# PROGRAMMATION PYTHON

**Formateur**:

**Noms**: Patrice SERUGENDO

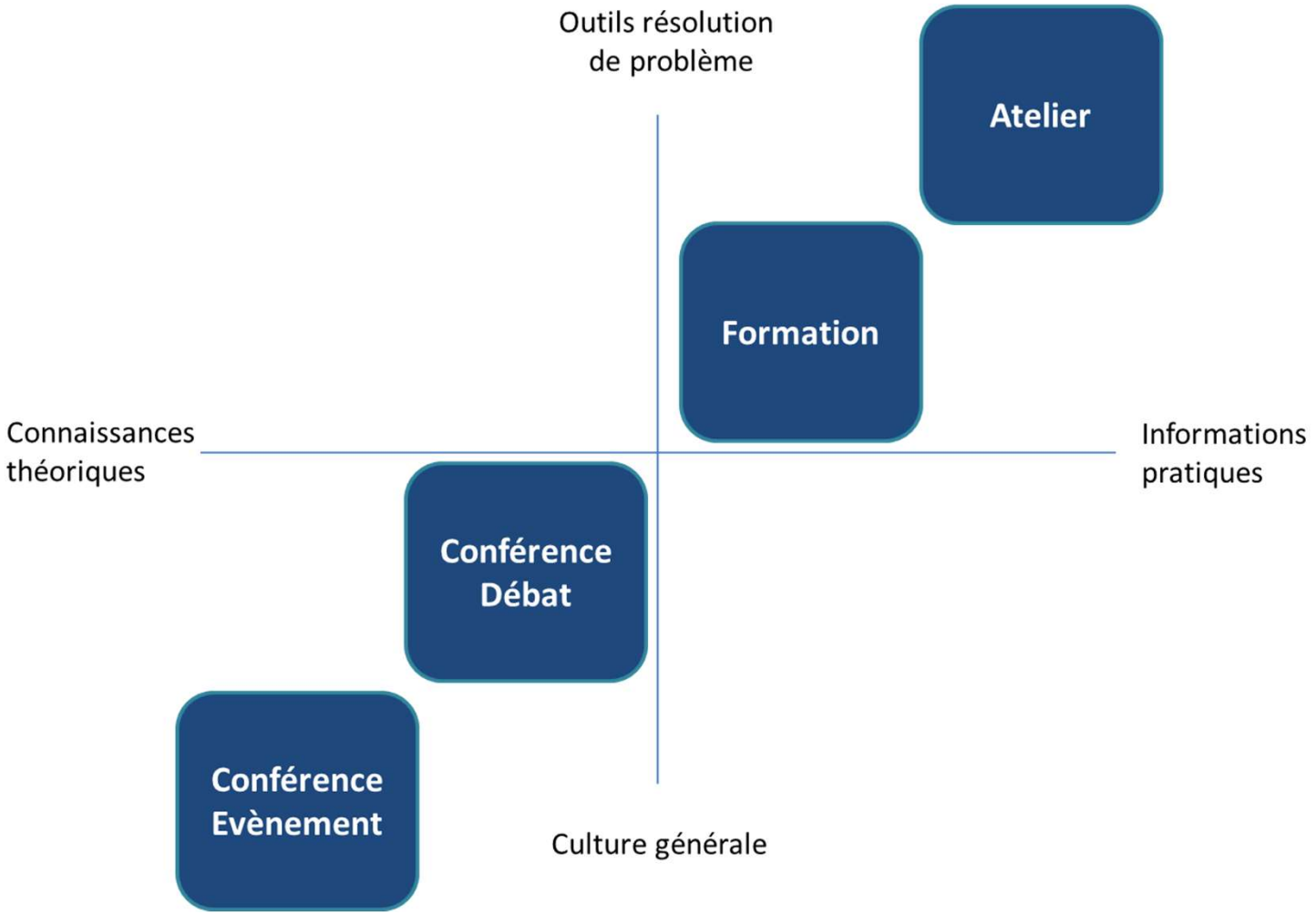
**Email**: patrice.serugendo@ifage.ch

# Introduction

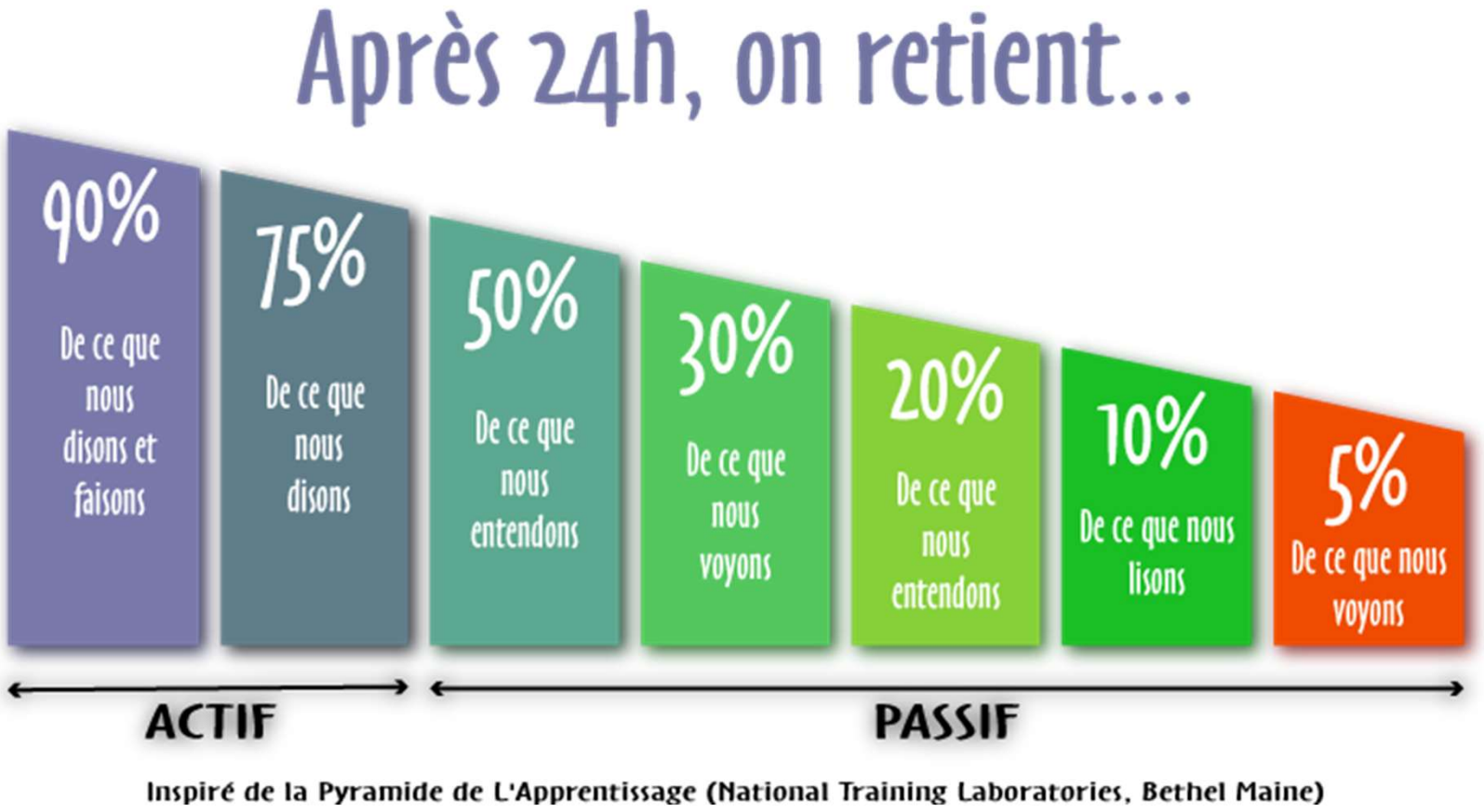
# Présentation Formateur

- **Noms:** Patrice Serugendo
- **Formation**:
  - Maîtrise en Informatique de Gestion, Université de Genève, Sun Certified Java Programmer – SCJP – 2000
  - Certificié FSEA (2014-2015) –> Brevet Fédéral de formateur d'adulte (BFFA) -- > 2019
- **Expérience**:
  - Java, C/C++, Android, iOS, CrossPlatform, Python
- **Contact**:
  - E-mail:patrice.serugendo@ifage.ch
  - Mobile: +41 78 845 46 78

# Cours Python: Objectif et moyen



# Efficacité des méthodes




Inspiré de la Pyramide de L'Apprentissage (National Training Laboratories, Bethel Maine)

# Méthodologies utilisées

- **Les méthodes pédagogiques**
  - **Méthode expositive, transmissive, passive ou magistrale**
  - **Méthode démonstrative**
  - **Méthode interrogative**
  - **Méthode active ou de découverte**
  - **Méthode expérientielle**

# Contenu

- Environnement
- Language
- Mots réservés & Fonctions
- Fonctions Lambda
- Gestion des fichiers et exceptions
- Programmation Orientée-Objet
- Expressions Régulières
- Accès Base de données (exemple avec MySQL et SQLite)

# Contenu (2)

- Programmation réseau (programmation socket serveur et client)
- Envoi des messages
- Programmation concurrente
- Traitement XML
- JSON et Python
- Interface-graphique utilisateur (avec Tkinter)
- Introduction au Blockchain

# Environnement

# Python: caractéristiques

- Interprété
- Intéractif
- Orienté-objet
- Facile à apprendre
- Dynamique
- Extensible
- Bibliothèque standard très riche
- Etc.

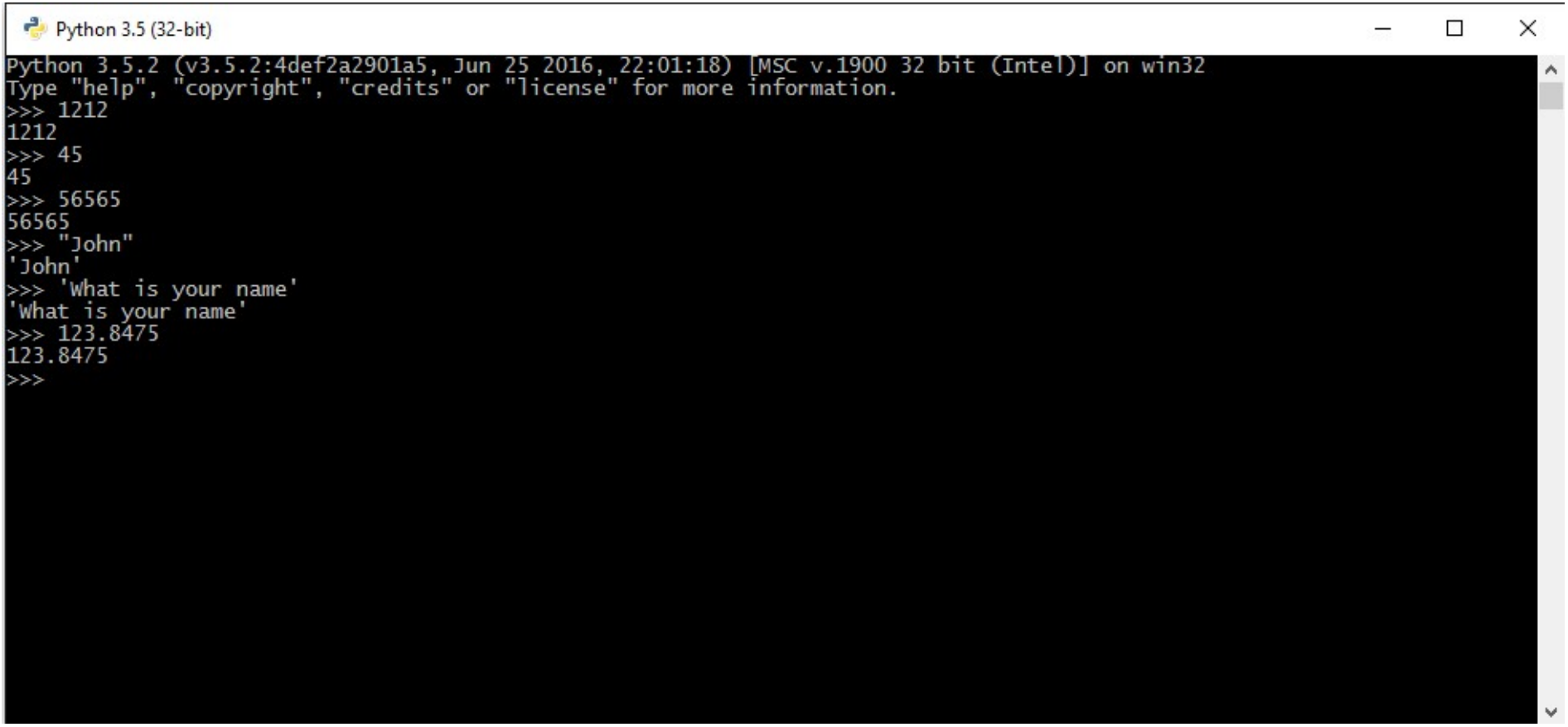
# Choisir l'interpreteur Python

- Interprêteurs Python:
  - CPython (original et le plus utilisé. Ecrit en C)
  - Jython (écrit en Java)
  - Python for .NET (utilise CPython et les librairies .NET)
  - IronPython (plus complet que Python for .NET et génère des apps .NET)
  - PyPy (écrit en Python)

# Installation de Python

- Télécharger et installer Python 3
  - https://www.python.org

# Python comme Perroquet



# Python comme calculatrice

| Opérateur | Description |
|---|---|
| + | Effectue une addition |
| - | Effectue une soustraction |
| * | Effectue une multiplication |
| / | Effectue une division |
| % | Effectue le reste d'une division (modulo) |
| ** | Effectue une opération d'exposant |

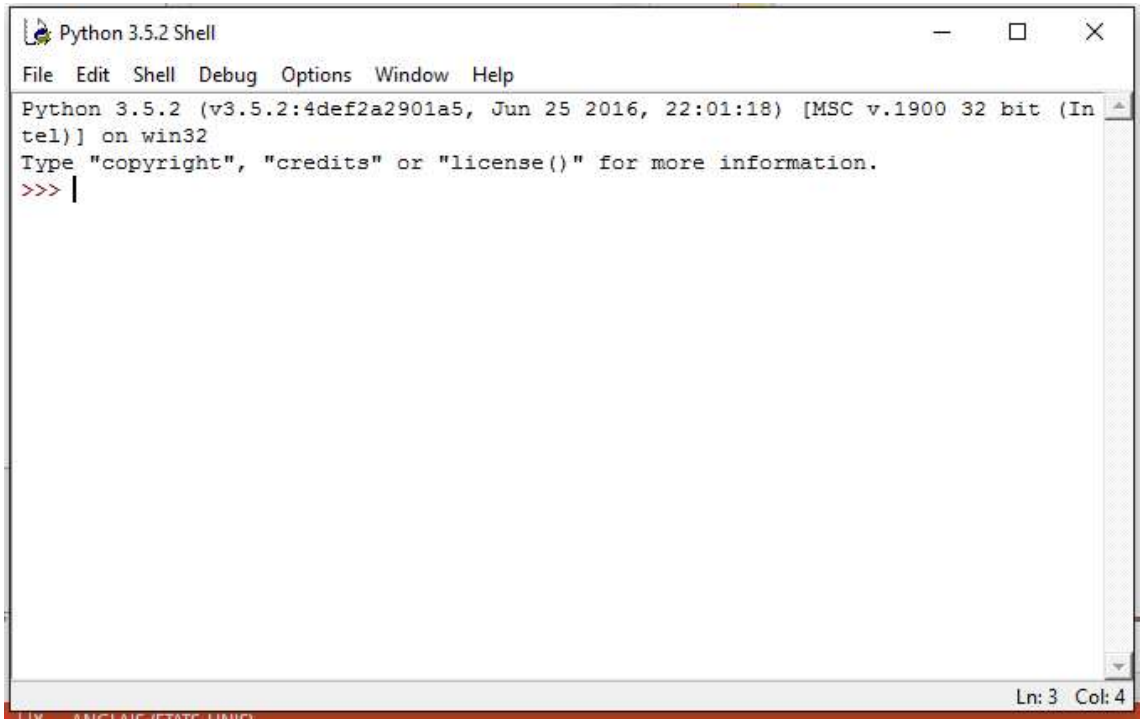
# Opérateurs de comparaison

| Opérateur | Description |
|---|---|
| == | Comparaison d'un égalité |
| != | Comparaison d'une différence |
| > | Comparaison de plus grand que |
| >= | Comparaison de plus grand ou égal que |
| < | Comparaison de plus petit que |
| <= | Comparaison de plus petit ou égal que |

# Opérations chaînes de caractères

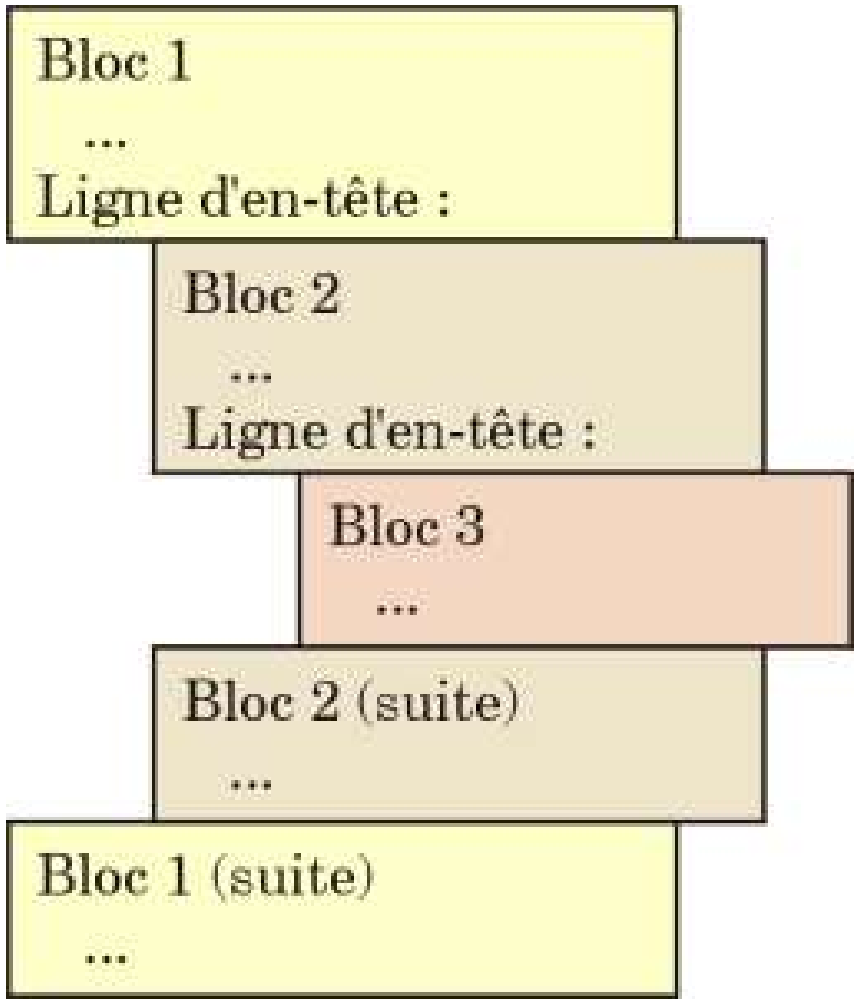
| Opérateur | Description |
|---|---|
| *sous-chaine* **in** *chaine* | Indique si une sous-chaine de caractères est contenu dans une chaine de caractères |
| *chaine1* + *chaine2* | Effectue la concaténation de chaine de caractères |
| *chaine* * *nombrerépétition* | Effectue une répétition de la chaine de caractères |
| u"*message*" | Définit une chaine de caractères de format *Unicode* |
| *chaine1* == *chaine2* | Comparaison d'un égalité |
| *chaine1* != *chaine2* | Comparaison d'une différence |
| *chaine1* > *chaine2* | Comparaison de plus grand que |
| *chaine1* >= *chaine2* | Comparaison de plus grand ou égal que |
| *chaine1* < *chaine2* | Comparaison de plus petit que |
| *chaine1* <= *chaine2* | Comparaison de plus petit ou égal que |

# IDE Python: IDLE



# Programmation

# Structure du programme




```
if embranchement == "vertébrés":              # 1
 if classe == "mammifères":                   # 2
     if ordre == "carnivores":                # 3
     if famille == "félins":                # 4
         print("c'est peut-être un chat")       # 5
     print("c'est en tous cas un mammifère")      # 6
 elif classe == "oiseaux":                    # 7
     print("c'est peut-être un canari")       # 8
print("la classification des animaux est complexe")  # 9
```

# Structure d'un programme

```
 1 # Créé par Chea, le 02/12/2013 en Python 3.2
 2 def afficher_tableau(tab):
 3     for i in range(len(tab)):
 4         for j in range(len(tab[i])):
 5             print(tab[i][j],"",end="")
 6         print("")
 7
 8 def pascal(n):
 9     tab=list(range(n+1))
10     for i in range(n+1):
11         tab[i]=list(range(i+1))
12         for j in range(i+1):
13             if j==0 or j==i:
14                 tab[i][j]=1
15             else:
16                 tab[i][j]=tab[i-1][j-1]+tab[i-1][j]
17     return tab
18
19 afficher_tableau(pascal(10))
```

# Modes de programmation

- Mode interactive

```
$ python
Python 3.0b3 (r30b3:66303, Sep  8 2008, 14:01:02) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

>>> 5
5
>>> print(5*7)
35
>>> "hello" * 4
'hellohellohellohello'
>>> "hello".__class__
<type 'str'>
```

- Mode script

```
python test.py
```

# Syntaxe

- Variable et nom de variable
- Mots réservés
- Lignes et identation
- Instructions multi-ligne
- Quotes
- Commentaires
- Lignes blanches
- Entrée/saisie de données-utilisateur
- Plusieurs instructions sur une seule ligne
- Arguments de la ligne de commande

# Variables

- C’est quoi une variable?
- Assigner / affecter une valeur à une variable
- Affectation multiple
- Types de données
  - Nombres
  - Chaînes de caractères
  - Listes
  - Tuple
  - Dictionnaire
- Conversion de types

# Types de variable

- Nombre:
  - Entier: 1, 947, -45, 2384755, 0xFFF, 0b00101011, 1e4
  - Décimal: 234.8569, 3/2
  - Complexe: 1+3j
- Chaînes de caractères:
  - “Ceci est un chaîne de caractère”
  - ‘Celle-ci aussi’
- Booléen:
  - True, False
- Type défini par l’utilisateur: class

# Opérateurs

- Arithmétiques: +, -, *, /, //, %, **
- Comparaison: ==, !=, >, <, >=, <=
- Affectations: =, +=, -=, *=, /=, %=, //=, **=,
- Binaires: &, |, ^, ~, <<, >>
- Logiques: and, or, not
- Appartenance: in, not in
- Identité: is, is not
- Régles de précédence

# Sélections & boucles

- Sélection: if …elseif … else
- Bouble: while, for, continue, break, pass
- Iterator & Generator: iter(), next()

# Sélection

```
var = 100

if ( var  == 100 ) : print ("Value of expression is 100")

print ("Good bye!")
```

# Boucle

```
list=[1,2,3,4]
it = iter(list) # this builds an iterator object
print (next(it)) #prints next available element in iterator
```

```
for x in it:
    print (x, end=" ")
```

```
while True:
    try:
        print (next(it))
    except StopIteration:
        sys.exit() #you have to import sys module for this
```

# Iterators & Generators

```
!usr//bin/python3
import sys
def fibonacci(n): #generator function
    a, b, counter = 0, 1, 0
    while True:
        if (counter > n):
            return
        yield a
        a, b = b, a + b
        counter += 1
f = fibonacci(5) #f is iterator object

while True:
   try:
      print (next(f), end=" ")
   except StopIteration:
      sys.exit()
```

# Comprendre yield

```
def fibonacci(n):
    """Fibonacci numbers generator, first n"""
    a, b, counter = 0, 1, 0
    while True:
        if (counter > n): return
        yield a
        a, b = b, a + b
        counter += 1
f = fibonacci(5)
for x in f:
    print (x)
print


def fibonacci2(n):
    """Fibonacci numbers generator, first n"""
    a, b, counter = 0, 1, 0
    result = []
    while True:
        if counter > n:
            return result
        result.append(a)
        a, b = b, a + b
        counter += 1
    #return result

f = fibonacci2(5)
print(f)
for x in f:
    print (x, end=" "),
print
```

# Mots réservés (keywords)

# Mots clés (keywords)

```
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
```

| | | | | |
|---|---|---|---|---|
| False | class | finally | is | return |
| None | continue | for | lambda | try |
| True | def | from | nonlocal | while |
| and | del | global | not | with |
| as | elif | if | or | yield |
| assert | else | import | pass | |
| break | except | in | raise | |

