but it lacks structure and it is ambiguous as it depends on context.

How to describe an algorithm?

- Natural language?

  Good for daily life but it lacks structure and it is ambiguous as it depends on context.

- Programming language?

How to describe an algorithm?

- Natural language?

  Good for daily life but it lacks structure and it is ambiguous as it depends on context.

- Programming language?

  Good as it is understood by machines but not when we try to solve a problem. The focus is shifted from how to solve the problem to details of syntax.

How to describe an algorithm?

- Pseudo-code

  A structured English that lets you think at an abstract level about the problem. Simple and readable.

Example

Problem: French apple tart

Ingredients

Pastry:

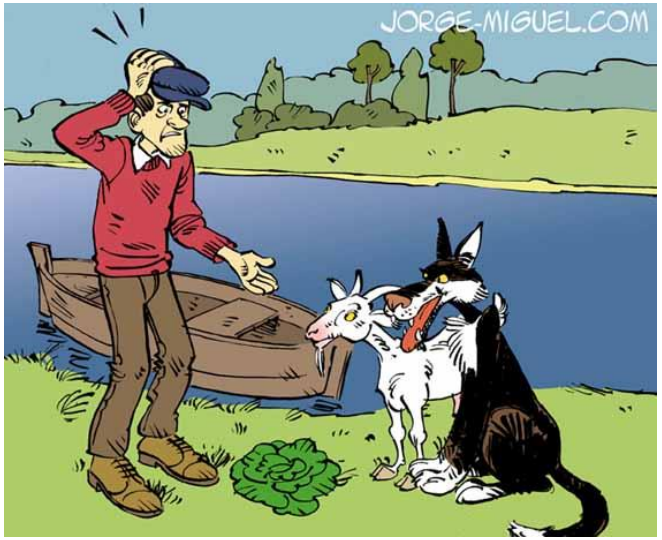
- 1 1/3 cups all-purpose flour
- 1 pinch salt
- 1/2 cup butter, softened
- 1 egg yolk
- 3 tablespoons cold water, or as needed

Frangipane:

- 1/2 cup butter, softened
- 1/2 cup white sugar
- 1 egg, beaten
- 1 egg yolk
- 1 tablespoon apple brandy
- 2/3 cup ground almonds
- 2 tablespoons all-purpose flour
- 4 medium sweet apples - peeled, cored, halved and thinly sliced
- 1 teaspoon white sugar for decoration
- 1/4 cup apricot jelly

allrecipes.com

Example



River crossing puzzle

A farmer must transport a goat, a wolf and cabbage from one side of a river to another using a boat.

- The boat can only hold one item in addition to the farmer.
- The wolf cannot be left alone with the goat.
- The goat cannot be left alone with the cabbage.

Design an algorithm that solves the river crossing puzzle

Example

Sorting algorithms

Description of the problem:

- Sort a series of number in the increasing order.
- Formally

Input: a series of n numbers $(a_1, a_2, \ldots, a_n)$

Output: rearrange the input series into $(a'_1, a'_2, \ldots, a'_n)$ such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

Example

Sorting algorithms

Description of the problem:

- Sort a series of number in the increasing order.
- Formally

Input: a series of n numbers $(a_1, a_2, \ldots, a_n)$

Output: rearrange the input series into $(a'_1, a'_2, \ldots, a'_n)$ such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

For instance : From the series (31, 41, 59, 26, 41, 58), a sorting algorithm outputs (26, 31, 41, 41, 58, 59)

Correctness of an algorithm

Correctness of an algorithm

- An algorithm is said to be correct only if it outputs correct results for all input instances.

Correctness of an algorithm

- An algorithm is said to be correct only if it outputs correct results for all input instances.
- Incorrect algorithm ?

Correctness of an algorithm

- An algorithm is said to be correct only if it outputs correct results for all input instances.
- Incorrect algorithm ?

  - Either it outputs an incorrect value or does not output any value.

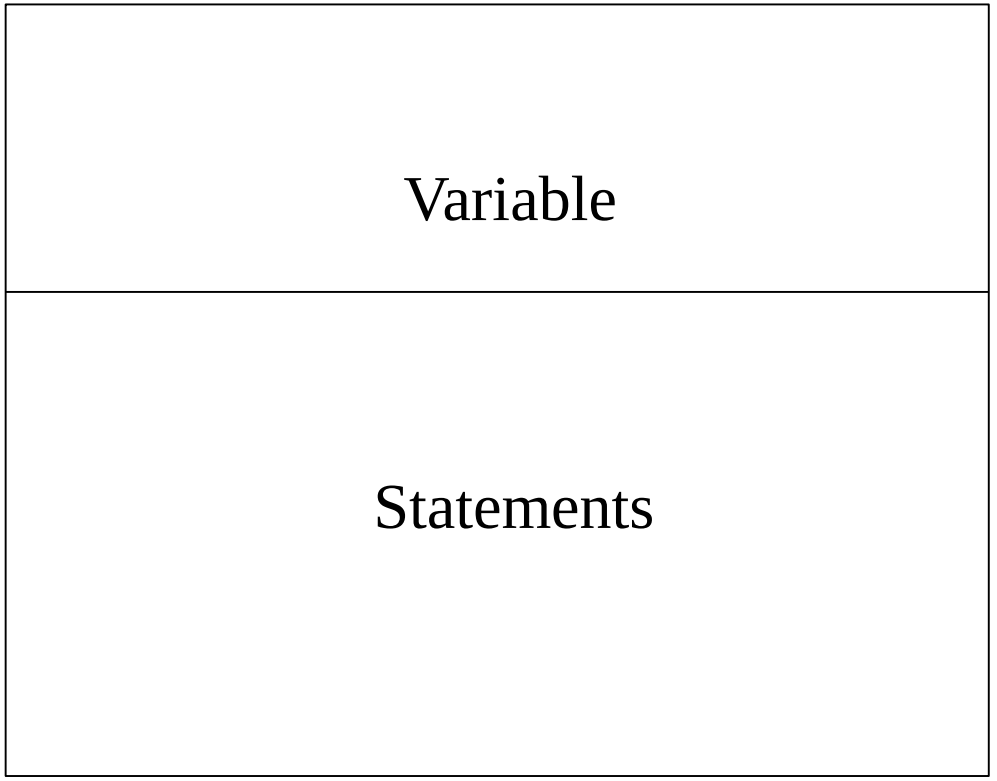
  - Can be sometimes useful if the error rate is controlled.

