Programing - Introduction

- Goals :
  - Learn how to think with algorithms.
  - Translate into informatic programs.
- Tools :
  - Python programming language .
  - text editor **Emacs** - programming environment.
  - Operating System **Unix - Linux** .

My first program in Python

- Respect the tradition
- Code:

```
print "Hello World"
```

- Or by using variable

```
c= "Hello World"
print c
```

How to run Python

- Run intrepreter and type the code when prompt appears `>>>`.
  Each time you type `Enter` your line is executed.
- By using a **source** file and run it in the shell window.

```
> python hello.py
```

Python rules

- Case sensitive language.
- Dynamic typing.
- Instruction has to begin in the first column of the line.
- Indentation has the same meaning than parentheses. It marks the block.
- Composed instructions have to end by : at the first line (header) followed by a set of instructions with indentation.

Python rules

- If an instruction is more than one line, a continuation character
  or `()` has to be added.

```
if a == b and c == d and \
   d == e :
       print ok
if (a == b and c == d and d == e) :
       print ok
```

Python types

Type	Example
Numbers	124 3.02 9999L 4.0e+2 3+4j
String	'spam' "d'guido"
List	[1,[2,'trois'],4]
Tuples	(1,'spam',4,'U',0)
Dictionary	{'food' : 'jam', 'taste': 'miam'}

Python types

- `integer` :
  - Size : 4 octets (32 bits) from -2 147 483 648 to 2 147 483 649
  - if too huge the error `overflow error` is raised.
- `float` :
  - Size : 8 octets (64 bits) de 10^{-308} à 10^{308}
  - if too huge no error but display `Inf` (infinity).
- `long` :
  - No limit - just the available memory on the machine.

Binary Coding

- Computers are binary machine - just 2 values exist.
- All variables, instructions are translated by using binary conversion.
- The smallest value is the *bit* - it takes 0 or 1 value.
- A machine word is a 8 bits word - i.e. an **octet**.

Binary Coding

- Example of integer coding:

Decimal Value	Binary Value
0	0
1	1
2	10
3	11
4	100

Binary Coding

- Adding two binary numbers

Decimal Value	Binary Value
4	100
+	
3	11
=	
7	111

Binary Coding

- Read a binary number

Binary Value	1	0	1	0	1	1
Power 2	$1 * 2^5$	$0 * 2^4$	$1 * 2^3$	$0 * 2^2$	$1 * 2^1$	$1 * 2^0$
Decimal Value	32 +	0 +	8 +	0 +	2 +	1
Result	43					

Hexadecimal Code

- Hexadecimal code is on 16 bits.
- Calcul is done with 16 base.
- Symbols are from $0 - 9ABCDEF$.
- Currently and largely used to code the colors - RGB system.

Character Coding

- For each character a numerical value is given.
- ASCII code: American Standard Code for Information Interchang.
- Written with 7 bits - i.e. 127 possible values.
- From 0 to 31 control characters.
- From 65 to 90 upper case characters.
- From 97 to 122 lower case characters.

Character Coding

- Translate an upper case character to the same lower case is adding 32 to the value - i.e. changing the 6*ieme* bit.
- Numbers are coded from 48 to 57.
- Space char is 32.
- Unix and windows have not the same value for the end of line - CR 13 and LF 10.
- Accents are not included in the ASCII code.
- ISO-Latin 1 takes into account with the 8*iem* bit.
- UNICODE is the new way to take into account all languages on 16 bits.

Python Types

- `String` :
  - Ordered collection of characters.
  - If a string is included in ”’ it can be set on several lines.
  - c=” is the empty string.
  - No `char` type.
  - No memory management by the user.

Operators

- Affectation :
  - $a = 3$
- Arithmetic :
  - $+ \quad - \quad * \quad / \quad \% \quad **$
- Comparison :
  - $< \quad > \quad <= \quad >= \quad ==$
    $<> \quad != \quad !$
- Logic :
  - or, and, not.
- Operations on bits :
  - $<< \quad >> \quad | \quad \&$

Control structures

- Syntax:`if`
  - Syntax :
    `if` (boolean test) :
        instructions if true
    `else` :
        instructions if false.
  - Example :
    `if ( a % 2 == 0):`
        print “a is even ”
    `else` :
        print “a is odd”

Loop and iterations

- Goal : repeat several times the same piece of code.
- Example : Ask 10 numbers to the user.
- Code to repeat:

```
print 'give a number'
number = input()
```

- Instruction:
  - `while <test>:`

Loop and iterations

- Python Code:

```
counter = 1
while counter <=10:
      print 'give a number'
      number = input()
      sum = sum+number
      counter=counter+1
#end of loop
print 'the sum of the 10 numbers is``, sum
```

Loop and iterations

- Iterations: `for <target> in <object>:`
- Python Code :

```
days=('monday','tuesday','wednesday')
for item  in days:
    print item,
```

- Execution :

```
monday tuesday wednesday
```

Loop and iterations

- Display all the values between $0 - 9$:

```
for i in range (10):
    print i
```

- The function `range` creates a list of all values in the range $[0 - 9]$
- It can be written as `range(BeginningValue,EndValue,step)`
- Python Code :

```
for i in range (2,10,3):
    print i,
```

- Result : 2 5 8

Nested loop

- Do the sum of 10 weight values for an given animal list.
- Python Code :

```
animals=('rat', 'mouse', 'cobaye')
for animal in animals:
    result=0
    for i in range(10):
        print 'give the weight for', animal
        weight=input()
        result=result+ weight
    print 'Sum of weight for  ',animal,
    print 'in ', result
print 'end'
```

List

A list is a data structure which allows to access to the element by the number of the box containing the data.

- How to declare a list

```
mylist=[]
mylistNew=[1,3.5,'la maison']
```

- Use a list :

```
mylist.append(25)
print mylistNew[2]
```

- Size of the list : `len(mylist)`

Dictionary

A dictionary is a data structure which allows to access to the element by the name of the box containing the data. This name is called the `key` of the element.

- How to declare a dictionary :

```
mydic={}
mydictNew={''bread'':3, ''milk'': 5 ,
           ''butter'': ''vide''}
mydic[''animal'']=''chat''
```

- Use a dictionary :
  - list of keys : `mydicNew.keys()`
  - dictionary size : len(mydicNew)
  - list of elements : `mydicNew.items()`
  - Python 2 - ask if a key exists : `mydictNew.has_key("coffee")`

Function

- Definition :

```
def myFunction (myListOfParameter):
    list of instructions
```

- Call a function :
  `myFunction(myListOfParameter)`
- Collect a function result :
  `myvar=myFunction(myListOfParameter)`