# and, or, not

```
>>> True and False
False
>>> True or False
True
>>> not False
True
```

| A | B | A and B |
|---|---|---|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

| A | B | A or B |
|---|---|---|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

| A | not A |
|---|---|
| True | False |
| False | True |

# True, False

```
>>> 1 == 1
True
>>> 5 > 3
True
>>> True or False
True
>>> 10 <= 1
False
>>> 3 > 7
False
>>> True and False
False
```

```
>>> True == 1
True
>>> False == 0
True
>>> True + True
2
```

# None

```
>>> None == 0
False
>>> None == []
False
>>> None == False
False
>>> x = None
>>> y = None
>>> x == y
True
```

```
def improper_return_function(a):
    if (a % 2) == 0:
        return True

x = improper_return_function(3)
print(x)
```

**Output**

```
None
```

```
def a_void_function():
    a = 1
    b = 2
    c = a + b

x = a_void_function()
print(x)
```

# as (alias)

```
>>> import math as myAlias
>>>myAlias.cos(myAlias.pi)
-1.0
```

# assert

```
>>> a = 4
>>> assert a < 5
>>> assert a > 5
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
AssertionError
```

```
>>> a = 4
>>> assert a > 5, "The value of a is too small"
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
AssertionError: The value of a is too small
```

At this point we can note that,

```
assert condition, message
```

is equivalent to,

```
if not condition:
    raise AssertionError(message)
```

# break, continue

```
for i in range(1,11):
    if i == 5:
        break
    print(i)
```

```
for i in range(1,11):
    if i == 5:
        continue
    print(i)
```

# class

```
class ExampleClass:
    def function1(parameters):
        …
    def function2(parameters):
        …
```

# def

```
def function_name(parameters):
    …
```

# del

```
>>> a = b = 5
>>> del a
>>> a
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
NameError: name 'a' is not defined
>>> b
5
```

# if, elif, else

```
def if_example(a):
    if a == 1:
        print('One')
    elif a == 2:
        print('Two')
    else:
        print('Something else')

if_example(2)
if_example(4)
if_example(1)
```

# try, except, else, finally

```
def reciprocal(num):
    try:
        r = 1/num
    except:
        print('Exception caught')
        return
    return r

print(reciprocal(10))
print(reciprocal(0))
```

```
>>> def reciprocal(num):
...     if (num == 0):
...             raise ZeroDivisionError("Cannot devide by zero")
...     r = 1/num
...     return r
...
>>> print(reciprocal(0))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in reciprocal
ZeroDivisionError: Cannot devide by zero
>>> print(reciprocal(10))
0.1
>>>
```

```
try:
    Try-block
except exception1:
    Exception1-block
except exception2:
    Exception2-block
else:
    Else-block
finally:
    Finally-block
```

# for

```
names = ['John','Monica','Steven','Robin']
for i in names:
    print('Hello '+i)
```

# from, import

- >>> import sys
- >>> sys.path

- >>> from sys import path
- >>> path

# global, nonlocal

```
globvar = 10
def read1():
    print(globvar)
def write1():
    global globvar
    globvar = 5
def write2():
    globvar = 15

read1()
write1()
read1()
write2()
read1()
```

```
def outer_funciton():
    a = 5
    def inner_function():
        nonlocal a
        a = 10
        print("Inner function: ",a)
    inner_function()
    print("Outer function: ",a)

outer_funciton()
```

```
def outer_funciton():
    a = 5
    def inner_function():
        a = 10
        print("Inner function: ",a)
    inner_function()
    print("Outer function: ",a)

outer_funciton()
```

# in

```
>>> a = [1, 2, 3, 4, 5]
>>> 5 in a
True
>>> 10 in a
False
```

```
for i in 'hello':
    print(i)
```

# is

```
>>> True is True
True
>>> False is False
True
>>> None is None
True
```

```
>>> [] == []
True
>>> [] is []
False
>>> {} == {}
True
>>> {} is {}
False
```

```
>>> '' == ''
True
>>> '' is ''
True
>>> () == ()
True
>>> () is ()
True
```

# lambda

```
>>> def f (x): return x**2
...
>>> print f(8)
64
>>>
>>> g = lambda x: x**2
>>>
>>> print g(8)
64
```

```
>>> def make_incrementor (n): return lambda x: x + n
>>>
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>>
>>> print f(42), g(42)
44 48
>>>
>>> print make_incrementor(22)(33)
55
```

# pass

```
def function(args):
    pass
```

```
class example:
    pass
```

# return

```
def func_return():
    a = 10
    return a

def no_return():
    a = 10

print(func_return())
print(no_return())
```

# while

```
i = 5
while(i):
    print(i)
    i = i - 1
```

# with

```
with open('example.txt', 'w') as my_file:
    my_file.write('Hello world!')
```

# yield

```
def city_generator():
    yield("London")
    yield("Hamburg")
    yield("Konstanz")
    yield("Amsterdam")
    yield("Berlin")
    yield("Zurich")
    yield("Schaffhausen")
    yield("Stuttgart")
```

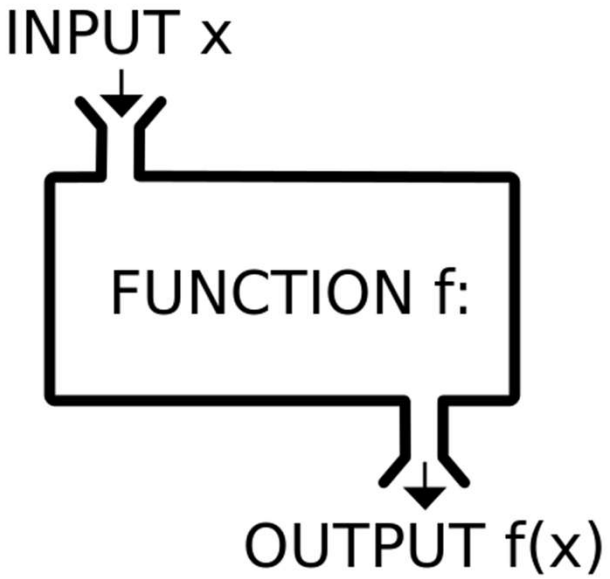
```
>>> from city_generator import city_generator
>>> city = city_generator()
>>> print(next(city))
London
>>> print(next(city))
Hamburg
>>> print(next(city))
Konstanz
>>> print(next(city))
Amsterdam
>>> print(next(city))
Berlin
>>> print(next(city))
Zurich
>>> print(next(city))
Schaffhausen
>>> print(next(city))
Stuttgart
>>> print(next(x))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

# yield ( suite )

```
def fibonacci(n):
    """Ein Fibonacci-Zahlen-Generator"""
    a, b, counter = 0, 1, 0
    while True:
        if (counter > n):
            return
        yield a
        a, b = b, a + b
        counter += 1
f = fibonacci(5)
for x in f:
         # no linefeed is enforced by  end="":
    print(x, " ", end="") #
print()
```

```
def fibonacci():
    """Generates an infinite sequence of Fibonacci numbers on demand"""
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

f = fibonacci()

counter = 0
for x in f:
    print(x, " ", end="")
    counter += 1
    if (counter > 10):
        break
print()
```

# Fonctions



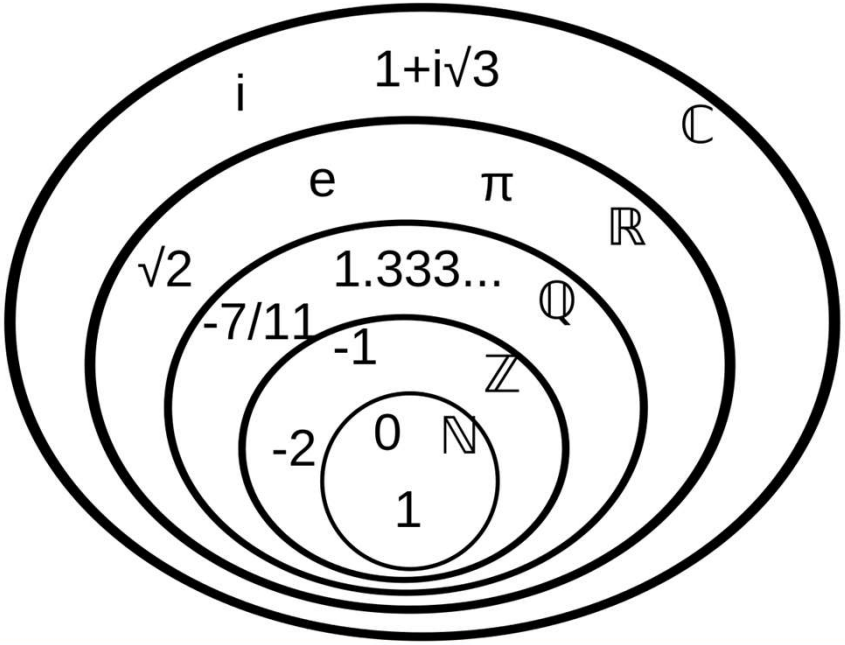
# Fonctions

- Pourquoi une function?
- Définition & Syntaxe
- Appel
- Passage de paramètres: par reference
- Arguments
  - obligatoires,
  - nominatifs (keyworded arguments),
  - par défaut (variable=value),
  - nombre variable d'arguments(*vars, **vars)
- Fonction anonyme (lambda)

# Fonctions (2)

- Valeur de retour
- Scope
- Variable globale vs variable locale

# Numbers



# Nombres

- **Type**: int, long, float, double, complex
- Conversion: int, long, float, complex
- **Fonctions mathématiques** (module math):
  - Abs(), ceil(), exp(), fabs(), floor(), log(), log10(), max(), min(), modf(), pow(), round(), sqrt(), etc

**Fonctions pour nombres aléatoires**:

- choice, randrange(), random(), seed(), shuffle(), uniform(), etc

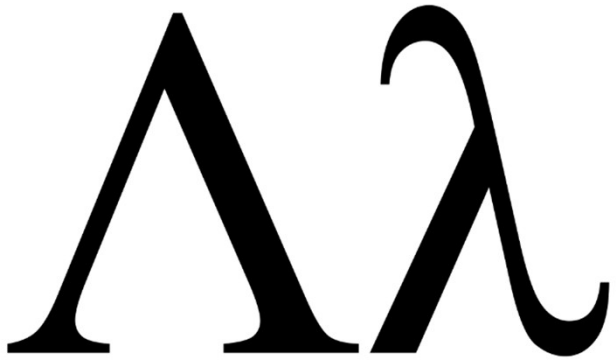
**Fonctions trigonométriques:**

acos(), asin(), atan(), cos(), sin(), etc

# Nombres

- Mathématique:
  - https://docs.python.org/3/library/math.html
  - https://docs.python.org/3/library/cmath.html
- Statistique:
  - https://docs.python.org/3/library/statistics.html
- Aléatoire:
  - https://docs.python.org/3/library/random.html
- Fractions:
  - https://docs.python.org/3/library/fractions.html
- Virgule flottante (décimal):
  - https://docs.python.org/3/library/decimal.html

# Fonctions Lambda



# Fonctions Lambda

- Fonctions lambda = fonctions anonymes
- Elles n'ont pas de nom
- Elles ont 1 seule expression dans le corps
- Exemple: fonction pour calculer le carré:
  - z = lambda x: x * x
- Très utilisées dans la **programmation fonctionnelle**

# Fonctions associées: map et filter

- La fonction map() applique la fonction lambda à chacun des éléments d'une séquence (tableau, liste, ensemble) et retourne une séquence
- Exemple:

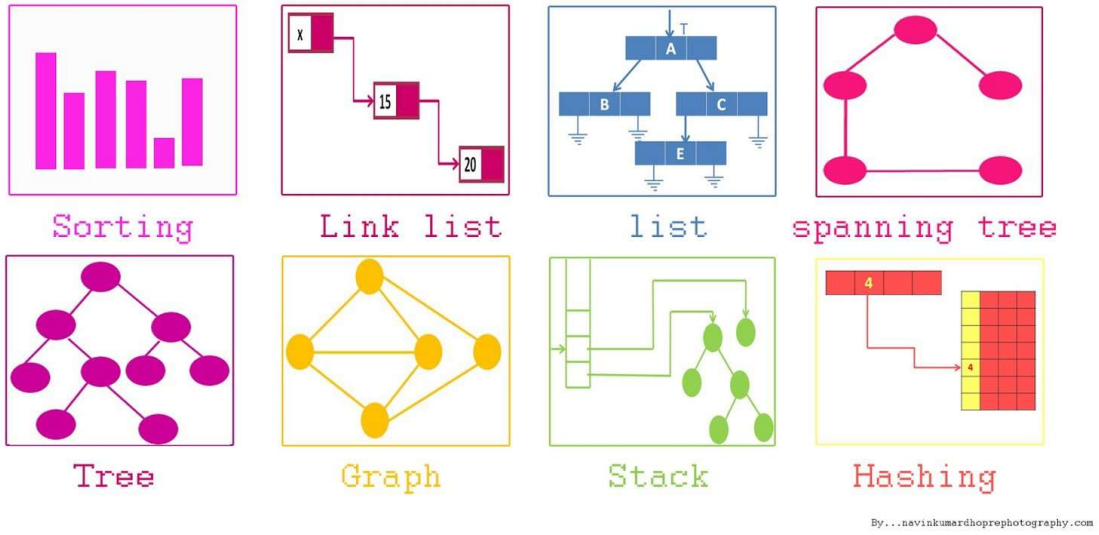
```
nombres=[2, 5, 6, 7, 8, 9, 10, 11, 12, 13]
carres= map(lambda x: x*x, nombres)
for unCarre in carres:
    print(unCarre)
```

# Fonctions associées: map et filter

- La fonction filter() cherche, dans une séquence, que les éléments qui correspondent à la condition vraie (true)
- Exemple: Recherche des nombres pairs

```
nombres=[2, 5, 6, 7, 8, 9, 10, 11, 12, 13]
pairs= filter(lambda x: x%2 == 0, nombres)
for pair in pairs:
    print(pair)
```

# Structures de données



# Structures de données: en général

- Recursion
- Array-Based Sequences
- Stacks, Queue, Deque
- Linked List
- Trees
- Priority Queue
- Maps, Hashtable
- B-Tree (Search Tree)
- Graphs

# Structures de données: Sequences

- **Array-Based Sequences** (ordre des données est important):
  - String (str: ’’, ””, ’’’’’’, ”””””””)
  - Tableau dynamique (list: [])
  - Tuplet (tuple: ())
    - Si le tuple contient un seul élément, une virgule doit être placée après le dernier élément pour dire que le type est tuple (sinon par **exemple t = (17)** serait considéré comme un nombre, d’où (17,) => tuple t=(17,)
- **Collections** (au sens mathématique = ensemble):
  - Ensemble (set = hashtable: {}),
  - Ensemble non modifiable (frozenset = immutable set)
- **Dictionnaires** (dictionary, tableau associatif, mapping)
  - dict({})

# Chaînes de caractères (Strings)

- Déclaration & initialization
- Accès aux caractères
- Caractères d'échappement
- Caractères spéciaux
- Formattage
- Triple quotes
- Fonctions: capitalize(), center(), count(), decode(), encode(), etc.

# Listes

- Déclaration: [], list()
- Accès
- Mise à jour
- Effacer
- Opérations de base:
  - Longueur, concaténation, répétition, appartenance, itération
  - Indexation, sous-liste,
  - Fonctions: cmp(), len(), max(), min(), list()
  - Autres fonctions sur 2 listes: append(), count(), insert(), etc.

# Tuples

- Déclaration: (), tuple()
- Accès: []
- Mise à jour: NON
- Effacer
- Opérations de base:
  - Longueur, concaténation, répétition, appartenance, itération
  - Indexation, sous-ensemble de tuple,
  - Fonctions: cmp(), len(), max(), min(), tuple()

# Dictionnaire

- Déclaration: {}, dict()
- Accès: []
- Mise à jour: NON
- Effacer
- Opérations de base:
  - Longueur, concaténation, répétition, appartenance, itération
  - Indexation, sous-ensemble d'un dictionnaire,
  - Fonctions: cmp(), len(), str(), type()
  - Autres fonctions: clear(), copy(), fromkeys(), get(), etc.

# Opérations sur les Séquences

| Operation | Result | Notes |
|---|---|---|
| `x in s` | `True` if an item of *s* is equal to *x*, else `False` | (1) |
| `x not in s` | `False` if an item of *s* is equal to *x*, else `True` | (1) |
| `s + t` | the concatenation of *s* and *t* | (6) |
| `s * n , n * s` | *n* shallow copies of *s* concatenated | (2) |
| `s[i]` | *i*'th item of *s*, origin 0 | (3) |
| `s[i:j]` | slice of *s* from *i* to *j* | (3), (4) |
| `s[i:j:k]` | slice of *s* from *i* to *j* with step *k* | (3), (5) |
| `len(s)` | length of *s* | |
| `min(s)` | smallest item of *s* | |
| `max(s)` | largest item of *s* | |

# Opérations logiques sur les séquences

| | |
|---|---|
| s == t | equivalent (element by element) |
| s != t | not equivalent |
| s < t | lexicographically less than |
| s <= t | lexicographically less than or equal to |
| s > t | lexicographically greater than |
| s >= t | lexicographically greater than or equal to |

# Opérations sur ensembles et dictionaires

| | |
|---|---|
| key **in** s | containment check |
| key **not in** s | non-containment check |
| s1 == s2 | s1 is equivalent to s2 |
| s1 != s2 | s1 is not equivalent to s2 |
| s1 <= s2 | s1 is subset of s2 |
| s1 < s2 | s1 is proper subset of s2 |
| s1 >= s2 | s1 is superset of s2 |
| s1 > s2 | s1 is proper superset of s2 |
| s1 \| s2 | the union of s1 and s2 |
| s1 & s2 | the intersection of s1 and s2 |
| s1 − s2 | the set of elements in s1 but not s2 |
| s1 ^ s2 | the set of elements in precisely one of s1 or s2 |

# Opérations sur les dictionaires

| | |
|---|---|
| d[key] | value associated with given key |
| d[key] = value | set (or reset) the value associated with given key |
| **del** d[key] | remove key and its associated value from dictionary |
| key **in** d | containment check |
| key **not in** d | non-containment check |
| d1 == d2 | d1 is equivalent to d2 |
| d1 != d2 | d1 is not equivalent to d2 |

# Affections sur les séquences