Correctness of an algorithm

- Given a problem, there may be more than one correct algorithm. However, the cost of each algorithm may be different.
- Cost is measured in different ways
  - In term of time
  - In term of space

} Complexity of an algorithm

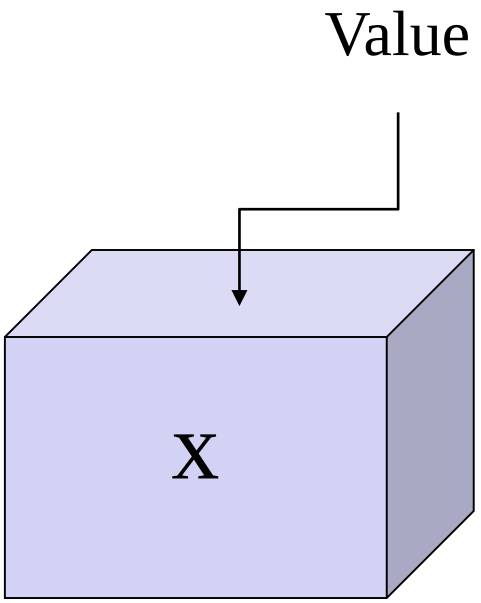
Algorithm (Pseudo-code)



An algorithm consists of a series of finite number of statements that involve variables

Variables

- A named memory location that can store a value.



Variables

- A named memory location that can store a value.
- Has a type.

Variables

- A named memory location that can store a value.
- Has a type: A variable's type determines the values that the variable can have and the operations that can be performed on it

Variables

- A named memory location that can store a value.
- Has a type: A variable's type determines the values that the variable can have and the operations that can be performed on it

Type	Domain
Integer	$\mathbb{Z}$
Float	$\mathbb{R}$
Character	Unicode characters
String	A sequence of characters
Boolean	True / False

Variables

Operators

Integers

Name	*Symbol*
Addition	+
Subtraction	-
Multiplication	*
Integer division	div
Modulo	mod

Variables

Operators

Real

Name	*Symbol*
Addition	+
Subtraction	-
Multiplication	*
Division	/

Variables

Operators

String

Name	*Symbol*
Concatenation	&

Variables

Operators

Logical

Name	*Symbol*
Conjunction	and
Disjunction	or
Negation	Not

Variables

Operators

Comparison

Name	*Symbol*
Equal to	=
Less than	<
Greater than	>
Less or equal	<=
Greater or equal	>=

Variables

Each variable needs to be defined before being used in an algorithm.

Variables

Each variable needs to be defined before being used in an algorithm.

Syntax Variable_name : type

Variables

Each variable needs to be defined before being used in an algorithm.

Syntax Variable_name : type

Examples

x, y : Integer

a : Boolean

r1, r2 : Float

mot : String

Expression

- A combination of one or more constants, variables and operators.

Expression

- A combination of one or more constants, variables and operators.
- The type of an expression is the type of its resulting value.

Expression

- A combination of one or more constants, variables and operators.
- The type of an expression is the type of its resulting value.

Examples

(x> 0) and (y>0) :

Expression

- A combination of one or more constants, variables and operators.
- The type of an expression is the type of its resulting value.

Examples

(x> 0) and (y>0) : Boolean

Expression

- A combination of one or more constants, variables and operators.
- The type of an expression is the type of its resulting value.

Examples

(x> 0) and (y>0) : Boolean

x mod 2 :

Expression

- A combination of one or more constants, variables and operators.
- The type of an expression is the type of its resulting value.