```
alpha = [1, 2, 3]
beta = alpha                    # an alias for alpha
beta += [4, 5]                  # extends the original list with two more elements
beta = beta + [6, 7]            # reassigns beta to a new list [1, 2, 3, 4, 5, 6, 7]
print(alpha)                    # will be [1, 2, 3, 4, 5]
```

- **Attention**: Pour les types 'non-modifiable'(immutable), une nouvelle variable est créée (nouvel objet) et remplace l'ancienne

# Affectation chaînée (chained assignment)

- **a = b = c = x = y = z = 0**
- **1 <= x + y <= 10** (➔ (1 <= x + y) and (x + y <= 10)

# Ordre de précédence des opérateurs

| Operator Precedence | | |
|---|---|---|
| | **Type** | **Symbols** |
| 1 | member access | expr.member |
| 2 | function/method calls<br>container subscripts/slices | expr(...)<br>expr[...] |
| 3 | exponentiation | ** |
| 4 | unary operators | +expr, −expr, ~expr |
| 5 | multiplication, division | *, /, //, % |
| 6 | addition, subtraction | +, − |
| 7 | bitwise shifting | <<, >> |
| 8 | bitwise-and | & |
| 9 | bitwise-xor | ^ |
| 10 | bitwise-or | \| |
| 11 | comparisons<br>containment | **is**, **is not**, ==, !=, <, <=, >, >=<br>**in**, **not in** |
| 12 | logical-not | **not** expr |
| 13 | logical-and | **and** |
| 14 | logical-or | **or** |
| 15 | conditional | val1 **if** cond **else** val2 |
| 16 | assignments | =, +=, −=, *=, etc. |

# Opérations courantes sur les séquences

| Name | Operator | Example |
|---|---|---|
| indexing | `[n]` | `data = [1,2,3,4,5]`<br>`data[3] # return 4` |
| concatenation | `+` | `data = [1,2,3,4,5]`<br>`data + [9]`<br>`# return [1,2,3,4,5,9]` |
| repetition | `*` | `data = [1,2,3,4,5]`<br>`data * 2`<br>`# return [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]` |
| membership | `in` | `data = [9,2,4,4,6,2,8]`<br>`for val in data: print(val),`<br>`# return 9 2 4 4 6 2 8` |

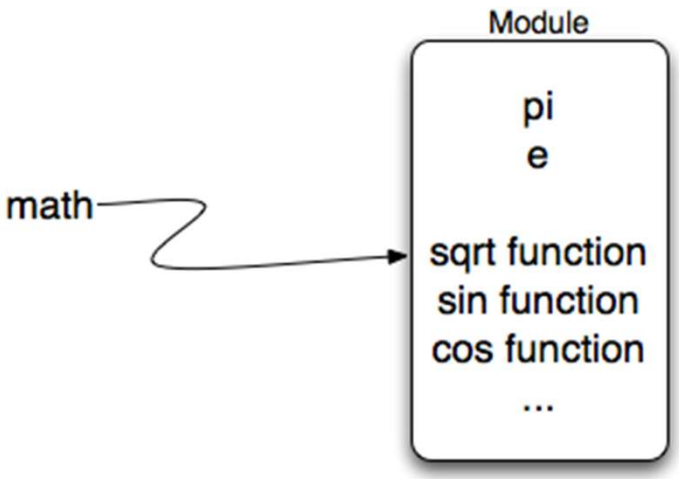
# Opérations courantes sur les séquences

| | | |
|---|---|---|
| length | `len()` | `data = [1,2,3,4,5]`<br>`len(data) # return 5` |
| slicing with step k | `[i:j:k]` | `data = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]`<br>`data[0:10:2] # return [1, 3, 5, 7, 9]` |
| slicing | `[i:j]` | `data = [1,2,3,4,5]`<br>`data[1:3] # return [2,3]` |
| Minimum | `min(data)` | `data = [1,2,3,4,5]`<br>`min(data) # return 1` |

# Opérations courantes sur les séquences

| | | |
|---|---|---|
| Index | `data.index(sub[, start[, end]])` | `data = [1,2,3,4,5]`<br>`data.index(3) # return 2` |
| Count | `data.count(i)` | `data = [1,2,3,4,5,3]`<br>`data.count(3) # return 2` |

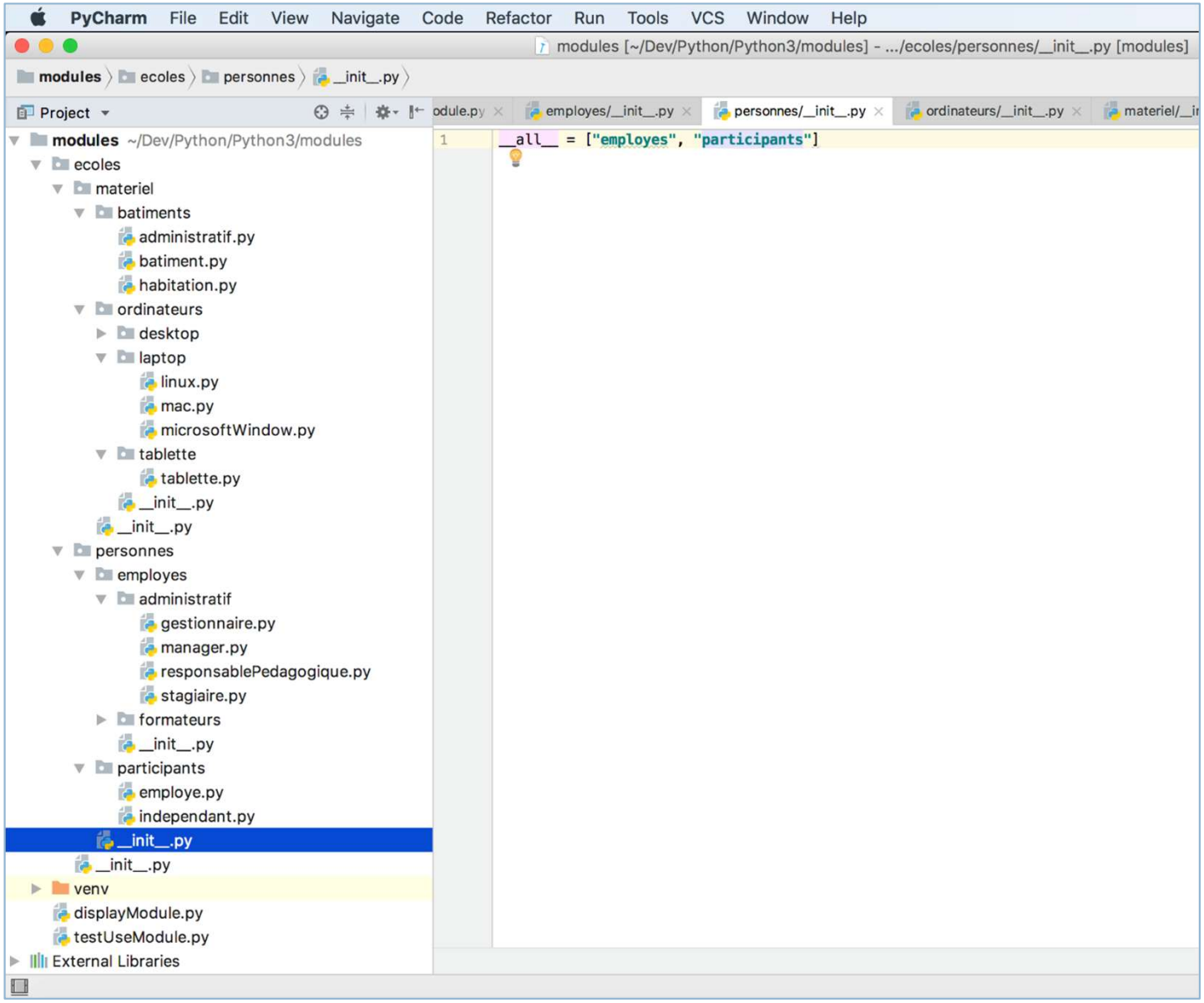
# Modules



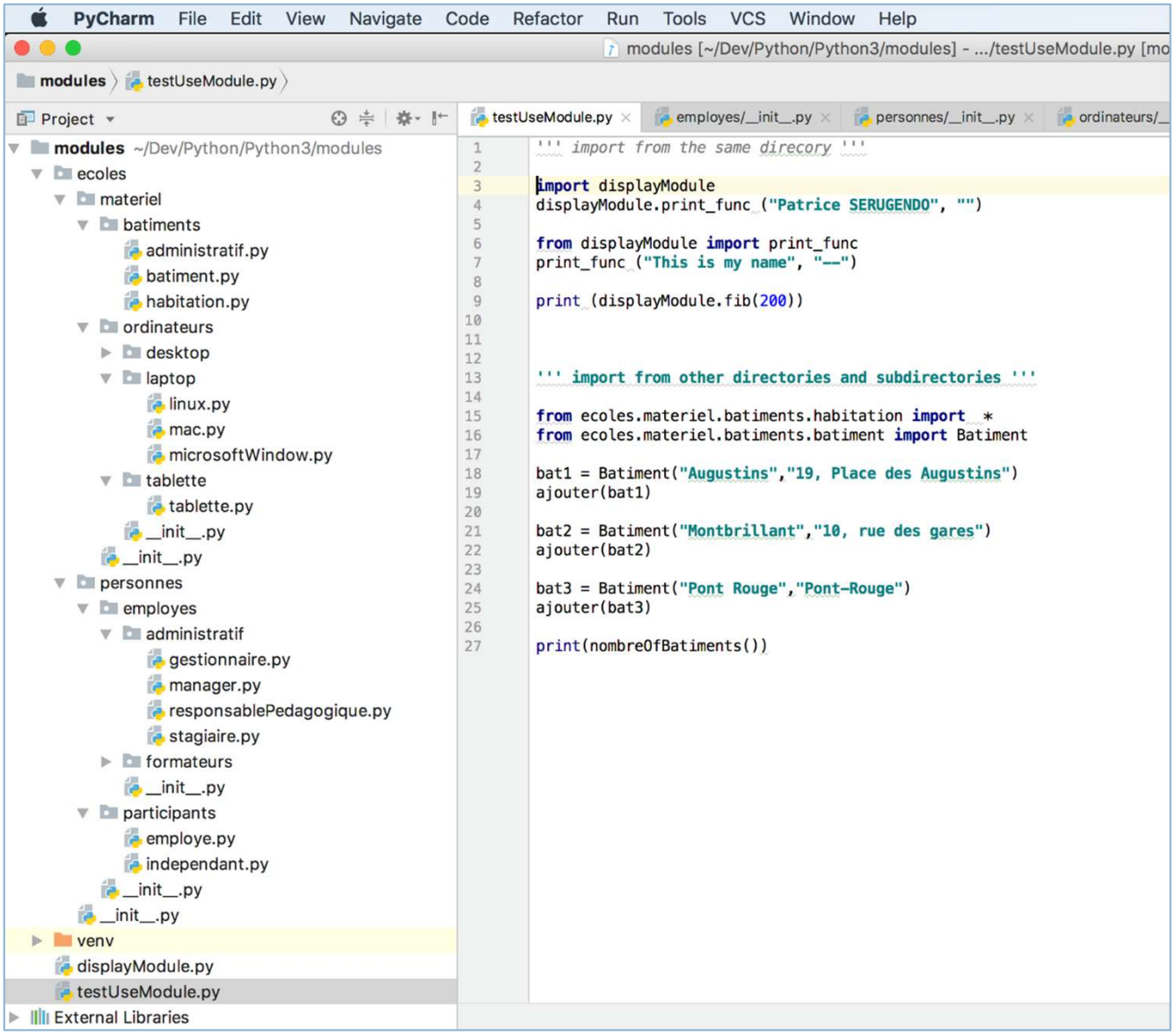
# Module

- import ….
- from …. import …
- **The *from…import* * Statement**
- Exécuter les modules comme des scripts
- Localisation des modules
- Variable d'environnement PYTHONPATH
- Namespace & scoping
- Fonctions: dir(), globals(), locals(), reload()
- Package

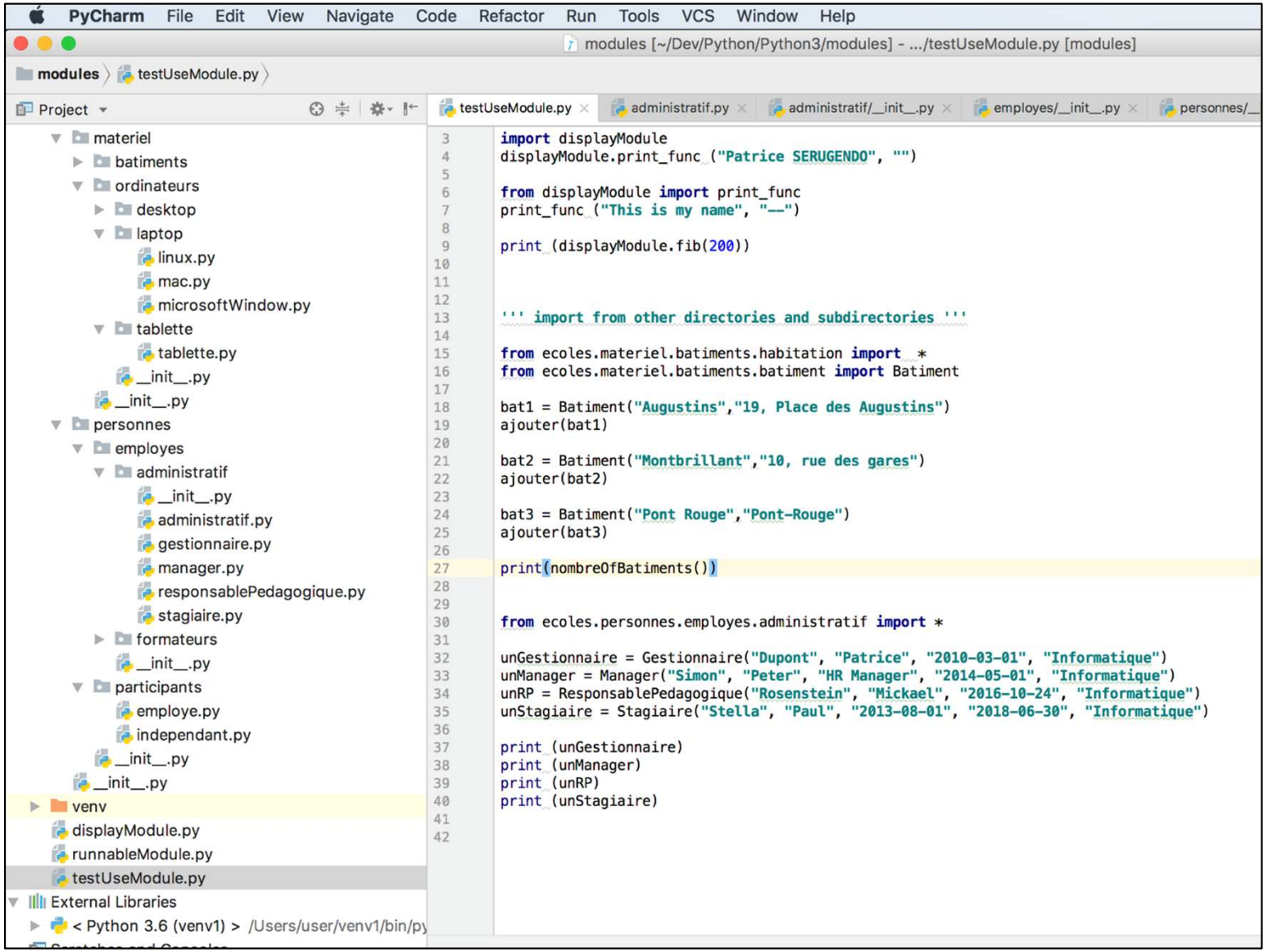
# Structure


PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help
modules [~/Dev/Python/Python3/modules] - .../ecoles/personnes/__init__.py [modules]
modules ecoles personnes __init__.py
Project
employes/__init__.py
personnes/__init__.py
ordinateurs/__init__.py
1 __all__ = ["employes", "participants"]
modules ~/Dev/Python/Python3/modules
ecoles
materiel
batiments
administratif.py
batiment.py
habitation.py
ordinateurs
desktop
laptop
linux.py
mac.py
microsoftWindow.py
tablette
tablette.py
__init__.py
__init__.py
personnes
employes
administratif
gestionnaire.py
manager.py
responsablePedagogique.py
stagiaire.py
formateurs
__init__.py
participants
employe.py
independant.py
__init__.py
__init__.py
venv
displayModule.py
testUseModule.py
External Libraries

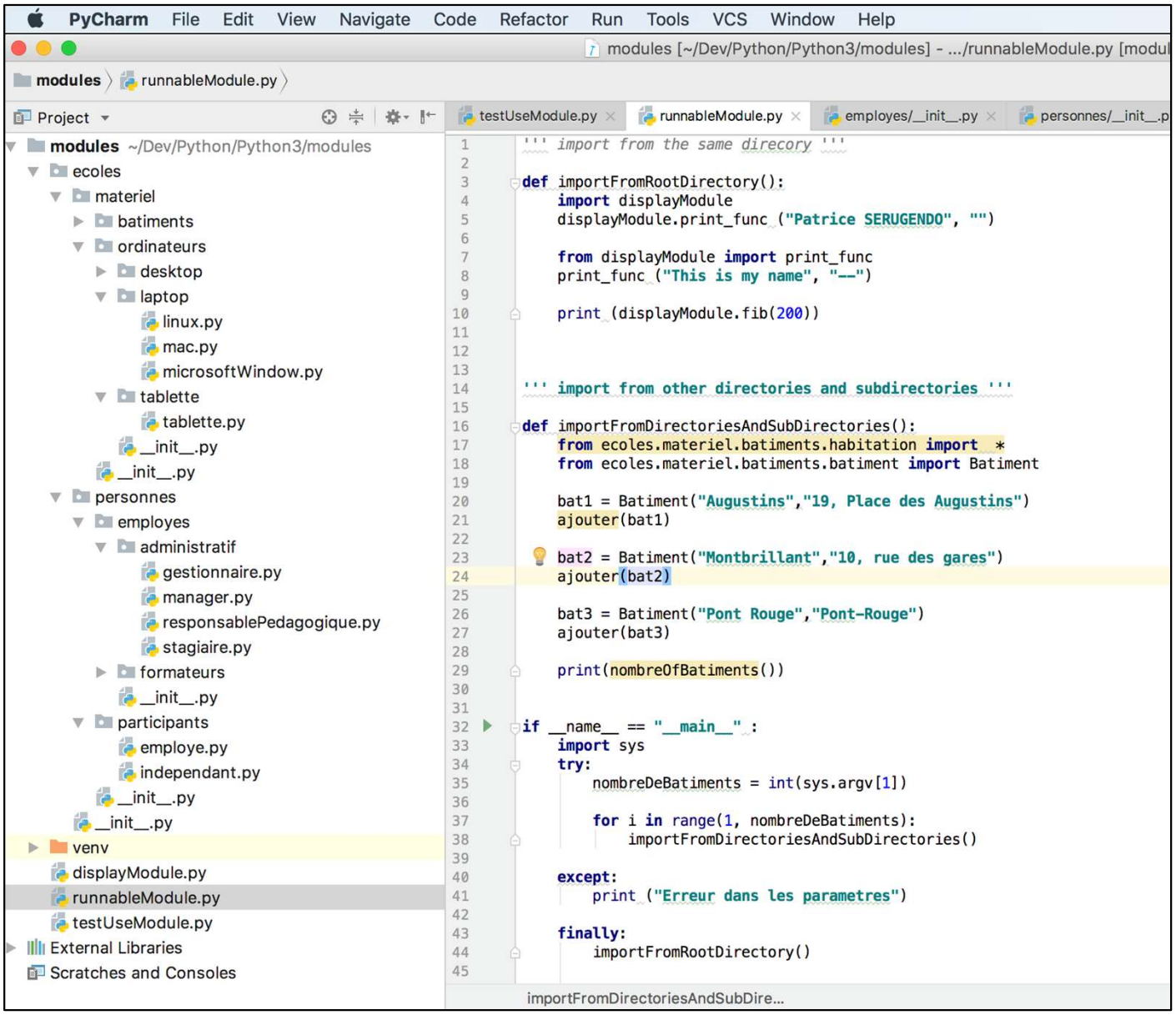
# Import from


PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help
modules [~/Dev/Python/Python3/modules] - .../testUseModule.py [mo
modules testUseModule.py
Project
testUseModule.py
employes/__init__.py
personnes/__init__.py
ordinateurs/_
modules ~/Dev/Python/Python3/modules
ecoles
materiel
batiments
administratif.py
batiment.py
habitation.py
ordinateurs
desktop
laptop
linux.py
mac.py
microsoftWindow.py
tablette
tablette.py
__init__.py
__init__.py
personnes
employes
administratif
gestionnaire.py
manager.py
responsablePedagogique.py
stagiaire.py
formateurs
__init__.py
participants
employe.py
independant.py
__init__.py
__init__.py
venv
displayModule.py
testUseModule.py
External Libraries
1 ''' import from the same direcory '''
2
3 import displayModule
4 displayModule.print_func ("Patrice SERUGENDO", "")
5
6 from displayModule import print_func
7 print_func ("This is my name", "--")
8
9 print (displayModule.fib(200))
10
11
12
13 ''' import from other directories and subdirectories '''
14
15 from ecoles.materiel.batiments.habitation import *
16 from ecoles.materiel.batiments.batiment import Batiment
17
18 bat1 = Batiment("Augustins","19, Place des Augustins")
19 ajouter(bat1)
20
21 bat2 = Batiment("Montbrillant","10, rue des gares")
22 ajouter(bat2)
23
24 bat3 = Batiment("Pont Rouge","Pont-Rouge")
25 ajouter(bat3)
26
27 print(nombreOfBatiments())

# Example


PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help
modules [~/Dev/Python/Python3/modules] - .../testUseModule.py [modules]
modules testUseModule.py
Project
testUseModule.py
administratif.py
administratif/__init__.py
employes/__init__.py
personnes/__
materiel
batiments
ordinateurs
desktop
laptop
linux.py
mac.py
microsoftWindow.py
tablette
tablette.py
__init__.py
__init__.py
personnes
employes
administratif
__init__.py
administratif.py
gestionnaire.py
manager.py
responsablePedagogique.py
stagiaire.py
formateurs
__init__.py
participants
employe.py
independant.py
__init__.py
__init__.py
venv
displayModule.py
runnableModule.py
testUseModule.py
External Libraries
< Python 3.6 (venv1) > /Users/user/venv1/bin/py
3 import displayModule
4 displayModule.print_func ("Patrice SERUGENDO", "")
5
6 from displayModule import print_func
7 print_func ("This is my name", "--")
8
9 print (displayModule.fib(200))
10
11
12
13 ''' import from other directories and subdirectories '''
14
15 from ecoles.materiel.batiments.habitation import *
16 from ecoles.materiel.batiments.batiment import Batiment
17
18 bat1 = Batiment("Augustins","19, Place des Augustins")
19 ajouter(bat1)
20
21 bat2 = Batiment("Montbrillant","10, rue des gares")
22 ajouter(bat2)
23
24 bat3 = Batiment("Pont Rouge","Pont-Rouge")
25 ajouter(bat3)
26
27 print(nombreOfBatiments())
28
29
30 from ecoles.personnes.employes.administratif import *
31
32 unGestionnaire = Gestionnaire("Dupont", "Patrice", "2010-03-01", "Informatique")
33 unManager = Manager("Simon", "Peter", "HR Manager", "2014-05-01", "Informatique")
34 unRP = ResponsablePedagogique("Rosenstein", "Mickael", "2016-10-24", "Informatique")
35 unStagiaire = Stagiaire("Stella", "Paul", "2013-08-01", "2018-06-30", "Informatique")
36
37 print (unGestionnaire)
38 print (unManager)
39 print (unRP)
40 print (unStagiaire)
41
42

# Scripts exécutables



# Terminologie

- A **Python module** is simply a **Python** source file, which can expose classes, functions and global variables. When imported from another **Python** source file, the file name is treated as a namespace. A **Python package** is simply a directory of **Python module**(s).

# Fichiers

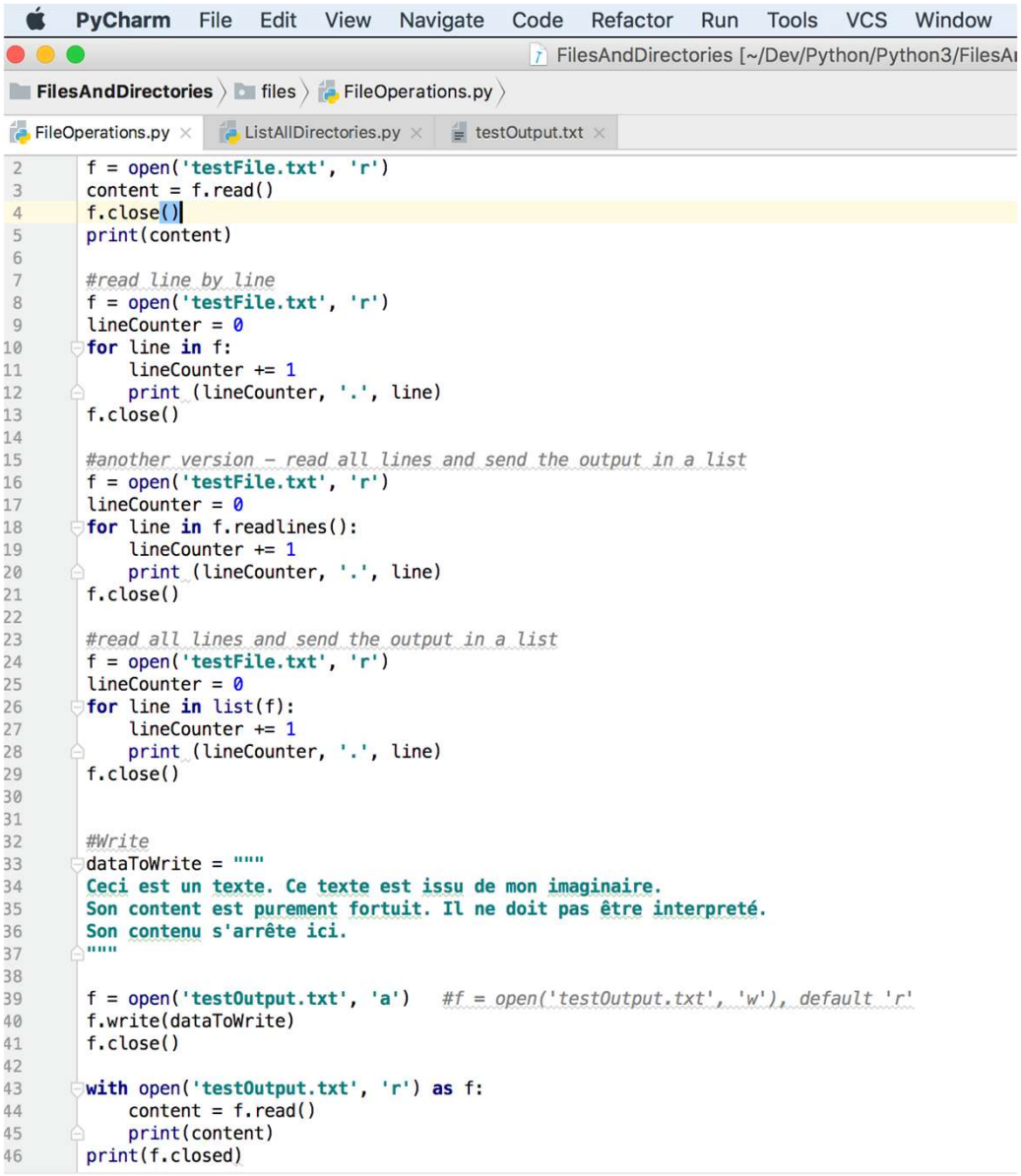
# Fichiers – lecture/écriture

- Imprimer vs afficher
- Imprimer à l'écran : print()
- Entrée des données par le clavier: input()
- Ouverture et fermeture d'un fichier
  - variableFichier = open(nomFichier[,mode d'accès][, taille buffer])
  - Propriétés de variableFichier: closed, mode, name
  - Méthodes de variableFichier: close(), write(), read(), tell(), seek(), etc

# Fichiers – imprimer et afficher

```
Old: print "The answer is", 2*2
New: print("The answer is", 2*2)

Old: print x,              # Trailing comma suppresses newline
New: print(x, end=" ")     # Appends a space instead of a newline

Old: print                 # Prints a newline
New: print()               # You must call the function!

Old: print >>sys.stderr, "fatal error"
New: print("fatal error", file=sys.stderr)

Old: print (x, y)          # prints repr((x, y))
New: print((x, y))         # Not the same as print(x, y)!
```

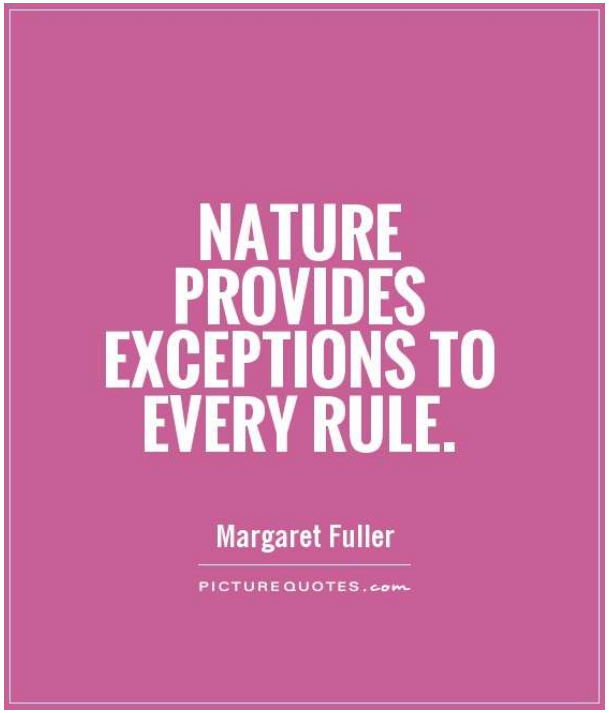
# Fichiers – ouverture et fermeture



# Fichiers : manipulation

- Utiliser le module os
- Méthodes du module os:
  - os.rename()
  - os.remove()
  - os.makedir()
  - os.chdir()
  - os.getcwd()
  - os.rmdir()
  - etc

# Exceptions


NATURE PROVIDES EXCEPTIONS TO EVERY RULE.
Margaret Fuller
PICTUREQUOTES.com

# Gestion des exceptions

- Deux manières de gérer les exceptions:
  - Exceptions

try

raise exception

except

finally

```
try:
   You do your operations here
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

# Exceptions définies par l'utilisateur

```
class Networkerror(RuntimeError):
   def __init__(self, arg):
      self.args = arg
```

```
try:
   raise Networkerror("Bad hostname")
except Networkerror,e:
   print e.args
```

# Gestion des exceptions

- Assertions
  - Les assertions servent essentiellement à tester les valeurs des paramètres pour s'assurer de leur validité avant de les utiliser.

```
def KelvinToFahrenheit(Temperature):
    assert (Temperature >= 0),"Colder than absolute zero!"
    return ((Temperature-273)*1.8)+32

print (KelvinToFahrenheit(273))
print (int(KelvinToFahrenheit(505.78)))
print (KelvinToFahrenheit(-5))
```

# Fonctions intégrées (Built-in)

# Fonctions intégrées (built-in)

| | | Built-in Functions | | |
|---|---|---|---|---|
| abs() | dict() | help() | min() | setattr() |
| all() | dir() | hex() | next() | slice() |
| any() | divmod() | id() | object() | sorted() |
| ascii() | enumerate() | input() | oct() | staticmethod() |
| bin() | eval() | int() | open() | str() |
| bool() | exec() | isinstance() | ord() | sum() |
| bytearray() | filter() | issubclass() | pow() | super() |
| bytes() | float() | iter() | print() | tuple() |
| callable() | format() | len() | property() | type() |
| chr() | frozenset() | list() | range() | vars() |
| classmethod() | getattr() | locals() | repr() | zip() |
| compile() | globals() | map() | reversed() | __import__() |
| complex() | hasattr() | max() | round() | |
| delattr() | hash() | memoryview() | set() | |

# Fonctions intégrées (suite)

| Fonction | Explication/Description |
|---|---|
| **abs()** | Return the absolute value of a number. |
| **all()** | Return True if all elements of the iterable are true (or if the iterable is empty). |
| **any()** | Return True if any element of the iterable is true. If the iterable is empty, return False. |
| **ascii()** | Return a string containing a printable representation of an object, but escape the non-ASCII characters. |
| **bin()** | Convert an integer number to a binary string. |
| **bool()** | Convert a value to a Boolean. |
| **bytearray()** | Return a new array of bytes. |
| **bytes()** | Return a new "bytes" object. |
| **callable()** | Return True if the object argument appears callable, False if not. |
| **chr()** | Return the string representing a character. |
| **classmethod()** | Return a class method for the function. |
| **compile()** | Compile the source into a code or AST object. |
| **complex()** | Create a complex number or convert a string or number to a complex number. |
| **delattr()** | Deletes the named attribute of an object. |

# Fonctions intégrées (suite 2)

| | |
|---|---|
| **dict()** | Create a new dictionary. |
| **dir()** | Return the list of names in the current local scope. |
| **divmod()** | Return a pair of numbers consisting of quotient and remainder when using integer division. |
| **enumerate()** | Return an enumerate object. |
| **eval()** | The argument is parsed and evaluated as a Python expression. |
| **exec()** | Dynamic execution of Python code. |
| **filter()** | Construct an iterator from elements of iterable for which function returns true. |
| **float()** | Convert a string or a number to floating point. |
| **format()** | Convert a value to a "formatted" representation. |
| **frozenset()** | Return a new frozenset object. |
| **getattr()** | Return the value of the named attribute of an object. |
| **globals()** | Return a dictionary representing the current global symbol table. |
| **hasattr()** | Return True if the name is one of the object's attributes. |
| **hash()** | Return the hash value of the object. |

# Fonctions intégrées (suite 3)

| | |
|---|---|
| **help()** | Invoke the built-in help system. |
| **hex()** | Convert an integer number to a hexadecimal string. |
| **id()** | Return the "identity" of an object. |
| **input()** | Reads a line from input, converts it to a string (stripping a trailing newline), and returns that. |
| **int()** | Convert a number or string to an integer. |
| **isinstance()** | Return True if the object argument is an instance. |
| **issubclass()** | Return True if class is a subclass. |
| **iter()** | Return an iterator object. |
| **len()** | Return the length (the number of items) of an object. |
| **list()** | Return a list. |
| **locals()** | Update and return a dictionary representing the current local symbol table. |
| **map()** | Return an iterator that applies function to every item of iterable, yielding the results. |
| **max()** | Return the largest item in an iterable. |
| **memoryview()** | Return a "memory view" object created from the given argument. |
| **min()** | Return the smallest item in an iterable. |
| **next()** | Retrieve the next item from the iterator. |

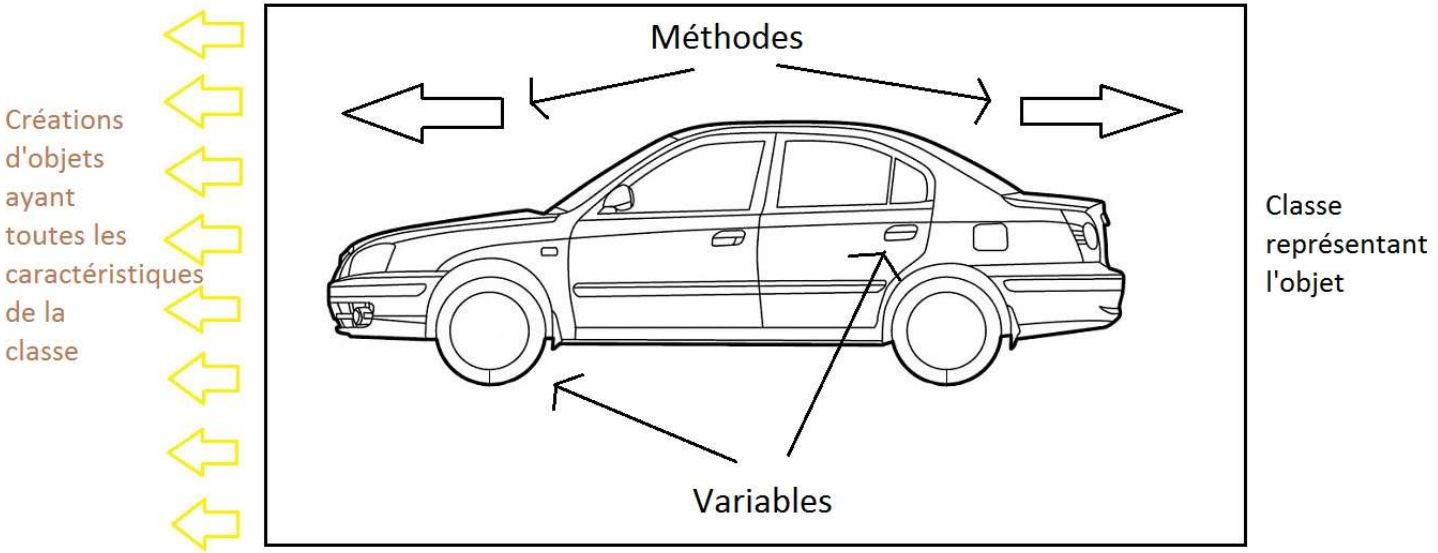
# Fonctions intégrées (suite 4)

| | |
|---|---|
| **object()** | Return a new featureless object. |
| **oct()** | Convert an integer number to an octal string. |
| **open()** | Open file and return a corresponding file object. |
| **ord()** | Return an integer representing the Unicode. |
| **pow()** | Return power raised to a number. |
| **print()** | Print objects to the stream. |
| **property()** | Return a property attribute. |
| **range()** | Return an iterable sequence. |
| **repr()** | Return a string containing a printable representation of an object. |
| **reversed()** | Return a reverse iterator. |
| **round()** | Return the rounded floating point value. |
| **set()** | Return a new set object. |
| **setattr()** | Assigns the value to the attribute. |
| **slice()** | Return a slice object. |
| **sorted()** | Return a new sorted list. |
| **staticmethod()** | Return a static method for function. |
| **str()** | Return a str version of object. |
| **sum()** | Sums the items of an iterable from left to right and returns the total. |

# Fonctions intégrées (suite 5)

| | |
|---|---|
| **super()** | Return a proxy object that delegates method calls to a parent or sibling class. |
| **tuple()** | Return a tuple |
| **type()** | Return the type of an object. |
| **vars()** | Return the __dict__ attribute for a module, class, instance, or any other object. |
| **zip()** | Make an iterator that aggregates elements from each of the iterables. |
| **__import__()** | This function is invoked by the import statement. |

# Programmation Orientée-Objet



# Programmation Orientée-Objet

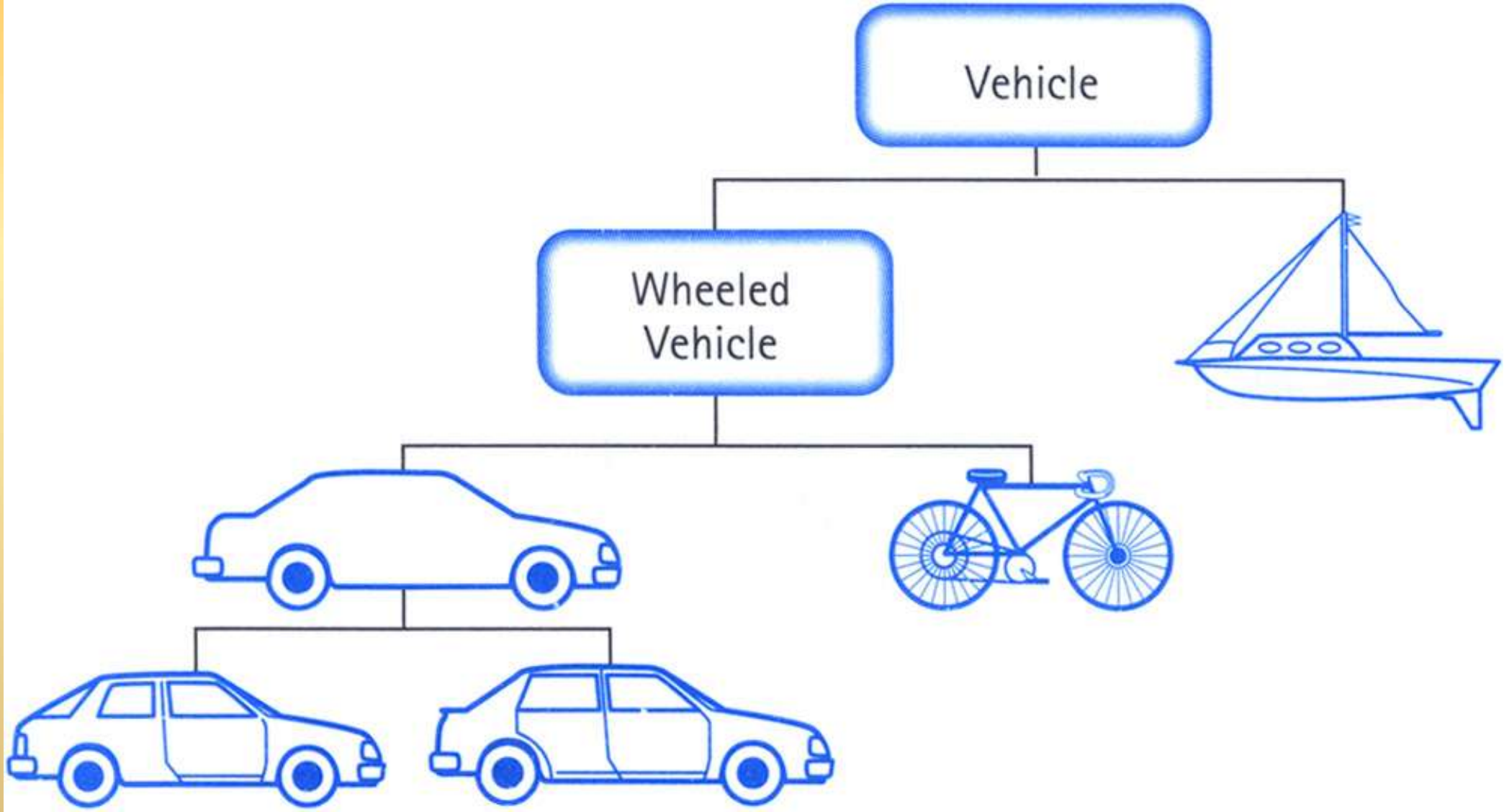
- Terminologie
  - Classe
  - Object
  - Instance
  - Méthode
  - Attribut/Propriété
  - Super classe  vs Sous classe
  - Généralisation vs spécialisation
  - Public vs Privé
  - Abstraction, encapsulation, heritage, polymorphisme

# Encapsulation

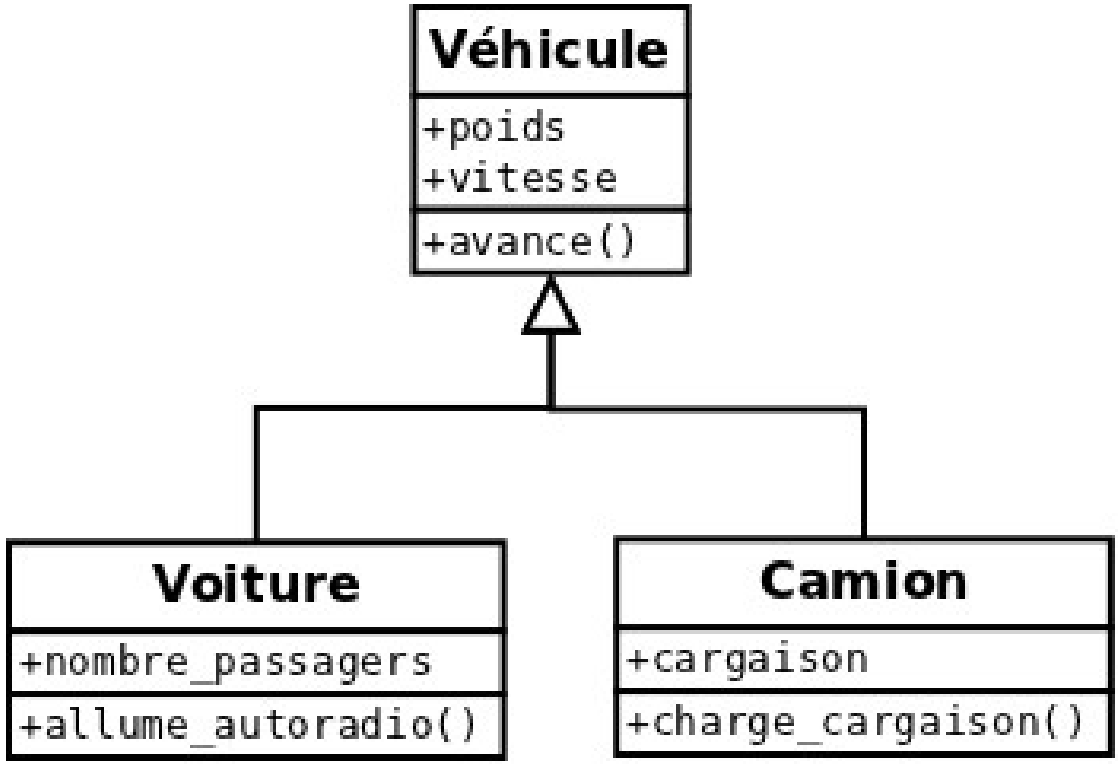
- Une méthode peut être publique ou privée
- Une propriété peut être publique ou privée

```
#!/usr/bin/env python

class Car:

    __maxspeed = 0
    __name = ""

    def __init__(self):
        self.__maxspeed = 200
        self.__name = "Supercar"

    def drive(self):
        print 'driving. maxspeed ' + str(self.__maxspeed)

redcar = Car()
redcar.drive()
redcar.__maxspeed = 10  # will not change variable because its private
redcar.drive()
```

# Héritage



# POO - Héritage



# Héritage

```
class Bear(object):
    def sound(self):
        print "Groarrr"

class Dog(object):
    def sound(self):
        print "Woof woof!"

def makeSound(animalType):
    animalType.sound()


bearObj = Bear()
dogObj = Dog()

makeSound(bearObj)
makeSound(dogObj)
```

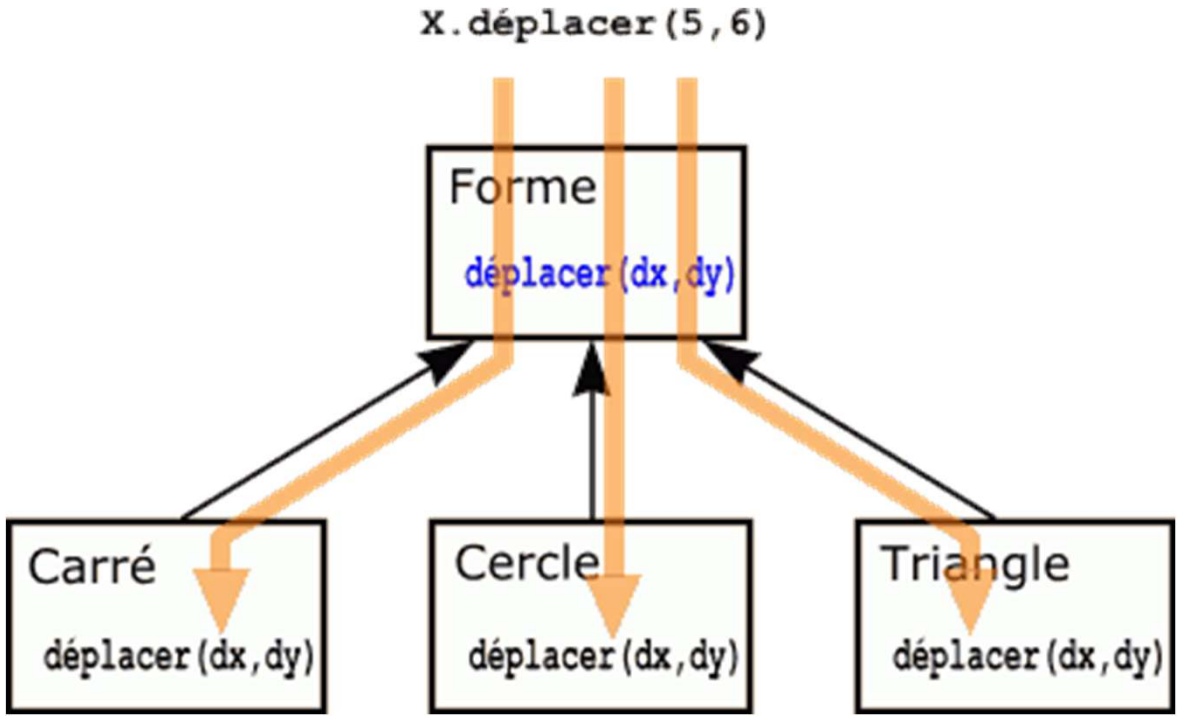
Output:

```
Groarrr
Woof woof!
```

# Héritage multiple

```
class MySuperClass1():

    def method_super1(self):
        print("method_super1 method called")

class MySuperClass2():

    def method_super2(self):
        print("method_super2 method called")

class ChildClass(MySuperClass1, MySuperClass2):

    def child_method(self):
        print("child method")

c = ChildClass()
c.method_super1()
c.method_super2()
```

# POO - Polymorphisme



# Polymorphisme

```
class Document:
    def __init__(self, name):
        self.name = name

    def show(self):
        raise NotImplementedError("Subclass must implement abstract method")

class Pdf(Document):
    def show(self):
        return 'Show pdf contents!'

class Word(Document):
    def show(self):
        return 'Show word contents!'

documents = [Pdf('Document1'),
             Pdf('Document2'),
             Word('Document3')]

for document in documents:
    print document.name + ': ' + document.show()
```

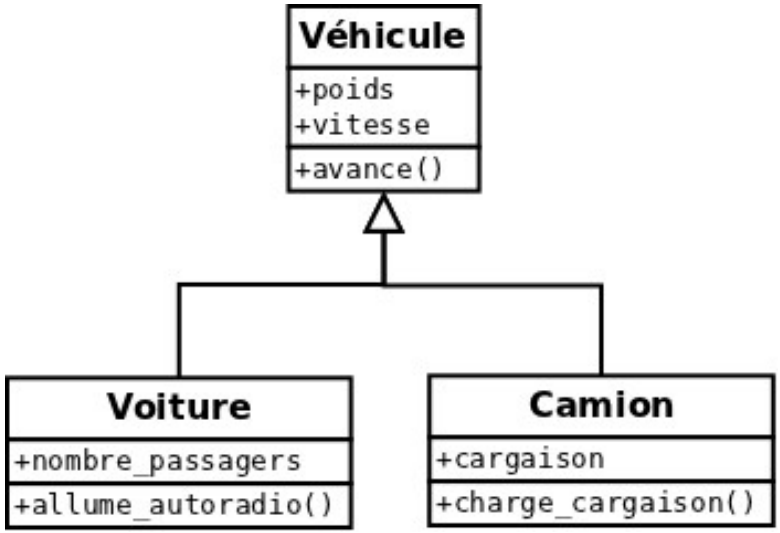
Output:

```
Document1: Show pdf contents!
Document2: Show pdf contents!
Document3: Show word contents!
```

# Polymorphisme (exemple 2)

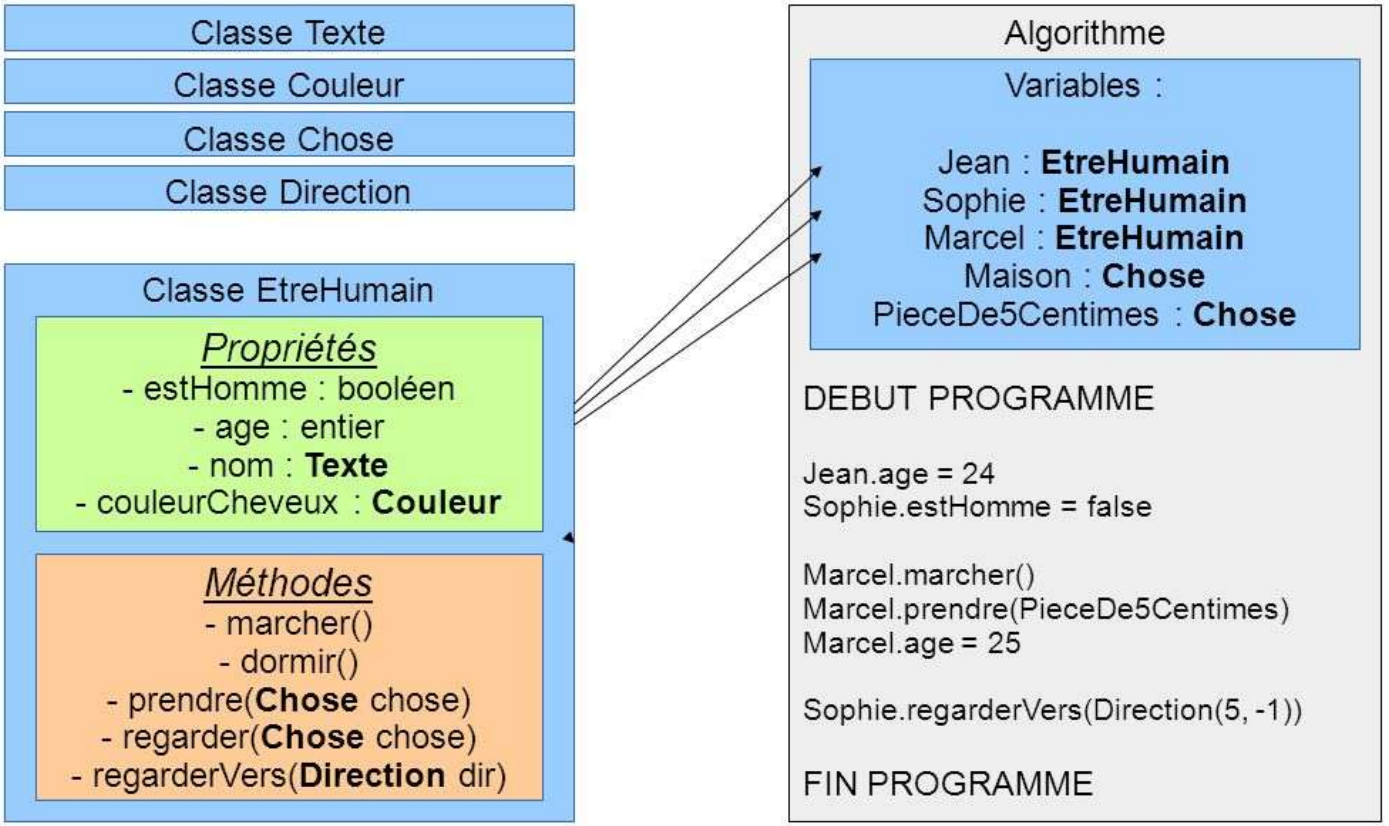
```
class Car:
    def __init__(self, name):
        self.name = name

    def drive(self):
        raise NotImplementedError("Subclass must implement abstract method")

    def stop(self):
        raise NotImplementedError("Subclass must implement abstract method")

class Sportscar(Car):
    def drive(self):
        return 'Sportscar driving!'

    def stop(self):
        return 'Sportscar breaking!'

class Truck(Car):
    def drive(self):
        return 'Truck driving slowly because heavily loaded.'

    def stop(self):
        return 'Truck breaking!'


cars = [Truck('Bananatruck'),
        Truck('Orangetruck'),
        Sportscar('Z3')]

for car in cars:
    print car.name + ': ' + car.drive()
```

# Classe


Véhicule
+poids
+vitesse
+avance()
Voiture
+nombre_passagers
+allume_autoradio()
Camion
+cargaison
+charge_cargaison()

# Instance

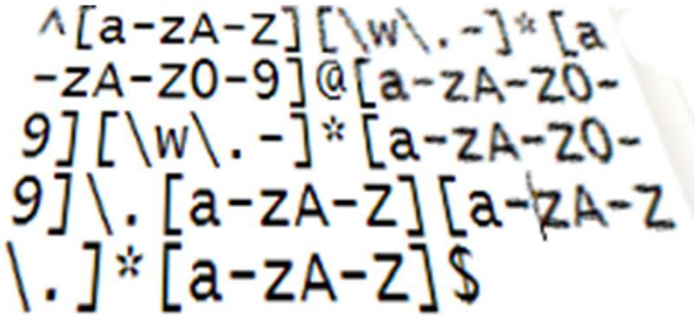
## Classes & Instances



# Classes intégrées (Built-in classes)

| Class | Description | Immutable? |
|---|---|---|
| **bool** | Boolean value | ✓ |
| **int** | integer (arbitrary magnitude) | ✓ |
| **float** | floating-point number | ✓ |
| **list** | mutable sequence of objects | |
| **tuple** | immutable sequence of objects | ✓ |
| **str** | character string | ✓ |
| **set** | unordered set of distinct objects | |
| **frozenset** | immutable form of set class | ✓ |
| **dict** | associative mapping (aka dictionary) | |

# Expressions Régulières (RegEx)



# Expression Régulières (RegEx) – (from wikipedia)

- A **regularexpression**, **regex** or **regexp**[1] (sometimes called a **rational expression**)[2][3] is, in theoretical computer science and formal language theory, a sequence of characters that define a *search pattern*.
- Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings.
- The concept arose in the 1950s when the American mathematician Stephen Cole Kleene formalized the description of a *regular language*.