Examples

(x> 0) and (y>0) : Boolean

x mod 2 : Integer

Statement

- An action to be executed

Statement

1. Assign a value to a variable.

Syntax variable_name <- expression

Statement

1. Assign a value to a variable.

Syntax variable_name <- expression

Example

x ←2 // x must be defined as an integer

y ←x+3 // y must also be defined as an integer

w ← 'hello'; // w must be declared as a string

sum ←8 * x + (y – x DIV 2) * 3; // sum must be defined as an integer. Variables x and y must be initialized

Statement

1. Input / Output statement.

Statement

1. Input / Output statement.

Transfer values from the user to a variable, or vice-versa

Statement

1. Input / Output statement.

Transfer values from the user to a variable, or vice-versa

Syntax

Get(name_variable)

Print(name_variable)

Statement

1. Input / Output statement.

Transfer values from the user to a variable, or vice-versa

Syntax

Get(name_variable)

Print(name_variable)

Example

Get(x)

Print(x)

Print(“Welcome”)

Algorithm

Example

```
PROGRAM test
VARIABLES
    message : String
Begin
    Print("something to say ?")
    Get(message)
    Print("You said : " & message)
END.
```

Conditional statements

Conditional statement

Syntax:

if <condition> **then**
    statements
Else
    statements
Endif

Conditional statement

Syntax:

if <condition> **then**
    statements
Else
    statements
Endif

<condition> : Boolean expression

Conditional statement

Syntax:

if <condition> **then**
    statements
Else
    statements
Endif

<condition> : Boolean expression

- Comparison operators

  = , <>, >=, <=, >, <

- Boolean operators

Conditional statement

Example:

```
PROGRAM Adult
VARIABLES
    age : integer
BEGIN
    Print(" Enter your age ")
    Get(age)
    if (age > 18) then
        Print("You're an adult")
    else
        Print(" A bit more to go ")
    FINSI
FIN
```

Conditional statement

- Syntax

```
Switch variable
    CASE value_1 : (* statements_1 *)
    CASE value_2 : (* statements_2 *)
    CASE value_3 : (* statements_3 *)
    [...]
    CASE value_n : (* statements_n *)
    DEFAULT (* Default statments*)
ENDSwitch
```

Loop statements

Loop statements

WHILE

- Repeat a sequence of statements *as long as the condition is true*
- Syntax

```
WHILE <condition>
    (* Statements *)
EndWhile
```

Loop statements

Repeat until

- Repeat a sequence of statements *as long as the condition is false*
- Syntax

```
Repeat
    (* Statements *)
Until <condition>
```

Loop statements

For

- Repeat a sequence of statements *for a given number of times*
- Syntax

```
For i=1 to n step 1 do
    (* Statements *)
EndFor
```

Loop statements

Example

Write an algorithm that computes the factorial of a given number n.

Functions

Functions

A set of statements that perform a task. Can be seen as a sub program that takes an input and returns an output.

Syntax

Function name (parameters) : type of the returned value
Variables
set of variables along with their respective types
Begin
statements
return …
End

Functions

A set of statements that perform a task. Can be seen as a sub program that takes an input and returns an output.

Example

```
Function square (n: integer) : integer
Variables
        result : integer
Begin
        result ← n * n
        return result
End
```

Procedures

Procedures

Similar to a function but does not return an output.

Syntax

Procedure name (parameters)
Variables
    set of variables along with their respective types
Begin
    statements
End

Procedure

Sub program that takes an input and does not return an output.

Example ax+b=0

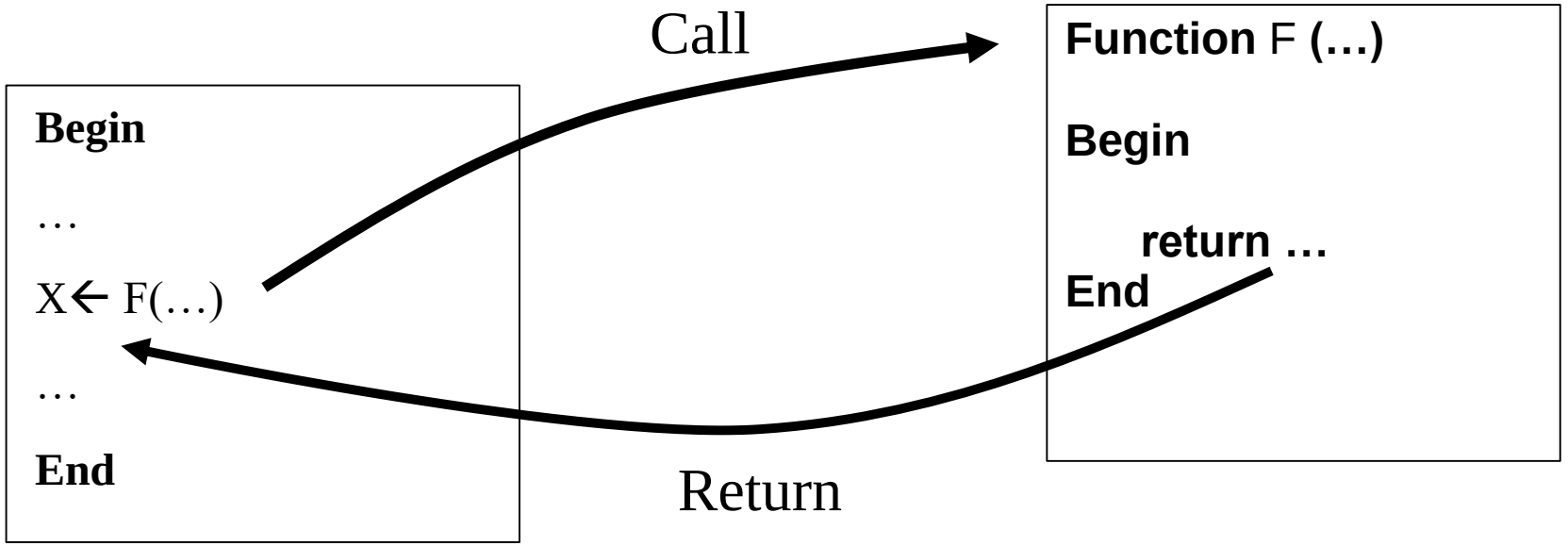
```
Procedure equation (a, b: integer)
Variables
        result : integer
Begin
    if a=0 then
        if b=0 then
            print(" infinite number of solutions")
        else
            print("no solution")
        endif
    else
        x ← -b/a
    endif
End
```

The execution starts from the first statement of a program !