# Expression Régulières (RegEx) – (from wikipedia)

- The concept came into common use with Unix text-processing utilities.
- Since the 1980s, different syntaxes for writing regular expressions exist, one being the POSIX standard and another, widely used, being the Perl syntax.
- Regular expressions are used in search engines, search and replace dialogs of word processors and text editors, in text processing utilities such as sedand AWK and in lexical analysis.
- Many programming languages provide regex capabilities, built-in or via libraries.

# RegEx : symboles

- . ^ $ * + ? {} [] \ | ()

| | |
|---|---|
| **.** | Le point correspond à n'importe quel caractère. |
| **^** | Indique un commencement de segment mais signifie aussi "contraire de" |
| **$** | Fin de segment |
| **[xy]** | Une liste de segment possibble. Exemple [abc] équivaut à : a, b ou c |
| **(x\|y)** | Indique un choix multiple type (nord\|sud) équivaut à "nord" OU "sud" |
| **\d** | le segment est composé uniquement de chiffre, ce qui équivaut à [0-9] |
| **\D** | le segment n'est pas composé de chiffre, ce qui équivaut à [^0-9] |
| **\s** | Un espace, ce qui équivaut à [ \t\n\r\f\v] |
| **\S** | Pas d'espace, ce qui équivaut à [^ \t\n\r\f\v] |
| **\w** | Présence alphanumérique, ce qui équivaut à [a-zA-Z0-9_] |
| **\W** | Pas de présence alphanumérique [^a-zA-Z0-9_] |
| **** | Est un caractère d'échappement |

# RegEx: Règles

- Il est possible d'imposer le nombre d'occurences avec la syntaxe suivante:
  - **A{2}** : on s'attend à ce que la lettre A (en majuscule) se répète 2 fois consécutives.
  - **BA{1,9}:** on s'attend à ce que le segment BA se répete de 1 a`9 fois consécutives.
  - **BRA{,10}** : on s'attend à ce que le segment BRA ne soit pas présent du tout ou présent jusqu'à 10 fois consécutives.
  - **VO{1,}** : on s'attend à ce que le segment VO soit present au mois une fois.

# RegEx: Règles (exemple)

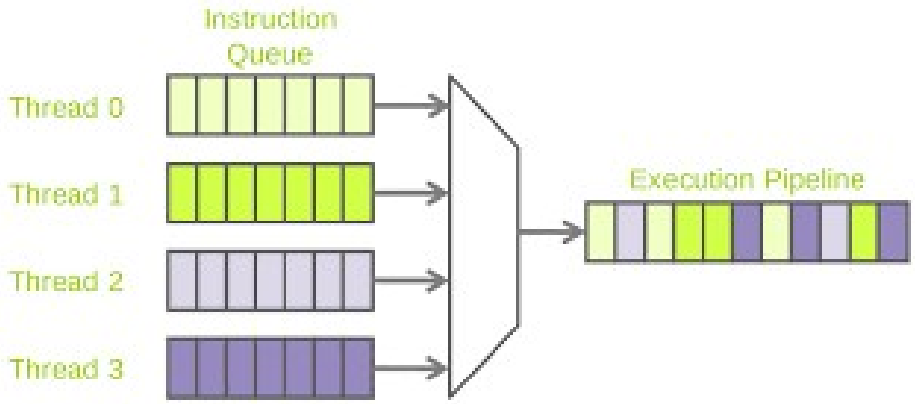
| Symbole | Nb Caractères attendus | Exemple | Cas possibles |
|---|---|---|---|
| ? | 0 ou 1 | GR**(.)?**S | GRS, GR**O**S, GR**I**S, GR**A**S |
| + | 1 ou plus | GR**(.)+**S | GR**O**S, GR**I**S, GR**A**S |
| * | 0, 1 ou plus | GR**(.)***S | GRS,GRO**O**,GR**III**S,GR**Olivier**S |

# Exemple: Contrôle validité d'un format d'adresse email

EmailValidator.py

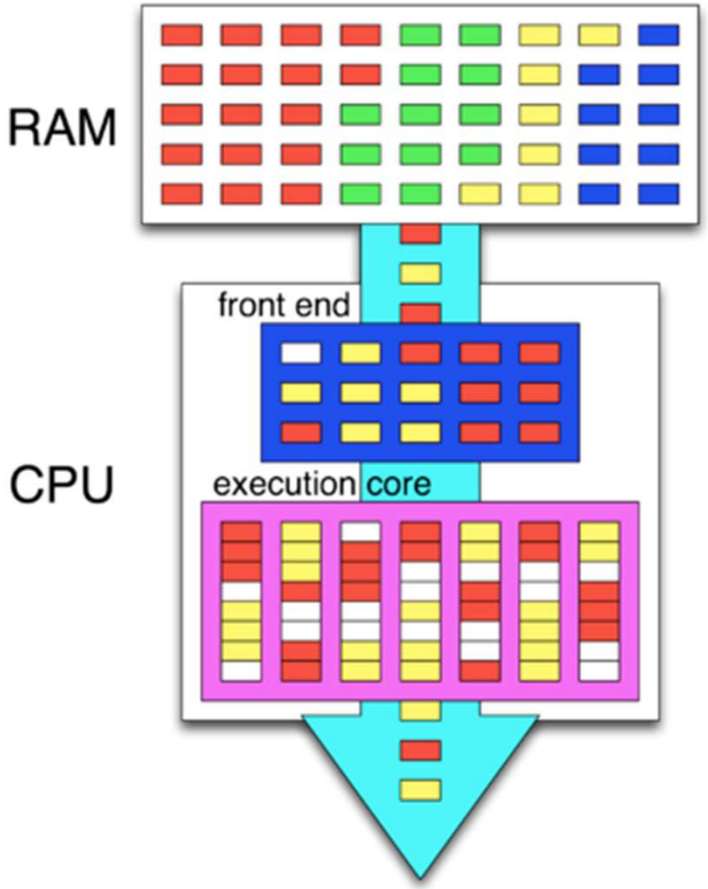
```
1  import re
2
3  mails = ["olivier@mailbidon.com", "olivier@mailbidon.ca", "8@mailbidon.com", "@mailbidon.com", "olivier@mailbidon"]
4  expression = r"^[a-z0-9._-]+@[a-z0-9._-]+\.[(com|ch|net|org|fr|de|it)]+"
5
6  regex = re.compile(expression)
7
8  for mail in mails:
9      if regex.match(mail) is not None:
10         print ("Ce mail : %s est valide" % mail)
11     else:
12         print ("Erreur ce mail : %s est non valide" % mail)
13
14 # print(str(regex.search(mails, expression).groups()))
15
16
```

# Programmation Concurrente



# Programmation concurrente

- Pourquoi la programmation concurrente ?
  - Pour exécuter plusieurs exécutions (precessus) en parallèle (multithreading)
  - Gain de temps



# Exemple sans multithreading

PingWithoutMultithreading.py | PingWithMultithreading.py | PingWithMultithreading2.py

```
#Without multithreading

import os
import subprocess
import re

# on prépare la regex
regex = re.compile(r"(?P<received>\d+) received")
# la fonction qui assure le ping

def ping(hostname):
    p = subprocess.Popen(["ping", "-c1", "-w100", hostname], stdout=subprocess.PIPE).stdout.read()
    r = regex.search(p.decode())

    try:
        if(r.group("received") == "1"):
            print("L'adresse %s existe!" % hostname)
    except:
        pass

    # on boucle sur les adresses du réseau local
    for i in range(254):
        hostname = "192.168.0.%i" % (i+1)
        ping(hostname)
```

# Exemple avec multithreading (1)

PingWithoutMultithreading.py | PingWithMultithreading.py | PingWithMultithreading2.py

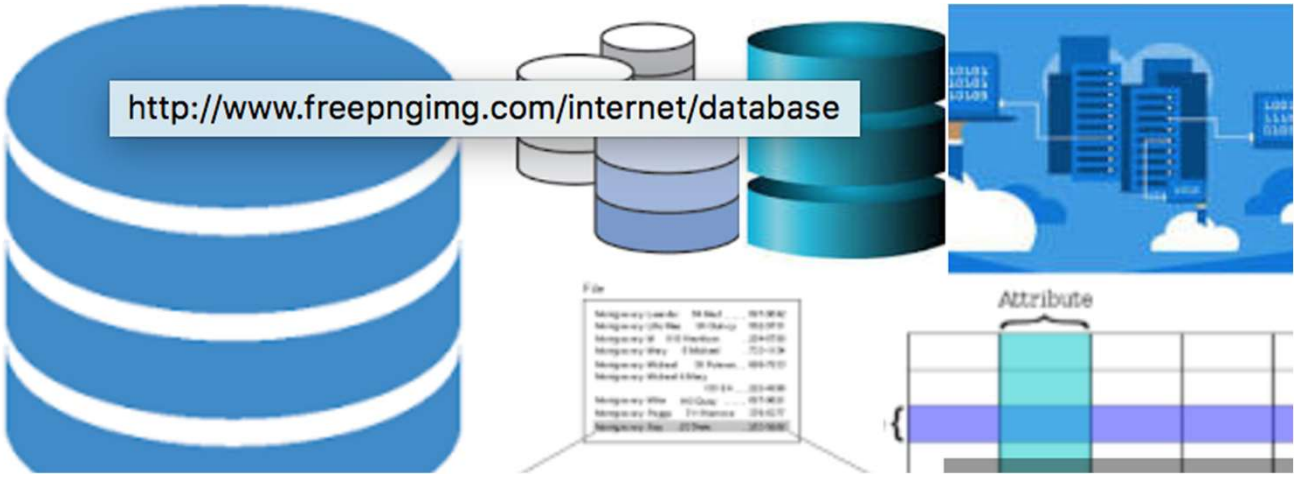
```python
import os
import subprocess
import re
import threading

# on prépare la regex
regex = re.compile(r"(?P<received>\d+) received")
# la fonction qui assure le ping

def ping(hostname):
    p = subprocess.Popen(["ping", "-c1", "-w100", hostname], stdout=subprocess.PIPE).stdout.read()
    r = regex.search(p.decode())
    try:
        if(r.group("received") == "1"):
            print("L'adresse %s existe!" % hostname)
    except:
        pass

# on boucle sur les adresses du réseau local
for i in range(254):
    hostname = "192.168.0.%i" % (i+1)
    threading.Thread(target=ping,args=(hostname,)).start()
```

# Exemple avec multithreading (2)

PingWithoutMultithreading.py | PingWithMultithreading.py | PingWithMultithreading2.py

```
1   import os
2   import subprocess
3   import re
4   import threading
5   regex = re.compile(r"(?P<received>\d+) received")
6
7   class Ping(threading.Thread):
8
9       def __init__(self, hostname):
10          threading.Thread.__init__(self)
11          self.hostname = hostname
12
13      def run(self):
14          p = subprocess.Popen(["ping", "-c1", "-w100", self.hostname], stdout=subprocess.PIPE).stdout.read()
15          r = regex.search(p.decode())
16          try:
17              if(r.group("received") == "1"):
18                  print("L'adresse %s existe!" % self.hostname)
19          except:
20              pass
21
22  for i in range(254):
23      hostname = "192.168.0.%i" % (i+1)
24      background = Ping(hostname)
25      background.start()
```

# Base de données



# Base de données - Définitions

- Une base de données est un conteneur dans lequel il est possible de stocker des données de façon structurée et permanente.
- Un langage standardisé -SQL- permet de faire des recherches mais aussi des modifications ou des suppressions de données d'une base de données

# Exemple avec SQLite

ManageSQlite.py | GestionDesErreurs.py

```
1   import sqlite3
2
3   try:
4       conn = sqlite3.connect('userDatabase.db')
5       cursor = conn.cursor()
6       cursor.execute("""
7           CREATE TABLE users2(
8               id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
9               name TEXT,
10              age INTERGER
11          ) """)
12      conn.commit()
13
14  except sqlite3.OperationalError:
15      print('Erreur la table existe déjà')
16
17  except Exception as e:
18      print("Erreur")
19      conn.rollback() # raise e
20
21  finally:
22      conn.close()
```

# Exemple avec les opérations CRUD

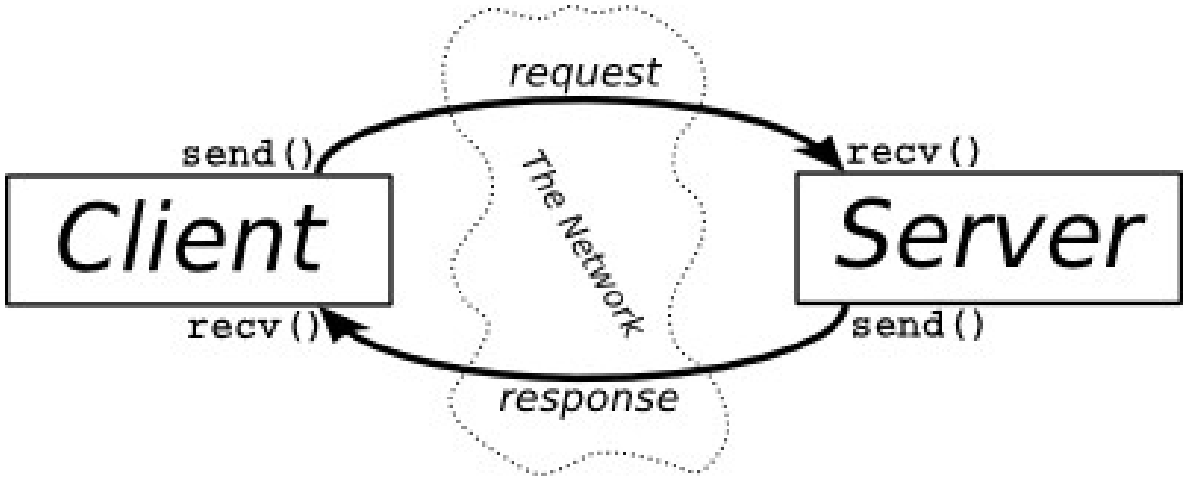
ManageSQlite.py

```python
import sqlite3

#connection
conn = sqlite3.connect('userDatabase.db')

#Suppression d'une table

cursor = conn.cursor()
cursor.execute("""
 DROP TABLE IF EXISTS users
 """)
conn.commit()

#creation d'une table
cursor = conn.cursor()
cursor.execute("""
 CREATE TABLE IF NOT EXISTS users(
      id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
      name TEXT,
      age INTERGER
 )
 """)
conn.commit()

#Insertion des données
cursor.execute("""
 INSERT INTO users(name, age) VALUES(?, ?)""", ("olivier", 30))

#Insertion des données par un dictionnaire
data = {"name" : "paul", "age" : 40}
cursor.execute("""
 INSERT INTO users(name, age) VALUES(:name, :age)""", data)

# trouver le dernier enregistrement
id = cursor.lastrowid
print('dernier id: %d' % id)
```

GestionDesErreurs.py

```python

# Insertion de plusieurs enregistrements
users = []
users.append(("olivier", 30))
users.append(("jean-louis", 90))
cursor.executemany("""
 INSERT INTO users(name, age) VALUES(?, ?)""", users)

#Recuperation 1 donnee
cursor.execute("""SELECT name, age FROM users""")
user1 = cursor.fetchone()
print(user1)

#Recuperation plusieurs donnees
cursor.execute("""SELECT id, name, age FROM users""" )
rows = cursor.fetchall()
for row in rows:
    print('{0} : {1} - {2}'.format(row[0], row[1], row[2]))

#Reccherche specifique
id = 2
cursor.execute("""SELECT id, name FROM users WHERE id=?""", (id,))
response = cursor.fetchone()

# Mise à jour des données
cursor.execute("""UPDATE users SET age = ? WHERE id = 2""", (31,))

# Transactions: Commit ou Rollback
conn.rollback()

# fermer la connexion
conn.close()

```

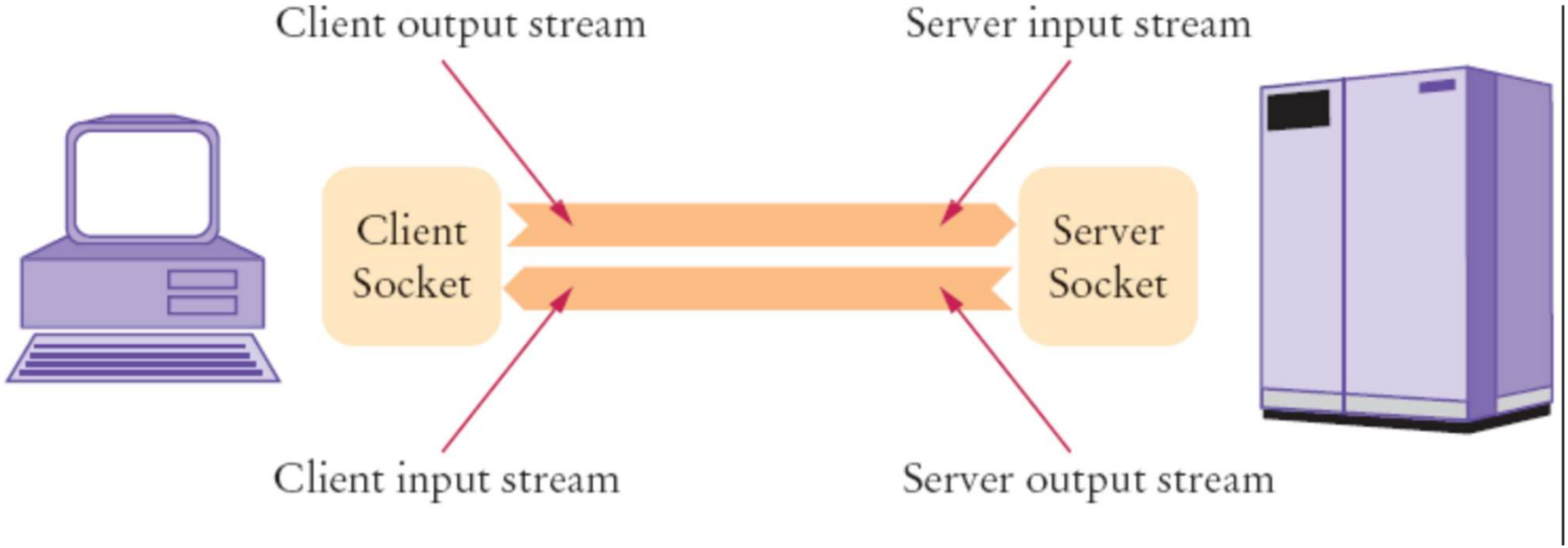
# Programmation Réseau



# Programmation Réseau

- Pourquoi
  - Utiliser un service qui se trouve sur une machine distante
- Questions
  - Comment communiquent des machines distantes?
  - Comment atteindre le bon service?

# Communiquer



# Un socket: définition

- Un socket c'est quoi?
  - En anglais un socket est un "trou" qui laisse passer des choses, comme une prise électrique, l'emplacement du processeur, ou une bouche -on va s'arrêter la pour les exemples -
  - Le socket est donc dans notre cas une association au niveau de l'OS entre un programme qui tourne en boucle et le port de la machine qui lui a été dédié. On dit d'ailleurs que le programme écoute le port qui lui a été réservé. Il écoute le port et répond aux demandes faites par ce port.

# Server & Client (Server.py & Client.py)

Server.py

```python
import socket

socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.bind(('', 15555))

while True:
    socket.listen(5)
    client, address = socket.accept()
    print ("{0} connected".format( address ))
    response = client.recv(255)
    if response != "":
        print (response)

print ("Close")
client.close()
stock.close()
```

Client.py

```python
import socket

hote = "localhost"
port = 15555

socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.connect((hote, port))

print ("Connection on {0}".format(port))
socket.send("Hey my name is Olivier!".encode())

print ("Close")
socket.close()
```

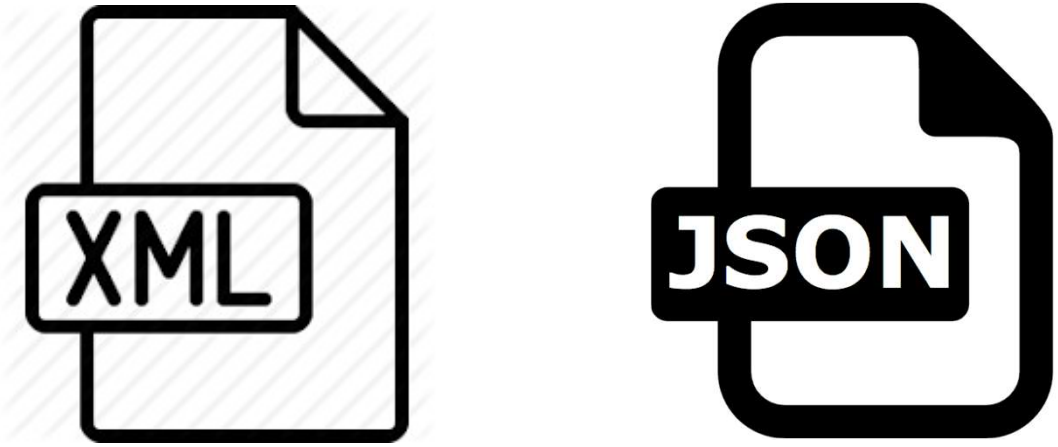
# Envoi de message

# Envoi de message

SendToGMail.py

```
1   import smtplib
2   # from email.MIMEMultipart import MIMEMultipart
3   # from email.MIMEText import MIMEText
4
5   from email.mime.multipart import MIMEMultipart
6   from email.mime.text import MIMEText
7
8   msg = MIMEMultipart()
9   msg['From'] = 'XXX@gmail.com'
10  msg['To'] = 'XXX@gmail.com'
11  msg['Subject'] = 'Le sujet de mon mail'
12  message = 'Bonjour !'
13  msg.attach(MIMEText(message))
14  mailserver = smtplib.SMTP('smtp.gmail.com', 587)
15  mailserver.ehlo()
16  mailserver.starttls()
17  mailserver.ehlo()
18  mailserver.login('XXX@gmail.com', 'PASSWORD')
19  mailserver.sendmail('XXX@gmail.com', 'XXX@gmail.com'
20   , msg.as_string())
21  mailserver.quit()
```

# XML et JSON



# XML et JSON

## XML

```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40></age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

## JSON

```
{  "empinfo" :
      {
            "employees" :  [
            {
                  "name" : "James Kirk",
                  "age" : 40,
            },
            {
                  "name" : "Jean-Luc Picard",
                  "age" : 45,
            },
            {
                  "name" : "Wesley Crusher",
                  "age" : 27,
            }
                              ]
      }
}
```

# XML - lecture

personnages.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user data-id="101">
        <nom>Zorro</nom>
        <metier>Danseur</metier>
    </user>
    <user data-id="102">
        <nom>Hulk</nom>
        <metier>Footballeur</metier>
    </user>
    <user data-id="103">
        <nom>Zidane</nom>
        <metier>Star</metier>
    </user>
    <user data-id="104">
        <nom>Beans</nom>
        <metier>Epicier</metier>
    </user>
    <user data-id="105">
        <nom>Batman</nom>
        <metier>Veterinaire</metier>
    </user>
    <user data-id="106">
        <nom>Spiderman</nom>
        <metier>Veterinaire</metier>
    </user>
</users>
```

Personnages.py

```python
from lxml import etree

tree = etree.parse("personnages.xml")

for user in tree.xpath("/users/user/nom"):
    print(user.text)

for user in tree.xpath("/users/user"):
    print(user.get("data-id"))

for user in tree.xpath("/users/user[metier='Veterinaire']/nom"):
    print(user.text)
```

# XML - création

personnages.xml | Personnages.py | GenerateUsers.py

```python
from lxml import etree

users = etree.Element("users")
user = etree.SubElement(users, "user")
user.set("data-id", "101")
nom = etree.SubElement(user, "nom")
nom.text = "Zorro"
metier = etree.SubElement(user, "metier")
metier.text = "Danseur"
print(etree.tostring(users, pretty_print=True))
```

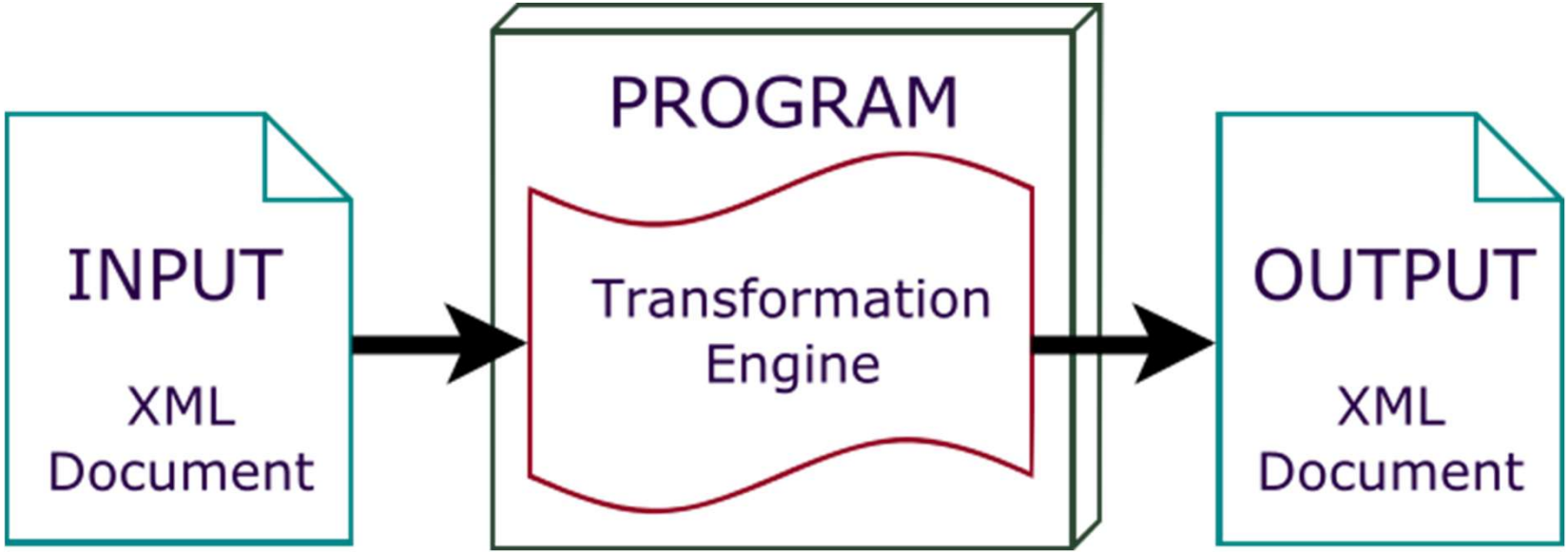
personnages.xml | Personnages.py | GenerateUsers2.py

```python
from lxml import etree

users = etree.Element("users")
users_data = [
 ("101", "Zorro", "Danseur"),
 ("102", "Hulk", "Footballeur"),
 ("103", "Zidane", "Star"),
 ("104", "Beans", "Epicier"),
 ("105", "Batman", "Veterinaire"),
 ("106", "Spiderman", "Veterinaire"),
 ]
for user_data in users_data:
    user = etree.SubElement(users, "user")
    user.set("data-id", user_data[0])
    nom = etree.SubElement(user, "nom")
    nom.text = user_data[1]
    metier = etree.SubElement(user, "metier")
    metier.text = user_data[2]

print(etree.tostring(users, pretty_print=True))
```

# XML – Pour aller plus loin

- addnext(element)
- addprevious(element)
- append(element)
- clear()
- extends(elements)
- findall(path)
- findtext(path)
- get(key)
- getchildren() :
- *list(element)*
- find(path)
- getnext()
- getparent()
- getprevious()
- index(child)
- insert(index)
- keys()
- remove(element)
- replace(el1, el2)
- set(key, value)
- values() ributs
- xpath(path)
- items()

# XML Transformation



# XML Transformation - XML

personnages.xml | Personnages.py | GenerateUsers2.py | studyData.xml | studyData.xslt

```xml
<contents>
  <part title="Lucene Basics(or Fundamentals)">
    <chapter title="Lucene Searching">
      <section type="internal" title="Lucene Scoring">
        <leaf title="How Lucene scoring works" seotitle="how-lucene-scoring-works">
        </leaf>
      </section>
      <section type="terminal" title="" seotitle="">
        <leaf title="hello world" seotitle="how-lucene-scoring-works">
        </leaf>
      </section>
    </chapter>
  </part>

  <part title="Lucene Index">
    <chapter title="Lucene Searching">
      <section type="internal" title="Lucene Scoring">
        <leaf title="How Lucene indexing works" seotitle="how-lucene-indexing-works">
        </leaf>
        <leaf title="Lucene Index tutorial" seotitle="lucene-index-tutorial">
        </leaf>
      </section>
      <section type="terminal" title="" seotitle="">
      </section>
    </chapter>
  </part>
</contents>
```

# XML Transformation - XSLT

personnages.xml | Personnages.py | GenerateUsers2.py | studyData.xml | studyData.xslt

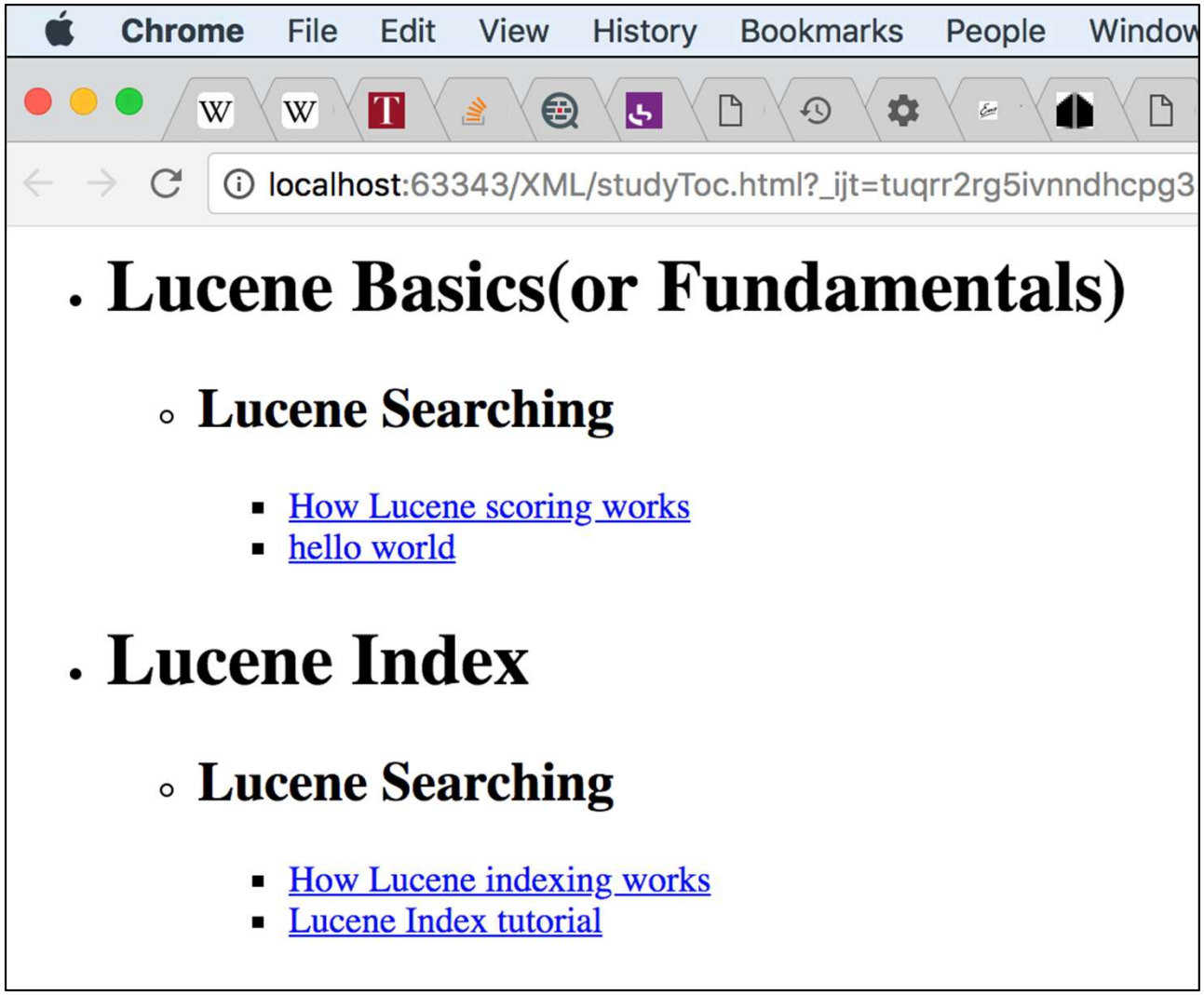
```xml
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <div class="toc-contents">
            <ul>
                <xsl:apply-templates/>
            </ul>
        </div>
    </xsl:template>
    <xsl:template match="part">
        <li>
         <div class="toc-part">
            <h1><xsl:value-of select="@title"/></h1>
            <ul>
                <xsl:apply-templates select="chapter"/>
            </ul>
         </div>
        </li>
    </xsl:template>
    <xsl:template match="chapter">
        <li>
         <div class="toc-chapter">
            <h2><xsl:value-of select="@title"/></h2>
            <ul>
                <xsl:apply-templates select=".//leaf"/>
            </ul>
         </div>
        </li>
    </xsl:template>
    <xsl:template match="leaf">

        <li>
            <a>
                <xsl:attribute name="href">
                    http://makble.com/<xsl:value-of select="@seotitle"/>
                </xsl:attribute>
                <xsl:value-of select="@title"/></a>
        </li>
    </xsl:template>
</xsl:stylesheet>

```

# XML Transformation - Code

personnages.xml | Personnages.py | StudyDataTransform.py

```
1   import lxml.html
2   from lxml import etree
3
4   xsltDoc = etree.parse("studyData.xslt")
5   xsltTransformer = etree.XSLT(xsltDoc)
6
7   sourceDoc = etree.parse("studyData.xml")
8   outputDoc = xsltTransformer(sourceDoc)
9
10  print(str(outputDoc))
11  outputDoc.write("studyToc.html", pretty_print=True)
```

# XML Transformation



# XML Transformation - CSV

Personnages.py | StudyDataTransform.py | residentData.xml | CSVResiden

```xml
<State>
<Resident Id="100">
<Name>Michael Jackson</Name>
    <PhoneNumber>1234567891</PhoneNumber>
    <EmailAddress>mj@example.com</EmailAddress>
    <Address>
        <StreetLine1>Street Line1</StreetLine1>
        <City>Nebraska</City>
        <StateCode>NB</StateCode>
        <PostalCode>12345</PostalCode>
    </Address>
</Resident>
<Resident Id="101">
    <Name>Donald Trump</Name>
    <PhoneNumber>22222222</PhoneNumber>
    <EmailAddress>donald.trump@example.com</EmailAddress>
    <Address>
        <StreetLine1>Pennslyvania Street</StreetLine1>
        <City>Washington DC</City>
        <StateCode>WA</StateCode>
        <PostalCode>56666</PostalCode>
    </Address>
</Resident>
    <Resident Id="102">
    <Name>Paul Simon</Name>
    <PhoneNumber>88888888</PhoneNumber>
    <EmailAddress>paul.simon@example.com</EmailAddress>
    <Address>
        <StreetLine1>19, Plade des Augustins</StreetLine1>
        <City>Geneve</City>
        <StateCode>GE</StateCode>
        <PostalCode>1205</PostalCode>
    </Address>
</Resident>
    <Resident Id="103">
    <Name>Roger Federer</Name>
    <PhoneNumber>7777777</PhoneNumber>
    <EmailAddress>roger.federer@example.com</EmailAddress>
    <Address>
        <StreetLine1>Banhofstrasse 8</StreetLine1>
        <City>Zurich</City>
        <StateCode>ZH</StateCode>
        <PostalCode>8700</PostalCode>
    </Address>
</Resident>
</State>
```

Personnages.py | StudyDataTransform.py | residentData.xml | CSVResider

```python
import xml.etree.ElementTree as ET
import csv

tree = ET.parse("residentData.xml")
root = tree.getroot()

# open a file for writing

Resident_data = open('ResidentData.csv', 'w')

# create the csv writer object

csvwriter = csv.writer(Resident_data)
resident_head = []

count = 0
for member in root.findall('Resident'):
    resident = []
    address_list = []
    if count == 0:
        name = member.find('Name').tag
        resident_head.append(name)
        PhoneNumber = member.find('PhoneNumber').tag
        resident_head.append(PhoneNumber)
        EmailAddress = member.find('EmailAddress').tag
        resident_head.append(EmailAddress)
        Address = member[3].tag
        resident_head.append(Address)
        csvwriter.writerow(resident_head)
        count = count + 1

    name = member.find('Name').text
    resident.append(name)
    PhoneNumber = member.find('PhoneNumber').text
    resident.append(PhoneNumber)
    EmailAddress = member.find('EmailAddress').text
    resident.append(EmailAddress)
    Address = member[3][0].text
    address_list.append(Address)
    City = member[3][1].text
    address_list.append(City)
    StateCode = member[3][2].text
    address_list.append(StateCode)
    PostalCode = member[3][3].text
    address_list.append(PostalCode)
    resident.append(address_list)
    csvwriter.writerow(resident)
Resident_data.close()
```

Personnages.py | StudyDataTransform.py | residentData.xml | CSVResidentGenerator.py | ResidentData.csv

```
Name,PhoneNumber,EmailAddress,Address
Michael Jackson,1234567891,mj@example.com,"['Street Line1', 'Nebraska', 'NB', '12345']"
Donald Trump,22222222,donald.trump@example.com,"['Pennslyvania Street', 'Washington DC', 'WA', '56666']"
Paul Simon,88888888,paul.simon@example.com,"['19, Plade des Augustins', 'Geneve', 'GE', '1205']"
Roger Federer,7777777,roger.federer@example.com,"['Banhofstrasse 8', 'Zurich', 'ZH', '8700']"
```

# JSON & Python

```
[
  {
    "Platform" : "Android",
    "Favorite Food" : "Noodle!",
    "Language" : "C#"
  },
  {
    "Platform" : "iOS",
    "Favorite Food" : "Pasta!",
    "Language" : "Swift"
  },
  {
    "Platform" : "iOS",
    "Favorite Food" : “Rice!”,
    "Language" : “Java”
  }
]
```

| Python | JSON |
|---|---|
| dict | object |
| list, tuple | array |
| str | string |
| int, float, int- & float-derived Enums | number |
| True | true |
| False | false |
| None | null |

# JSON vs Python

## JSON Document: Flexible Schema

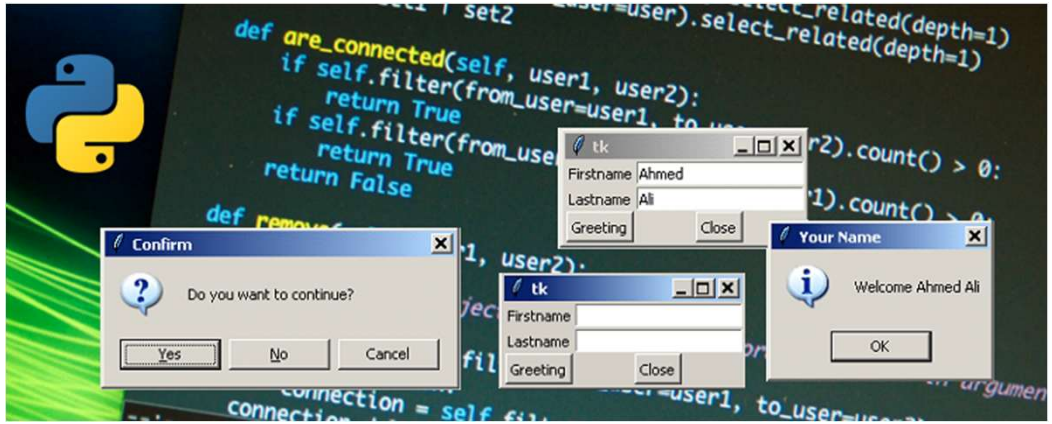
- Complex data model
- Easy to evolve
- Data type support
- Developer friendly

# JSON & Python - Exemples

ReadJSON.py

```python
import json
jsonData = '{"name": "Frank", "age": 39, "address" : {"street": "19, Place des Augustins", "zipCode":"1205", "State": "Geneva"}}'
jsonToPython = json.loads(jsonData)
print(jsonToPython)

#Python Dictionaries to JSON strings
student = {"105":{"class":'V', "Name":'Rohit',  "Roll_no":7},
           "102":{"class":'V', "Name":'David',  "Roll_no":8},
           "103":{"class":'V', "Name":'Samiya', "Roll_no":12}}

print(json.dumps(student, sort_keys=True))

#Python tuple to JSON array
tup1 = 'Red', 'Black', 'White';
print(json.dumps(tup1))

#Python list to JSON array
list1 = [5, 12, 13, 14];
print(json.dumps(list1));

#Python string to JSON string
string1 = 'Python and JSON';
print(json.dumps(string1));

#JSON strings to Python Dictionaries
json_data = '''
            {"103": {"class": "V", "Name": "Samiya", "Roll_n": 12},
             "102": {"class": "V", "Name": "David", "Roll_no": 8},
             "101": {"class": "V", "Name": "Rohit", "Roll_no": 7}}
           '''
print(json.loads(json_data));

#JSON array Python tuple
json_array = ["Red", "Black", "White"]
print(json.dumps(json_array));

#Python list to JSON array
list1 = [5, 12, 13, 14];
print(json.dumps(list1));
```

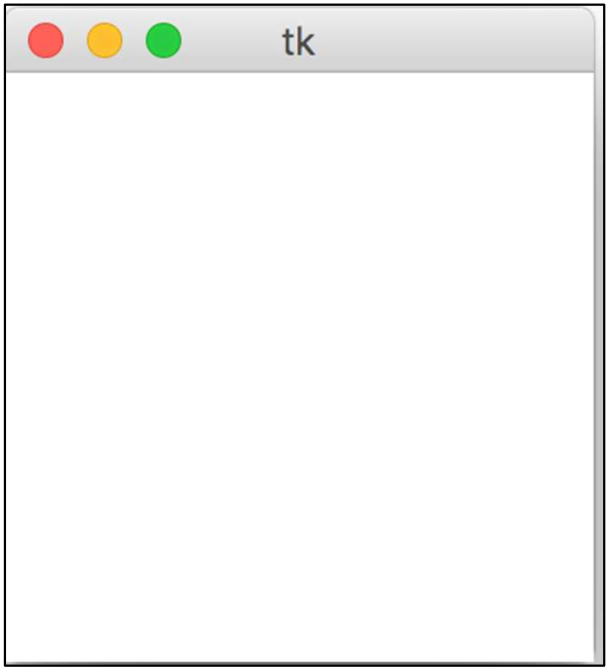
# Interface Graphique (GUI)



# Python Tkinter

- Simple GUI App

```
from tkinter import *

class Window(Frame):

    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master

root = Tk()

app = Window(root)

root.mainloop()
```



# Python Tkinter

```
from tkinter import *

class Window(Frame):

    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
        self.addComponents()

    def addComponents(self):
        lbl = Label(self.master, text="Press the button below to exit")
        lbl.pack()
        btn = Button(self.master, text="Quit", command=self.quit)
        btn.pack()

    def quit(self):
        import sys;
        sys.exit()


root = Tk()
root.title("My First GUI App")

app = Window(root)

root.mainloop()
```

# Python Tkinter MessageBox

```
import tkinter.messagebox
import tkinter.simpledialog
import tkinter.colorchooser

tkinter.messagebox.showinfo("showinfo", "This is an info msg")

tkinter.messagebox.showwarning("showwarning", "This is a warning")

tkinter.messagebox.showerror("showerror", "This is an error")

isYes = tkinter.messagebox.askyesno("ashyesno", "Continue?")
print(isYes)

isOK = tkinter.messagebox.askokcancel("ashokcancle", "OK?")
print(isOK)

isYesNoCancel = tkinter.messagebox.askyesnocancel(
    "askyesnocancel", "Yes, No, Cancel?")
print(isYesNoCancel)

name = tkinter.simpledialog.askstring(
    "askstring", "Enter your name")
print(name)

age = tkinter.simpledialog.askinteger(
    "askinteger", "Enter your age")
print(age)

weight = tkinter.simpledialog.askfloat(
    "askfloat", "Enter your weight")
print(weight)
```

# Tkinter – Evénements

```
from tkinter import *
def hello(event):
    print("Single Click, Button-1")
def quit(event):
    print("Double Click, so let's stop")
    import sys; sys.exit()

widget = Button(None, text='Mouse Clicks')
widget.pack()
widget.bind('<Button-1>', hello)
widget.bind('<Double-1>', quit)
widget.mainloop()
```

```
from tkinter import *

def motion(event):
  print("Mouse position: (%s %s)" % (event.x, event.y))
  return

master = Tk()
whatever_you_do = "Whatever you do will be insignificant, but it is very
important that you do
it.\n(Mahatma Gandhi)"
msg = Message(master, text = whatever_you_do)
msg.config(bg='lightgreen', font=('times', 24, 'italic'))
msg.bind('<Motion>',motion)
msg.pack()
mainloop()
```

# Types d'événements et descriptions

| Event | Description |
|---|---|
| <Button> | A mouse button is pressed with the mouse pointer over the widget. The detail <Button-1>, the middle button by <Button-2>, and the rightmost mouse button by <Button-3>. |
| | <Button-4> defines the scroll up event on mice with wheel support and and <Button-5> the scroll down. |
| <Motion> | The mouse is moved with a mouse button being held down. Details <B1-Motion>, <B2-Motion> and <B3-Motion> respectively. Position of the mouse pointer is provided in the x and y members of the event object passed to the callback, i.e. event.x, event.y |
| <ButtonRelease> | Event, if a button is released. To specify the left, middle or right mouse button use <ButtonRelease-1>, <ButtonRelease-2>, and <ButtonRelease-3> respectively. The current position of the mouse pointer is provided in the x and y members of the event object passed to the callback, i.e. event.x, event.y |
| <Double-Button> | Similar to the Button event, see above, but the button is double clicked instead of a single click. To specify the left, middle or right mouse button use <Double-Button-1>, <Double-Button-2>, and <Double-Button-3> respectively. |
| | You can use Double or Triple as prefixes. Note that if you bind to both a single click (<Button-1>) and a double click (<Double-Button-1>), both bindings will be called. |
| <Enter> | The mouse pointer entered the widget. |
| | Attention: This doesn't mean that the user pressed the Enter key!. <Return> is used for this purpose. |
| <Leave> | The mouse pointer left the widget. |
| <FocusIn> | Keyboard focus was moved to this widget, or to a child of this widget. |
| <FocusOut> | Keyboard focus was moved from this widget to another widget. |
| <Return> | The user pressed the Enter key. You can bind to virtually all keys on the keyboard: The special keys are Cancel (the Break key), BackSpace, Tab, Return(the Enter key), Shift_L (any Shift key), Control_L (any Control key), Alt_L (any Alt key), Pause, Caps_Lock, Escape, Prior (Page Up), Next (Page Down), End, Home, Left, Up, Right, Down, Print, Insert, Delete, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, Num_Lock, and Scroll_Lock. |
| <Key> | The user pressed any key. The key is provided in the char member of the event object passed to the callback (this is an empty string for special keys). |
| a | The user typed an "a" key. Most printable characters can be used as is. The exceptions are space (<space>) and less than (<less>). Note that 1 is a keyboard binding, while <1> is a button binding. |
| <Shift-Up> | The user pressed the Up arrow, while holding the Shift key pressed. You can use prefixes like Alt, Shift, and Control. |
| <Configure> | The size of the widget changed. The new size is provided in the width and height attributes of the event object passed to the callback. On some platforms, it can mean that the location changed. |