Call to a function

Beginning of the execution.

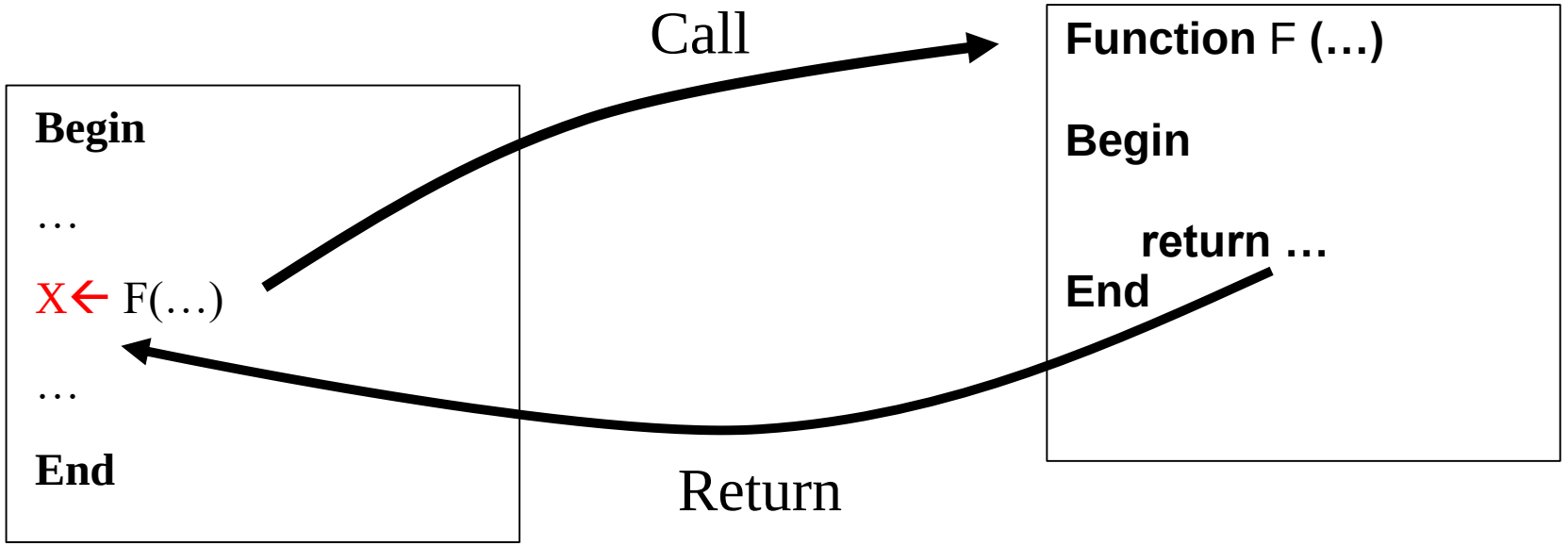
Example



Call to a function

Beginning of the execution.

Example



Example

```
Function square (n: integer) : integer
Variables
        result : integer
Begin
    result ← n * n
    return result
End
```

Example

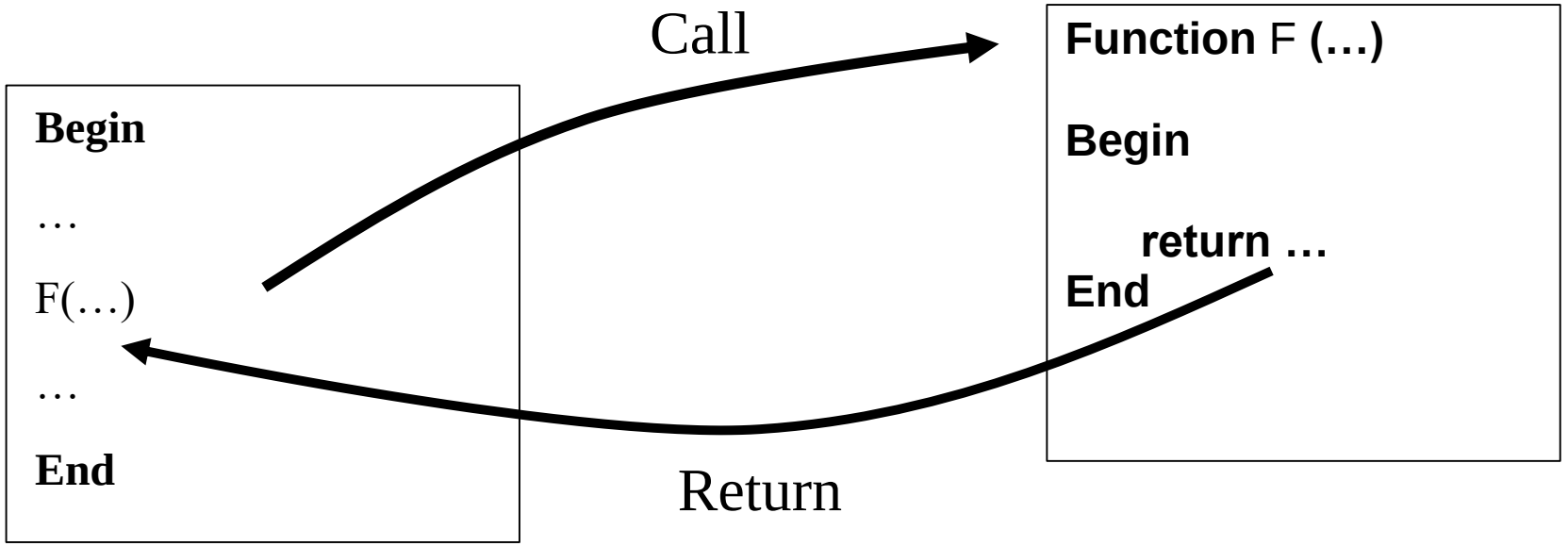
```
Function square (n: integer) : integer
Variables
        result : integer
Begin
    result ← n * n
    return result
End
```

```
Program Test
Variables
                x : integer
Begin
        x ←square (12)
        print(x)
End
```

Call to a procedure

Beginning of the execution.

Example



Example ax+b=0

```
Procedure equation (a, b: integer)

Variables

        result : integer

Begin

    if a=0 then
      if b=0 then
         print(" infinite number of solutions")
      else
         print("no solution")
      endif
   else
      x ← -b/a
   endif
End
```

Example ax+b=0

```
Procedure equation (a, b: integer)

Variables

        result : integer

Begin

    if a=0 then
      if b=0 then
         print(" infinite number of solutions")
      else
         print("no solution")
      endif
    else
      x ← -b/a
    endif
End
```

```
Program Test

Variables

            x, y : integer

Begin

        Get(x,y)

        equation(x,y)

        print(x)

End
```

Exercise

Write a function that computes x^y

Exercise

Given two integers a and b, write a program that displays a^b

Exercise

Given three integers a, b and c, write a procedure that solves the following equation: $ax^2+bx+c=0$

Exercise

Given three integers a, b and c, write a program that displays the solution of the following equation: $ax^2+bx+c=0$

Local vs Global variables

Local variable

Declared and visible only within a function or a procedure.

Global variable

Visible in the program by all the functions and procedures.

Example

Function Test1 (y: integer): integer

Variables

test : integer

Begin

test ← y+5
y ← 10
return test

End

Program Test

Variables

x, result : integer

Begin

x ← 7

result ← Test1(x)

print(result)

print(x)

End

Example

Variable x: integer

Function Test1 (y: integer): integer

Variables

test : integer

Begin

test ← y+5
y ← 10
return test

End

Program Test

Variables

result : integer

Begin

x ← 7

result ← Test1(x)

print(result)

print(x)

End

What if there are more than one result to compute?

Solutions:

1. Use a data structure

2. Use a procedure with input and outputs parameters

Example:

Write a procedure that computes the quotient and the remainder of a division.

Exercise

Variable x: integer

Procedure exchange (x,y: integer)

Variables

z : integer

Begin

z ← x
x ← y
y ← z

End

Program Test

Variables

a,b : integer

Begin

a ← 7

b ← 2

change(a,b)

print(a,b) // what is printed out ?

End

Example

Variable x: integer

Procedure exchange (**I/O** x,y: integer)

Variables

z : integer

Begin

z ← x
x ← y
y ← z

End

Program Test

Variables

a,b : integer

Begin

a ← 7

b ← 2

change(a,b)

print(a,b)

End

Benefits

- Provides modularity to the program.
- Easy code re-usability.

Records

What if there are more than one result to compute for a function?

Solutions:

1. Use a data structure.

2. Use a procedure with input and outputs parameters

Records:

User defined data type which allows us to combine data of different types together.

Syntax:

Type name

Record name

Field_name

name_1 : type_name_1

name_2 : type_name_2

…

name_3 : type_name_n

EndRecord

Records:

User defined data type which allows us to combine data of different types together.

Example:

To store student information (name, given name, age, address), we can define a new type t_student using records.

Record t_student

name : string

g_name: string

age : integer

address : string

EndRecord

Records:

User defined data type which allows us to combine data of different types together.

Example:

A date consists of the day, the month and the year.

```
Record t_date
    day : integer
    month: integer
    year : integer
EndRecord
```

Records:

User defined data type which allows us to combine data of different types together.

Variable declaration

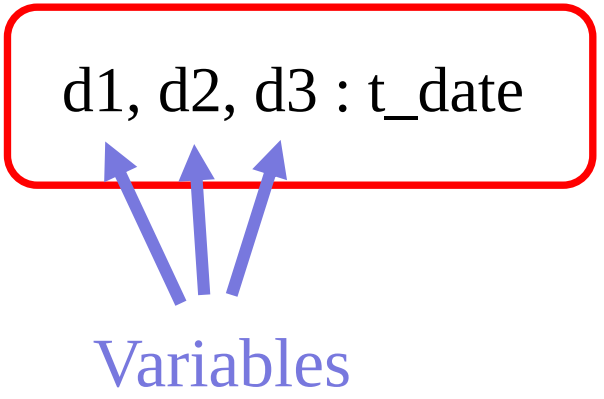
Record t_date

day : integer

month: integer

year : integer

EndRecord



Records:

User defined data type which allows us to combine data of different types together.