# Evénements

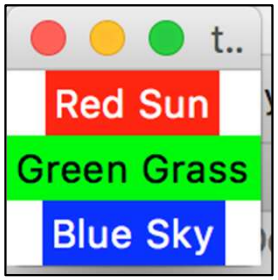
```
<Button-1>          : Click gauche
<Button-2>          : Click milieu
<Button-3>          : Click droit
<Double-Button-1>   : Double click droit
<Double-Button-2>   : Double click gauche
<KeyPress>          : Pression sur une touche
<KeyPress-a>        : Pression sur la touche A (minuscule)
<KeyPress-A>        : Pression sur la touche A (majuscule)
<Return>            : Pression sur la touche entrée
<Escape>            : Touche Echap
<Up>                : Pression sur la flèche directionnelle haut
<Down>              : Pression sur la flèche directionnelle bas
<ButtonRelease>     : Lorsque qu'on relache le click
<Motion>            : Mouvement de la souris
<B1-Motion>         : Mouvement de la souris avec click gauche
<Enter>             : Entrée du curseur dans un widget
<Leave>             : Sortie du curseur dans un widget
<Configure>         : Redimensionnement de la fenêtre
<Map> <Unmap>       : Ouverture et iconification de la fenêtre
<MouseWheel>        : Utilisation de la roulette
```

# Tkinter Layout Managers

- Pack
- Grid
- Place

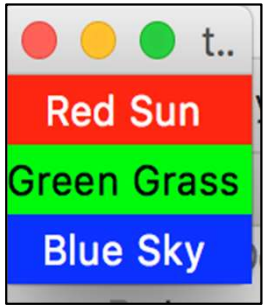
# Tkinter – Layout management - pack

```
1  from tkinter import *
2
3  root = Tk()
4
5  Label(root, text="Red Sun", bg="red", fg="white").pack()
6  Label(root, text="Green Grass", bg="green", fg="black").pack()
7  Label(root, text="Blue Sky", bg="blue", fg="white").pack()
8
9  mainloop()
```



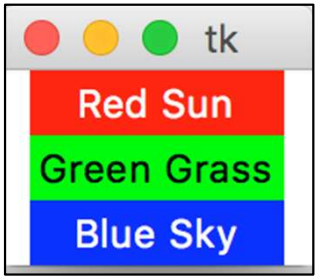
# Tkinter – Layout management - fill

```
1   from tkinter import *
2
3   root = Tk()
4
5   w = Label(root, text="Red Sun", bg="red", fg="white")
6   w.pack(fill=X)
7   w = Label(root, text="Green Grass", bg="green", fg="black")
8   w.pack(fill=X)
9   w = Label(root, text="Blue Sky", bg="blue", fg="white")
10  w.pack(fill=X)
11
12  mainloop()
```



# Tkinter – Layout management – external padding (X et Y)

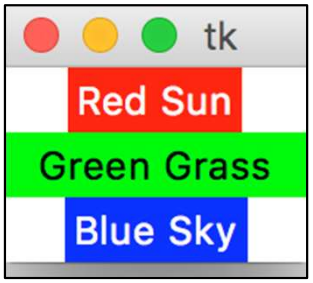
```
1  from tkinter import *
2  root = Tk()
3  w = Label(root, text="Red Sun", bg="red", fg="white")
4  w.pack(fill=X,padx=10)
5  w = Label(root, text="Green Grass", bg="green", fg="black")
6  w.pack(fill=X,padx=10)
7  w = Label(root, text="Blue Sky", bg="blue", fg="white")
8  w.pack(fill=X,padx=10)
9  mainloop()
```



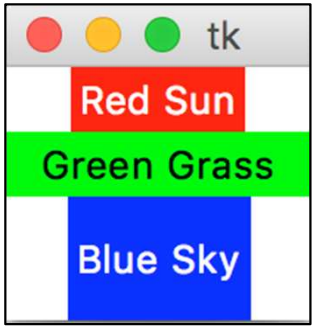

```
1  from tkinter import *
2  root = Tk()
3  w = Label(root, text="Red Sun", bg="red", fg="white")
4  w.pack(fill=X,pady=10)
5  w = Label(root, text="Green Grass", bg="green", fg="black")
6  w.pack(fill=X,pady=10)
7  w = Label(root, text="Blue Sky", bg="blue", fg="white")
8  w.pack(fill=X,pady=10)
9  mainloop()
```

# Tkinter – Layout management – internal padding (horizontally et vertically)

```
1  from tkinter import *
2  root = Tk()
3  w = Label(root, text="Red Sun", bg="red", fg="white")
4  w.pack()
5  w = Label(root, text="Green Grass", bg="green", fg="black")
6  w.pack(ipadx=10)
7  w = Label(root, text="Blue Sky", bg="blue", fg="white")
8  w.pack()
9  mainloop()
10
```




```
1  from tkinter import *
2  root = Tk()
3  w = Label(root, text="Red Sun", bg="red", fg="white")
4  w.pack()
5  w = Label(root, text="Green Grass", bg="green", fg="black")
6  w.pack(ipadx=10)
7  w = Label(root, text="Blue Sky", bg="blue", fg="white")
8  w.pack(ipady=10)
9  mainloop()
```

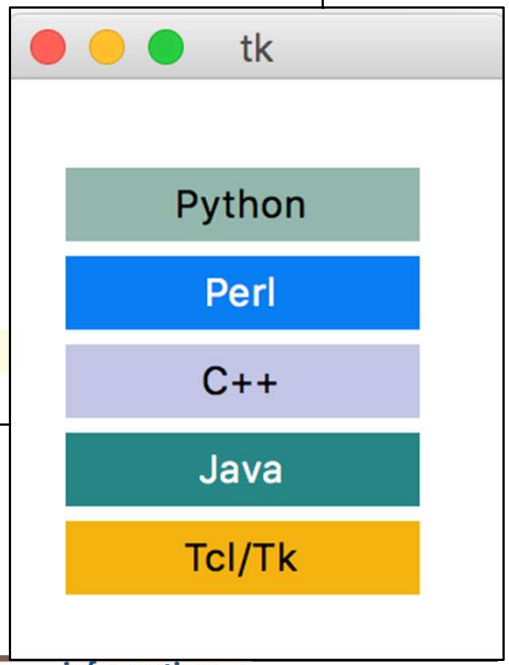


# Tkinter – Layout management – side by side

```
1   from tkinter import *
2
3   root = Tk()
4
5   w = Label(root, text="red", bg="red", fg="white")
6   w.pack(padx=5, pady=10, side=LEFT)
7   w = Label(root, text="green", bg="green", fg="black")
8   w.pack(padx=5, pady=20, side=LEFT)
9   w = Label(root, text="blue", bg="blue", fg="white")
10  w.pack(padx=5, pady=20, side=LEFT)
11
12  mainloop()
```

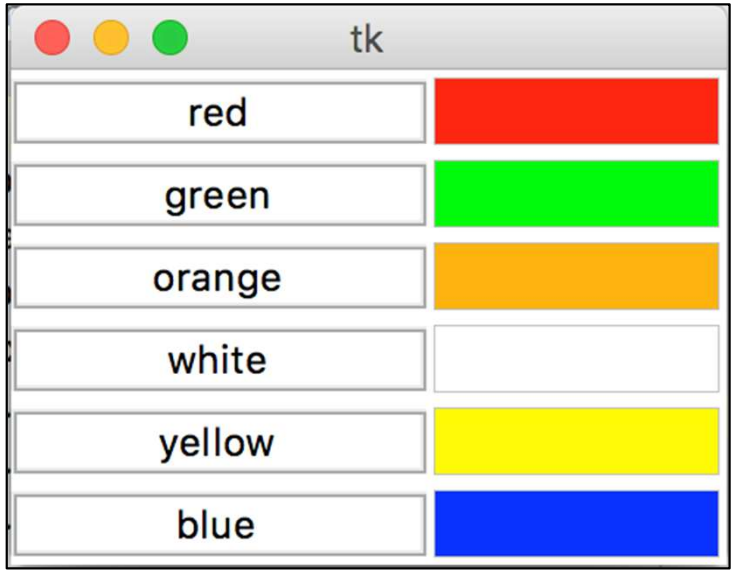
# Tkinter – Layout management - geometry

```
1   import tkinter as tk
2   import random
3
4   root = tk.Tk()
5   # width x height + x_offset + y_offset:
6   root.geometry("170x200+30+30")
7
8   languages = ['Python', 'Perl', 'C++', 'Java', 'Tcl/Tk']
9   labels = range(5)
10  for i in range(5):
11      ct = [random.randrange(256) for x in range(3)]
12      brightness = int(round(0.299 * ct[0] + 0.587 * ct[1] + 0.114 * ct[2]))
13      ct_hex = "%02x%02x%02x" % tuple(ct)
14      bg_colour = '#' + "".join(ct_hex)
15      l = tk.Label(root,
16                   text=languages[i],
17                   fg='White' if brightness < 120 else 'Black',
18                   bg=bg_colour)
19      l.place(x=20, y=30 + i * 30, width=120, height=25)
20
21  root.mainloop()
```

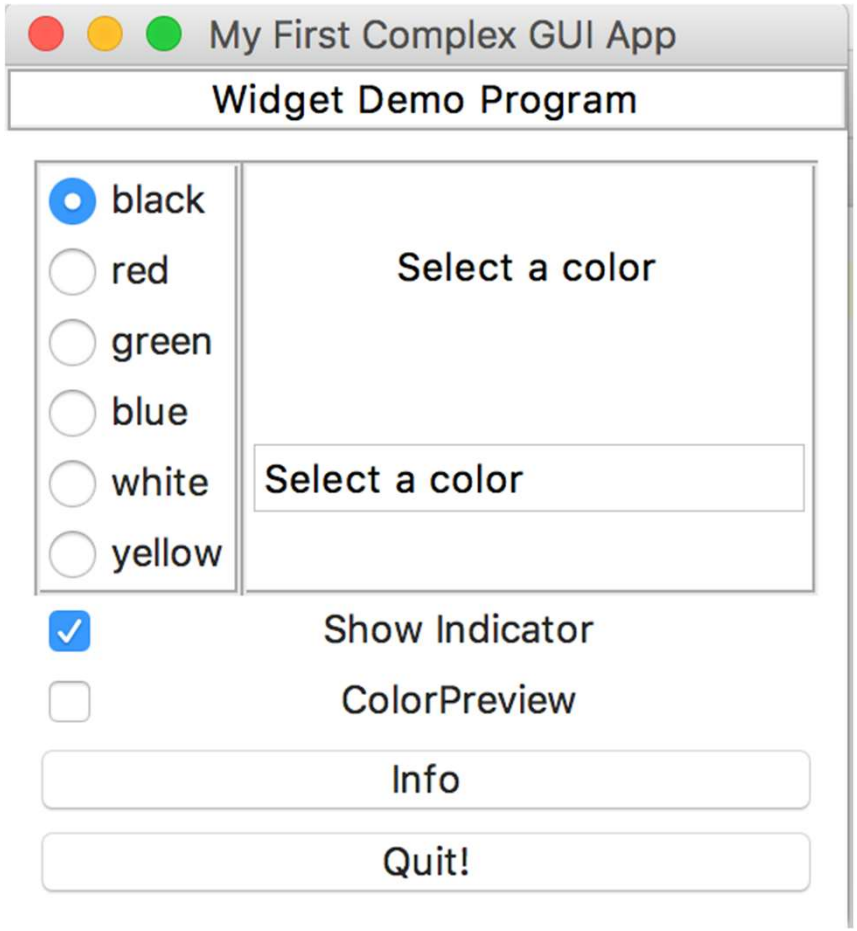


# Tkinter – Layout management - grid

```
from tkinter import *

colours = ['red','green','orange','white','yellow','blue']

r = 0
for c in colours:
    Label(text=c, relief=RIDGE,width=15).grid(row=r,column=0)
    Entry(bg=c, relief=SUNKEN,width=10).grid(row=r,column=1)
    r = r + 1

mainloop()
```



# Tkinter more …



# Tkinter more …

```
from tkinter import *

class Window(Frame):

    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.addComponents(master)

    def addComponents(self, master):

        self.master = master

        frame = Frame(master)

        self.headLbl = Label(frame, text="Widget Demo Program", relief=RIDGE)
        self.headLbl.pack(side=TOP, fill=X)
        # Create a border using a dummy frame with a large border width
        spacerFrame = Frame(frame, borderwidth=10)

        centerFrame = Frame(spacerFrame)
        leftColumn = Frame(centerFrame, relief=GROOVE, borderwidth=2)
        rightColumn = Frame(centerFrame, relief=GROOVE, borderwidth=2)

        # Create some colorful widgets
        self.colorLabel = Label(rightColumn, text="Select a color")
        self.colorLabel.pack(expand=YES, fill=X)
        entryText = StringVar(master)
        entryText.set("Select a color")
        self.colorEntry = Entry(rightColumn, textvariable=entryText)
        self.colorEntry.pack(expand=YES, fill=X)

        # Create some Radiobuttons
        self.radioBtns = []
        self.radioVal = StringVar(master)
        btnList = ("black", "red", "green", "blue", "white", "yellow")
        for color in btnList:
            self.radioBtns.append(Radiobutton(leftColumn, text=color, value=color, indicatoron=TRUE,
                                              variable=self.radioVal, command=self.updateColor))
        else:
            if (len(btnList) > 0):
                self.radioVal.set(btnList[0])
            self.updateColor()

        for btn in self.radioBtns:
            btn.pack(anchor=W)
```

# Tkinter more …

```
46
47          # Make the frames visible
48          leftColumn.pack(side=LEFT, expand=YES, fill=Y)
49          rightColumn.pack(side=LEFT, expand=YES, fill=BOTH)
50          centerFrame.pack(side=TOP, expand=YES, fill=BOTH)
51
52          # Create the Indicator Checkbutton
53          self.indicVal = BooleanVar(master)
54          self.indicVal.set(TRUE)
55          self.updateIndic()
56          Checkbutton(spacerFrame, text="Show Indicator", command=self.updateIndic,
57                      variable=self.indicVal).pack(side=TOP, fill=X)
58
59          # Create the Color Preview Checkbutton
60          self.colorprevVal = BooleanVar(master)
61          self.colorprevVal.set(FALSE)
62          self.updateColorPrev()
63          Checkbutton(spacerFrame, text="ColorPreview", command=self.updateColorPrev,
64                      variable=self.colorprevVal).pack(side=TOP, fill=X)
65
66          # Create the Info Button 65
67          Button(spacerFrame, text="Info", command=self.showInfo).pack(side=TOP, fill=X)
68          # Create the Quit Button
69          Button(spacerFrame, text="Quit!", command=self.quit).pack(side=TOP, fill=X)
70          spacerFrame.pack(side=TOP, expand=YES, fill=BOTH)
71          frame.pack(expand=YES, fill=BOTH)
72
73
74      def quit(self):
75          import sys;
76          from tkinter import messagebox
77          ok = messagebox.askokcancel ("Confirm", "Do you really want to quit ?")
78          if ok:
79              sys.exit()
80          else:
81              pass
82
83      def updateColor(self):
84          self.colorLabel.configure(fg=self.radioVal.get())
85          self.colorEntry.configure(fg=self.radioVal.get())
86
87      def updateIndic(self):
88          for btn in self.radioBtns:
89              btn.configure(indicatoron=self.indicVal.get())
90
```

# Tkinter more …

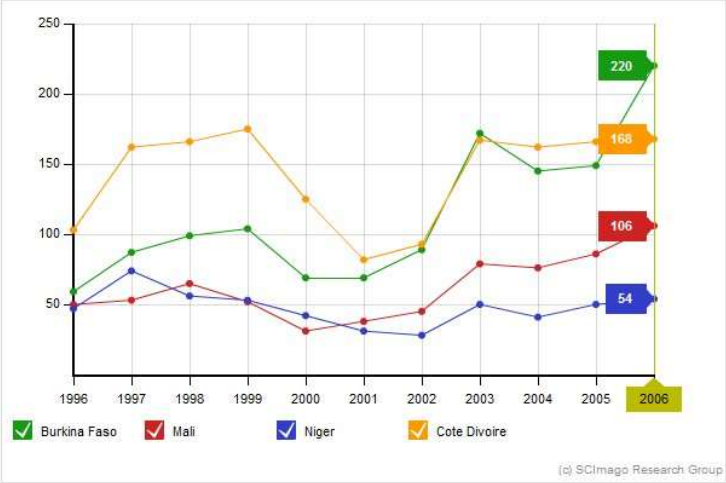
```
91      def updateColorPrev(self):
92          if (self.colorprevVal.get()):
93              for btn in self.radioBtns: color = btn.cget("text")
94              btn.configure(fg=color)
95          else:
96              for btn in self.radioBtns:
97                  btn.configure(fg="black")
98
99      def showInfo(self):
100         toplevel = Toplevel(self.master, bg="pink")
101         toplevel.transient(self.master)
102         toplevel.title("Program info")
103         Label(toplevel, text="Ifage - Cours Python", fg="navy", bg="pink").pack(pady=20)
104         Label(toplevel, text="Written by Ifage-Programmation-Python", bg="white").pack()
105         Label(toplevel, text="http://www.ifage.ch/", bg="white").pack()
106         Button(toplevel, text="Close", command=toplevel.withdraw).pack(pady=30)
107
108     def showMessageBox(self, title="Help", message="Not Yet Implemented"):
109         from tkinter import messagebox
110         messagebox.showinfo(title, message)
111
112     def askOpenFile(self):
113         from tkinter import filedialog
114         filename = filedialog.askopenfilename()
115         self.showMessageBox("Open file" , filename)
116
117     def askSaveAsFile(self):
118         from tkinter import filedialog
119         filename = filedialog.asksaveasfile()
120         self.showMessageBox("Save as", filename)
121
122 def donothing():
123     filewin = Toplevel(root)
124     button = Button(filewin, text="Do nothing button")
125     button.pack()
126
127 root = Tk()
128 root.title("My First Complex GUI App")
129
130 app = Window(root)
131 #app.configure(background='black')
132
133 menubar = Menu(root)
134
```

# Tkinter more …

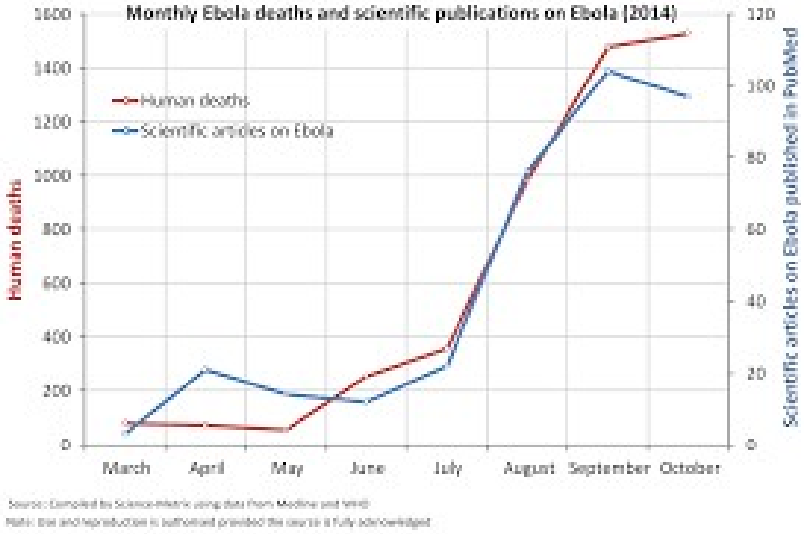
```
133 menubar = Menu(root)
134
135 filemenu = Menu(menubar, tearoff = 0)
136 filemenu.add_command(label="New", command = donothing)
137 filemenu.add_command(label = "Open", command = app.askOpenFile)
138 filemenu.add_command(label = "Save", command = app.askSaveAsFile)
139 filemenu.add_command(label = "Save as...", command = donothing)
140 filemenu.add_command(label = "Close", command = donothing)
141
142 filemenu.add_separator()
143
144 filemenu.add_command(label = "Exit", command = root.quit)
145 menubar.add_cascade(label = "File", menu = filemenu)
146 editmenu = Menu(menubar, tearoff=0)
147 editmenu.add_command(label = "Undo", command = donothing)
148
149 editmenu.add_separator()
150
151 editmenu.add_command(label = "Cut", command = donothing)
152 editmenu.add_command(label = "Copy", command = donothing)
153 editmenu.add_command(label = "Paste", command = donothing)
154 editmenu.add_command(label = "Delete", command = donothing)
155 editmenu.add_command(label = "Select All", command = donothing)
156
157 menubar.add_cascade(label = "Edit", menu = editmenu)
158 helpmenu = Menu(menubar, tearoff=0)
159 helpmenu.add_command(label = "Help Index", command = app.showMessageBox)
160 helpmenu.add_command(label = "About...", command = app.showInfo)
161 menubar.add_cascade(label = "Help", menu = helpmenu)
162
163 root.config(menu = menubar)
164
165 root.mainloop()
```

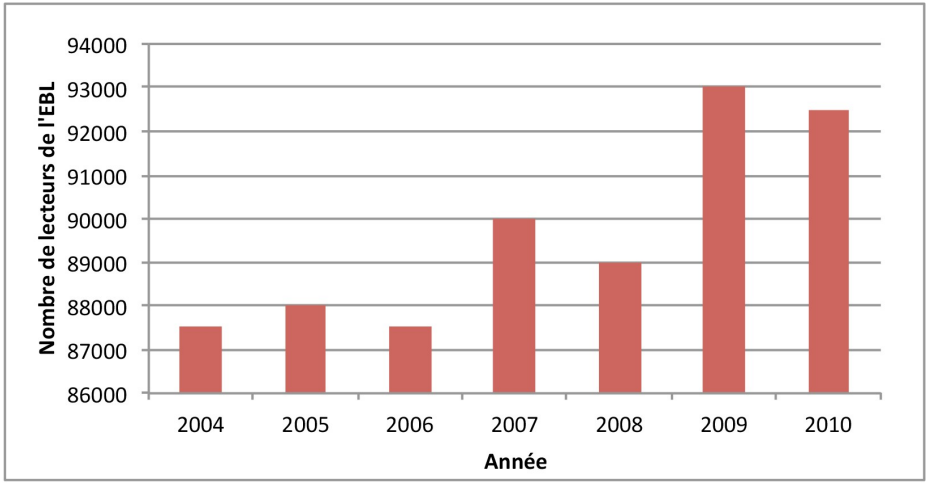
PEP 8: unexpected spaces around keyword / parameter equals