Access to a field

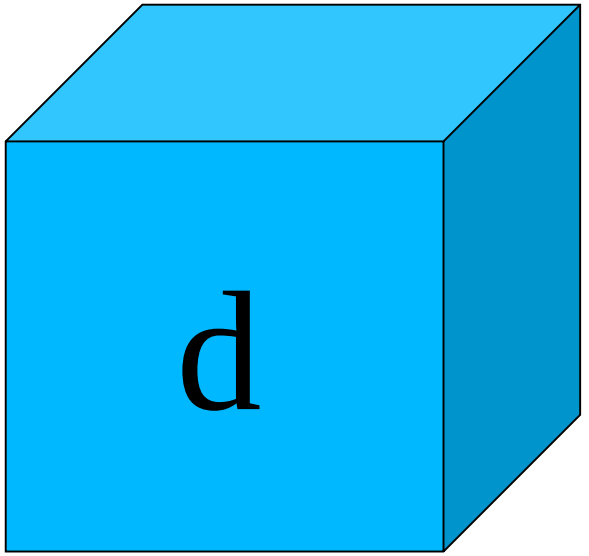
Record t_date

day : integer

month: integer

year : integer

EndRecord



d : t_date

Records:

User defined data type which allows us to combine data of different types together.

Access to a field

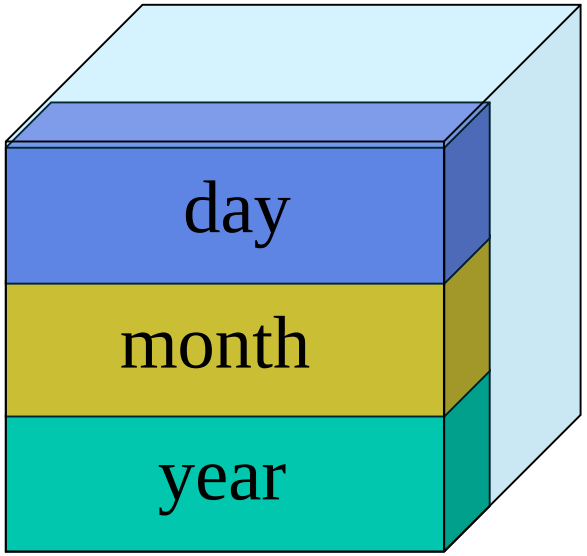
Record t_date

day : integer

month: integer

year : integer

EndRecord



d : t_date

Records:

User defined data type which allows us to combine data of different types together.

Access to a field

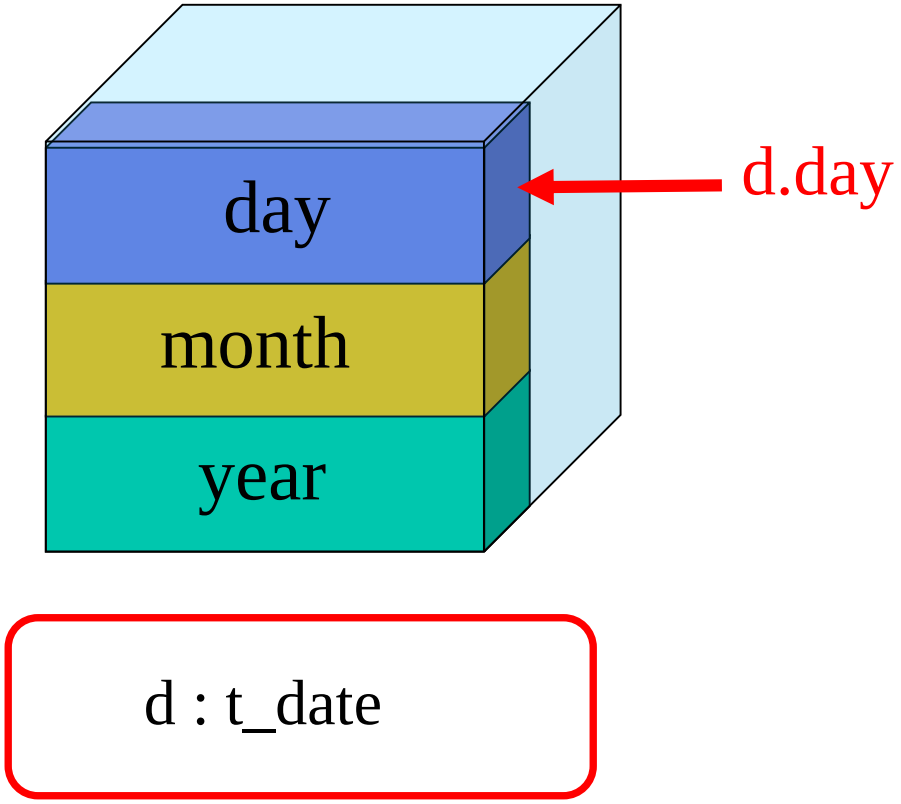
Record t_date

day : integer

month: integer

year : integer

EndRecord



Records:

User defined data type which allows us to combine data of different types together.

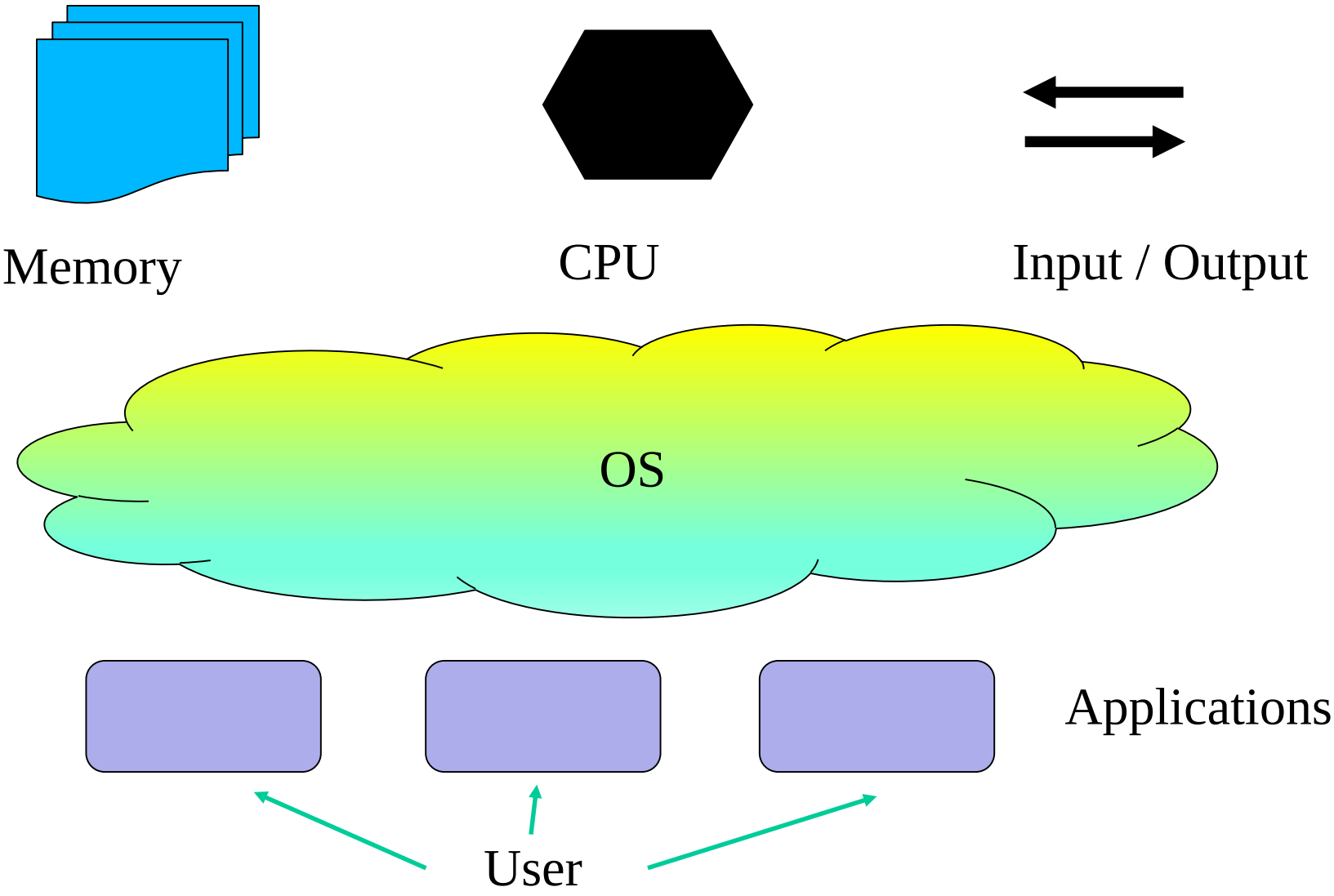
Access to a field

Syntax

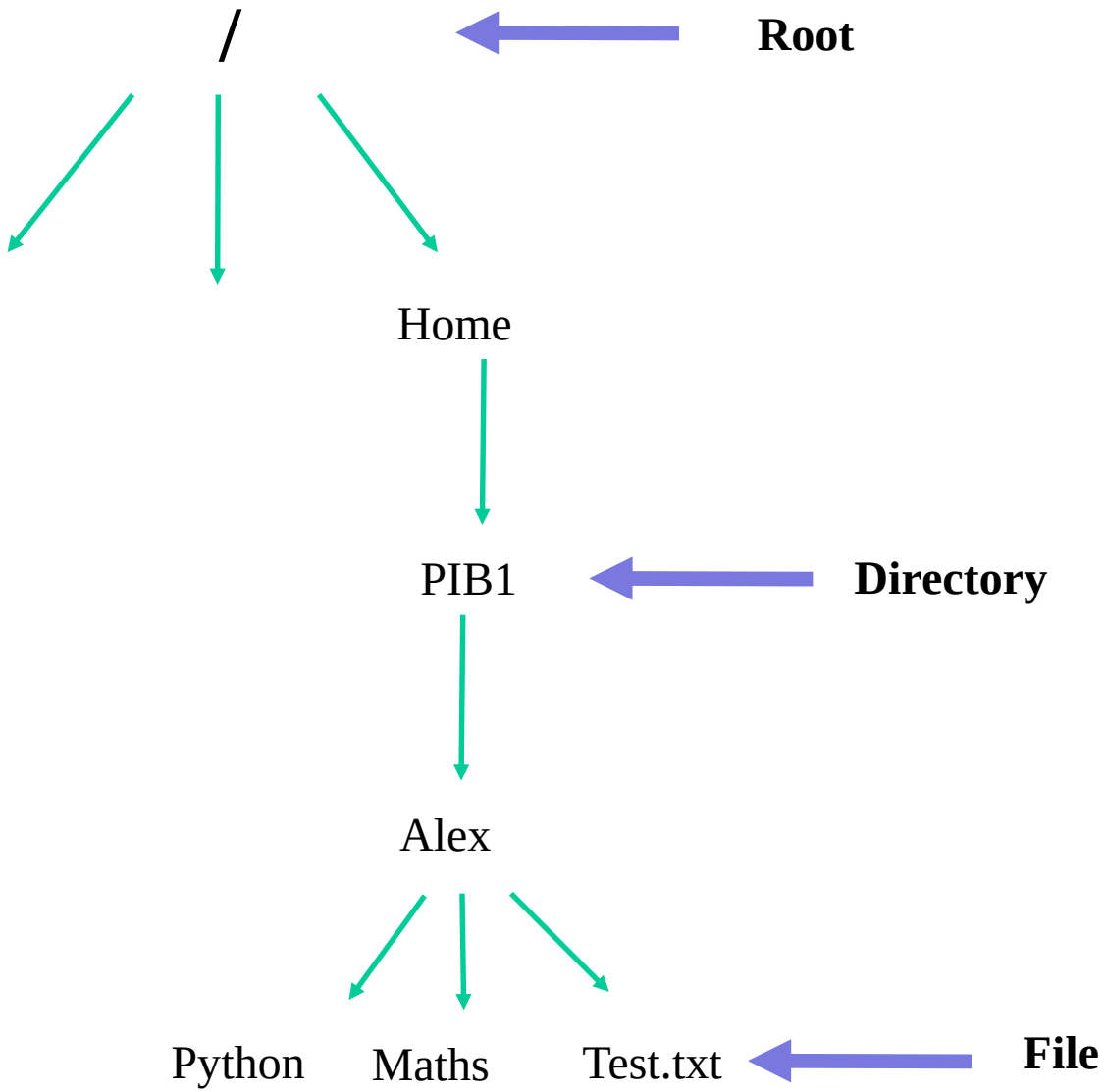
Variable_name.field_name

Shell

Operating System ?



Unix file system



Shell

Command line interpreter that provides a command line user interface for Unix-like operating systems.

Shell command

Syntax

command_name options argument

Get information about a command

man command_name

Basic shell commands

Listing

ls : list files and directories in the current directory.

ls -a : list all directories and files including dot files

ls -l : use long listing format

Basic shell commands

Path

A unique location to a file or a directory in a file system

Absolute path

Specify the location of the file or directory from the root directory (/)

Relative path

Specify the location of the file or directory with respect to the current position.

Basic shell commands

Navigation

cd : go to home director.

cd *path* : go to the directory indicated by the path

cd .. : go up one directory

Basic shell commands

Position

pwd : show the present working directory.

Basic shell commands

File/ directory manipulation

mkdir directory_name : create a new directory

touch file_name***:*** create a new file.

move source destination : move a file from source to destination

rm file_name: remove a file or directory with –r option.

cp source destination : copy a file from source to destination.

Basic shell commands

Permissions

r : READ

w : WRITE

x : EXECUTE

Basic shell commands

Permissions

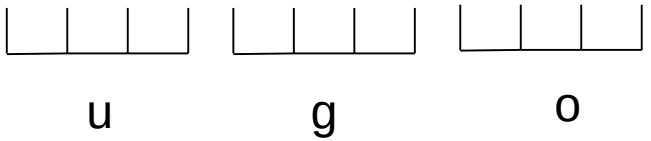
•**Owner permissions** (u) determine what actions the owner of the file can perform on the file.

•**Group permissions** (g) determine what actions a user of the group the file belongs to, can perform on the file.

•**Other permissions** (o) indicate what action all other users can perform on the file.

Basic shell commands

Permissions



Basic shell commands

Permissions

chmod: command used to change directory/file permissions

chmod og-w file_name : remove write permission to the group and the others

chmod g+x file_name : allow the group to execute the file_name.

Basic shell commands

Text manipulation

grep pattern file: search for the lines in file that contain pattern

sort file : sort file lines lexicographically**.**

Basic shell commands

Viewing file

cat file_name: display the file_name content

Basic shell commands

Viewing file

head –n file_name: display the first n lines of file_name

Basic shell commands

Viewing file

tail option file_name:

-n x: display the last x lines

-n +x: display from line x

Basic shell commands

Viewing file

More **:** display page per page

Basic shell commands

Viewing file

wc option file_name:

-c: display the number of characters

-w: displays the number of words

-l: displays the number of lines

Basic shell commands

Meta characters

*: any sequence of characters

?: one character

[]:a character from either a set or an interval

Basic shell commands

Meta characters

ls *.txt

ls ???

ls [ct]*

Basic shell commands

Input / output redirection

Standard input : keyboard

Standard output : screen

Output redirection : >, >>, 2>

Basic shell commands

Pipe |

Use more than one command such that the output of one command serves as the input to the next

Example: ls | sort