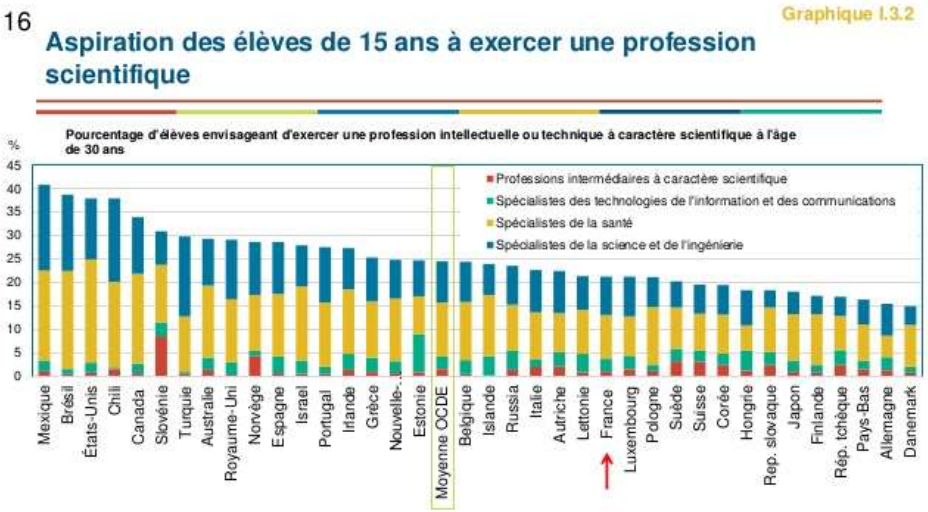
# Programmation Graphique Scientifique



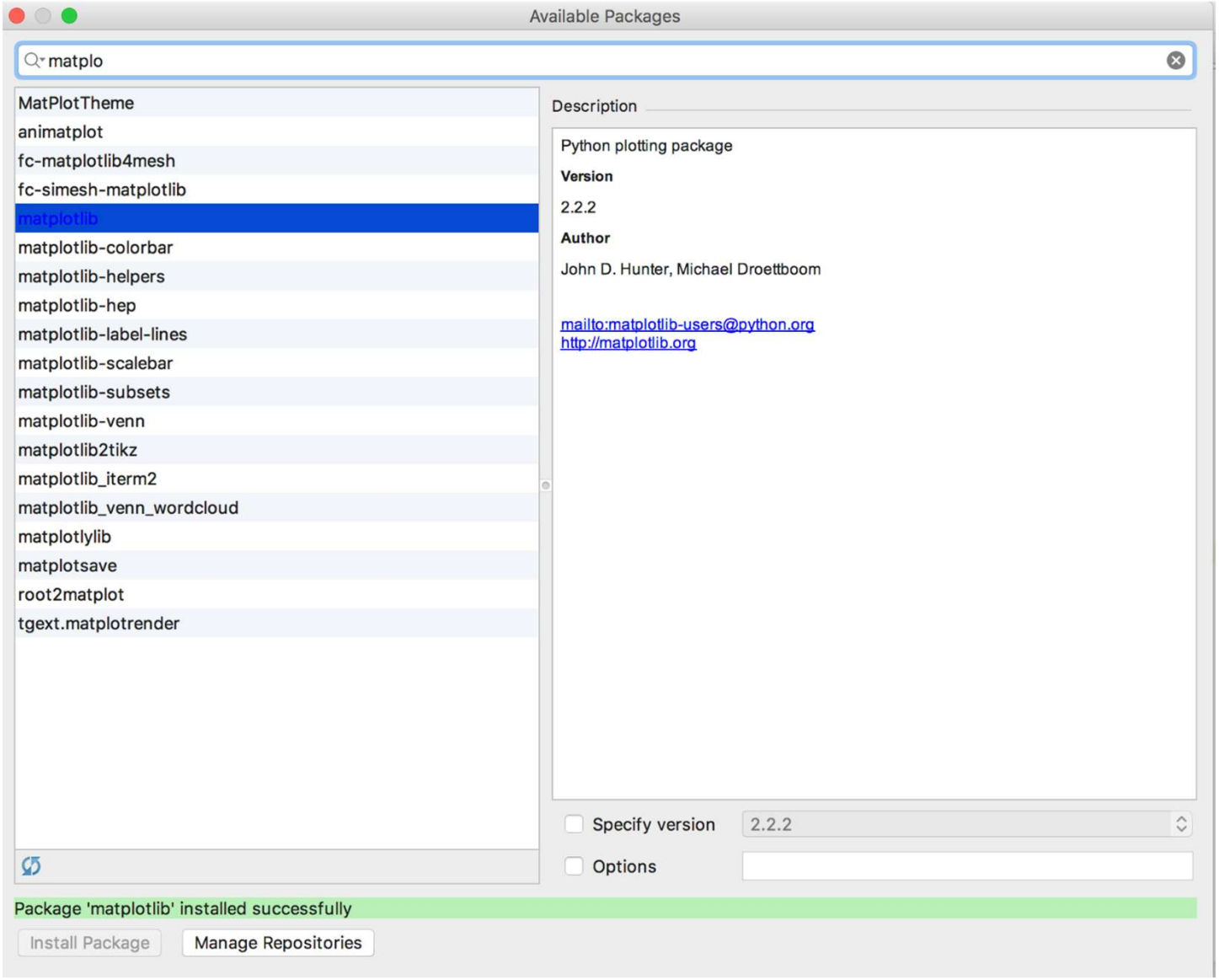
# Graphiques scientifiques



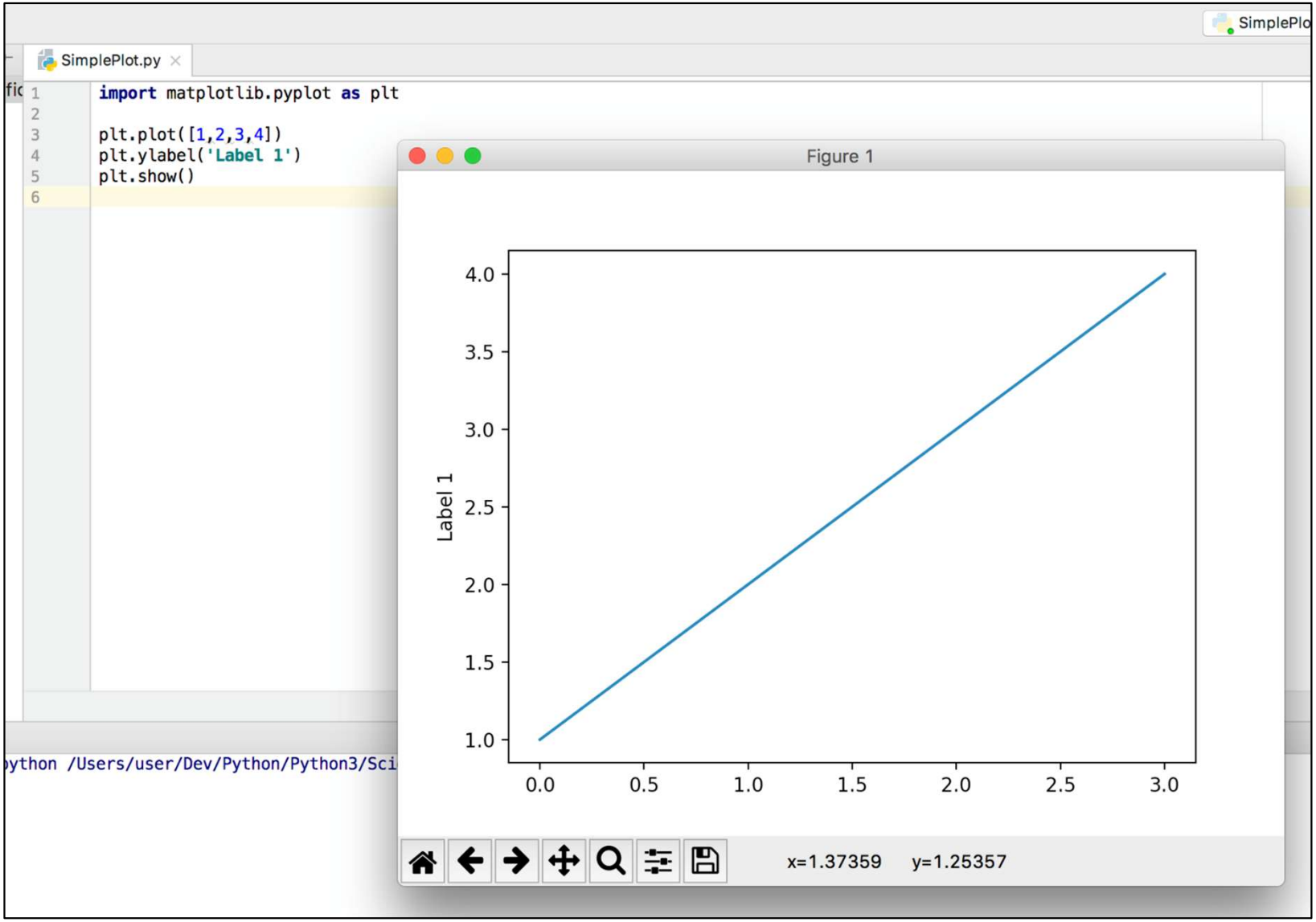



# Exemple: matplotlib



# Exemples de graphes

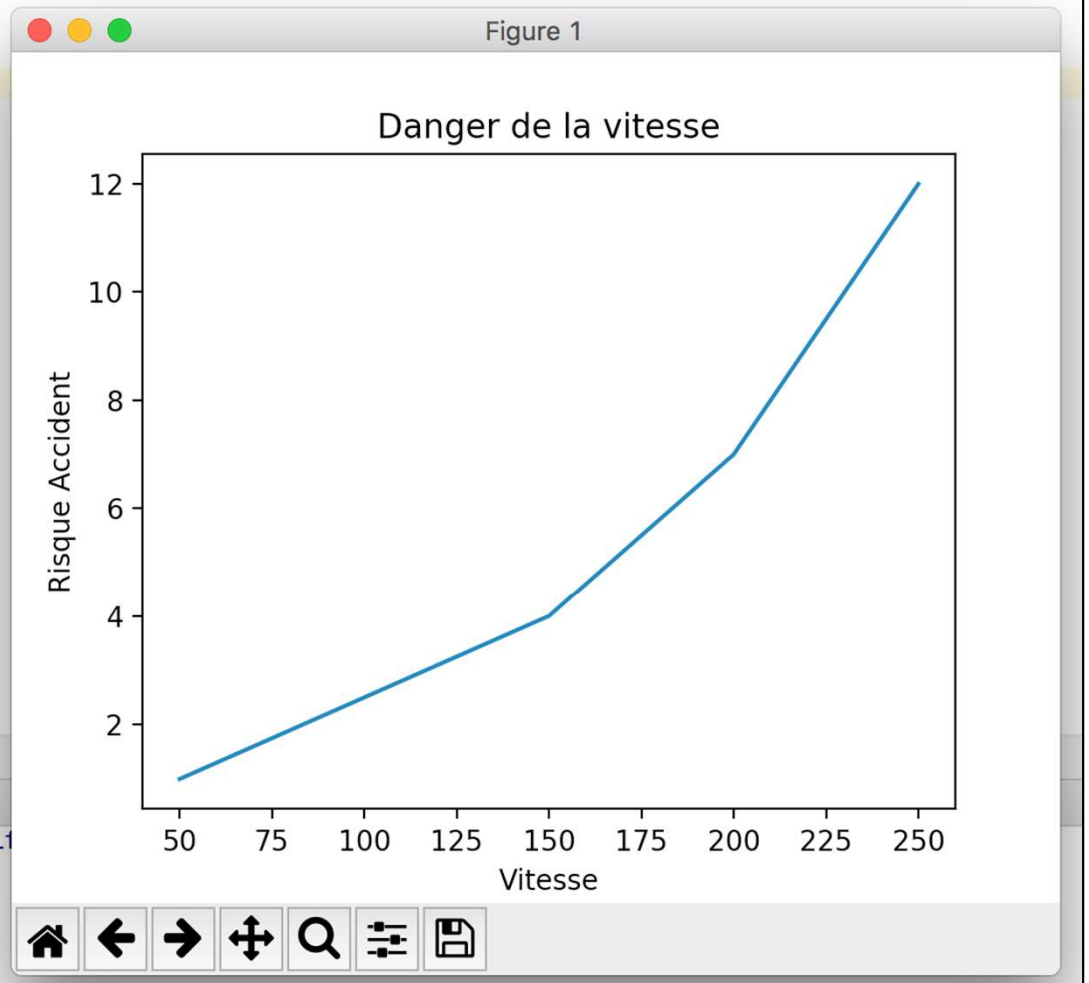


# Exemples de graphes (2)

SimplePlot.py | SimplePlot2.py

```
1  import matplotlib.pyplot as plt
2
3  plt.title("Danger de la vitesse")
4  plt.plot([50,100,150,200, 250], [1,2.5,4,7,12])
5  plt.xlabel('Vitesse')
6  plt.ylabel('Risque Accident')
7  plt.show()
8
```

thon /Users/user/Dev/Python/Python3/Scientif

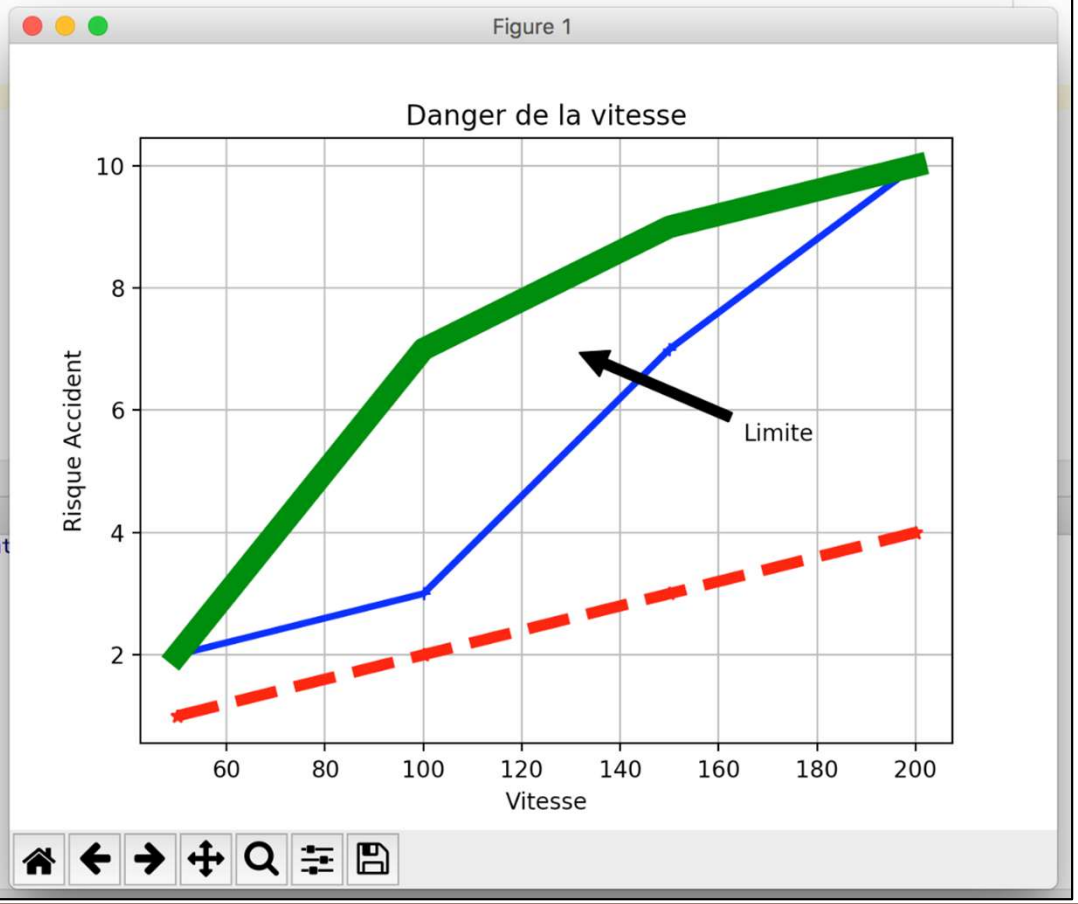


# Exemples de graphes (3)

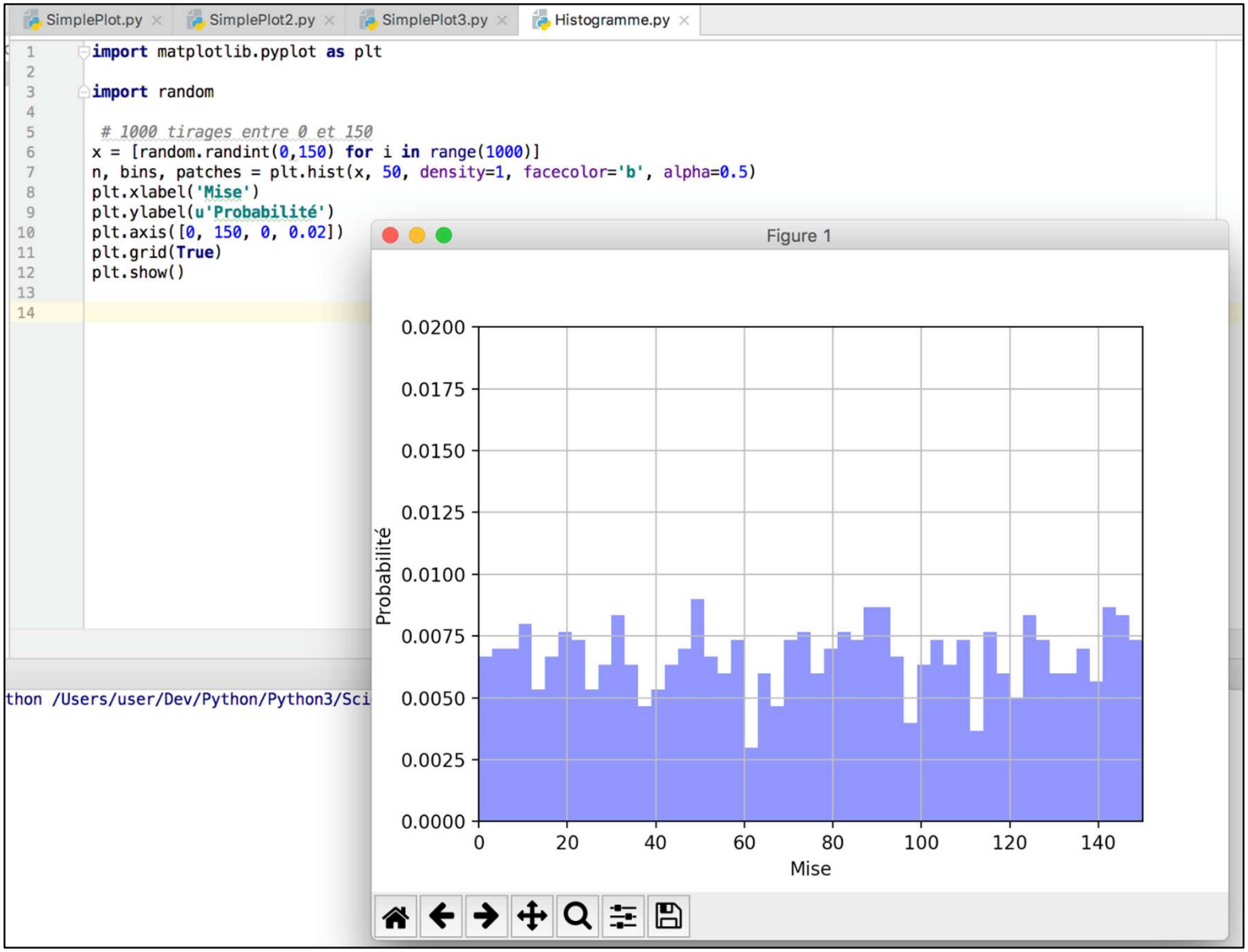
SimplePlot.py | SimplePlot2.py | SimplePlot3.py

```
1  import matplotlib.pyplot as plt
2
3  plt.title("Danger de la vitesse")
4  plt.grid(True)
5
6  plt.plot([50,100,150,200], [1,2,3,4], "r--", linewidth=5, marker="*", label='Suisse')
7  plt.plot([50,100,150,200], [2,3,7,10], "b", linewidth=3, marker="+", label='Espagne')
8  plt.plot([50,100,150,200], [2,7,9,10], "g", linewidth=10, label='Italie')
9
10 plt.annotate('Limite', xy=(130, 7), xytext=(165, 5.5), arrowprops={'facecolor':'black', 'shrink':0.05} )
11
12 plt.xlabel('Vitesse')
13 plt.ylabel('Risque Accident')
14 plt.show()
15
```

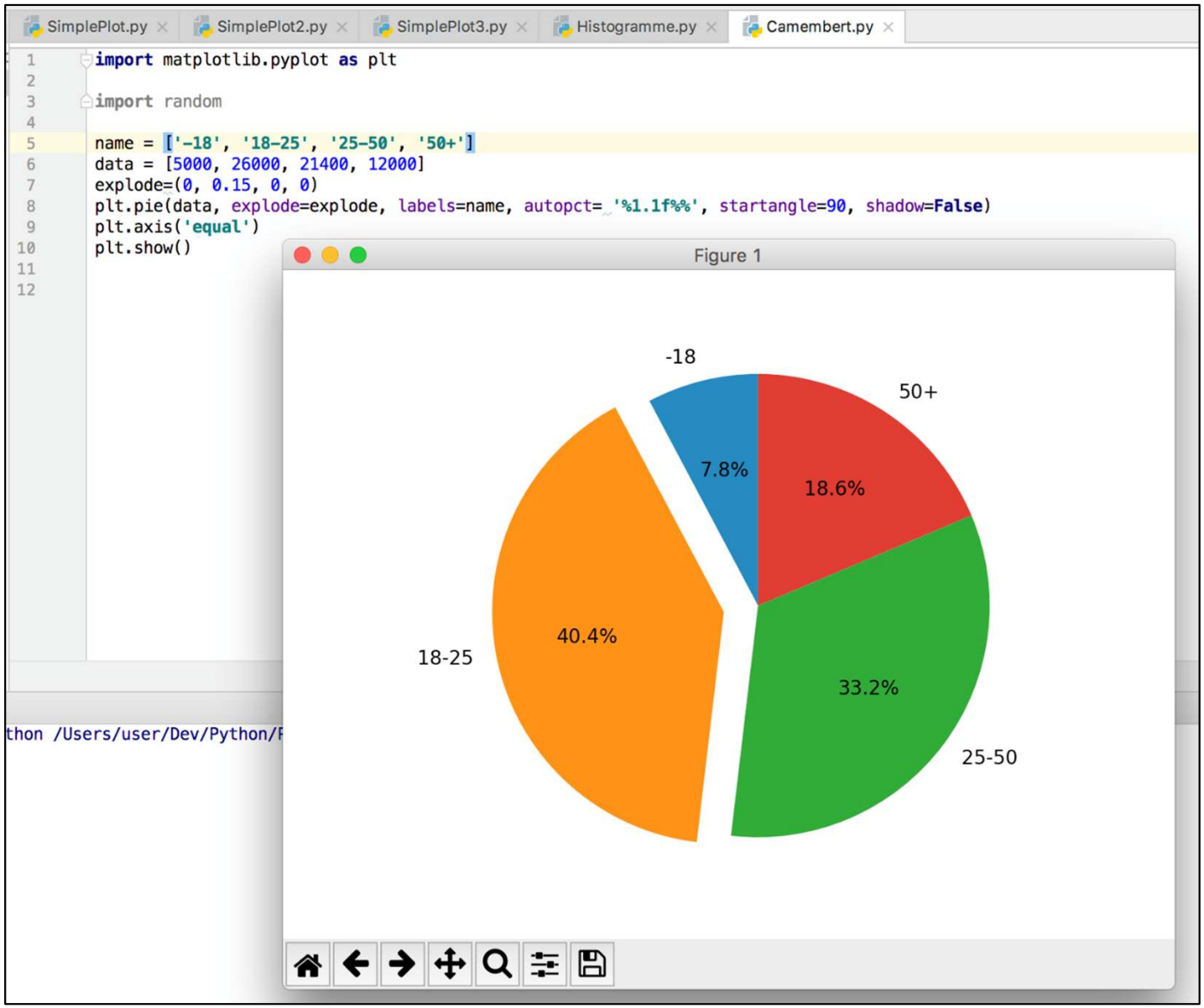
thon /Users/user/Dev/Python/Python3/Scient



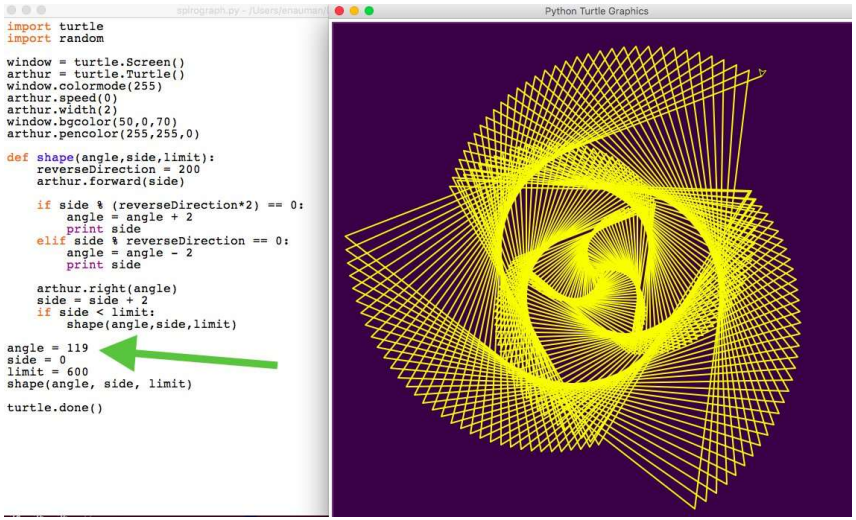
# Exemples de graphes (4)



# Exemples de graphes (5)



# Programmation Graphique
# Dessins/Jeux



# Dessins/Jeux

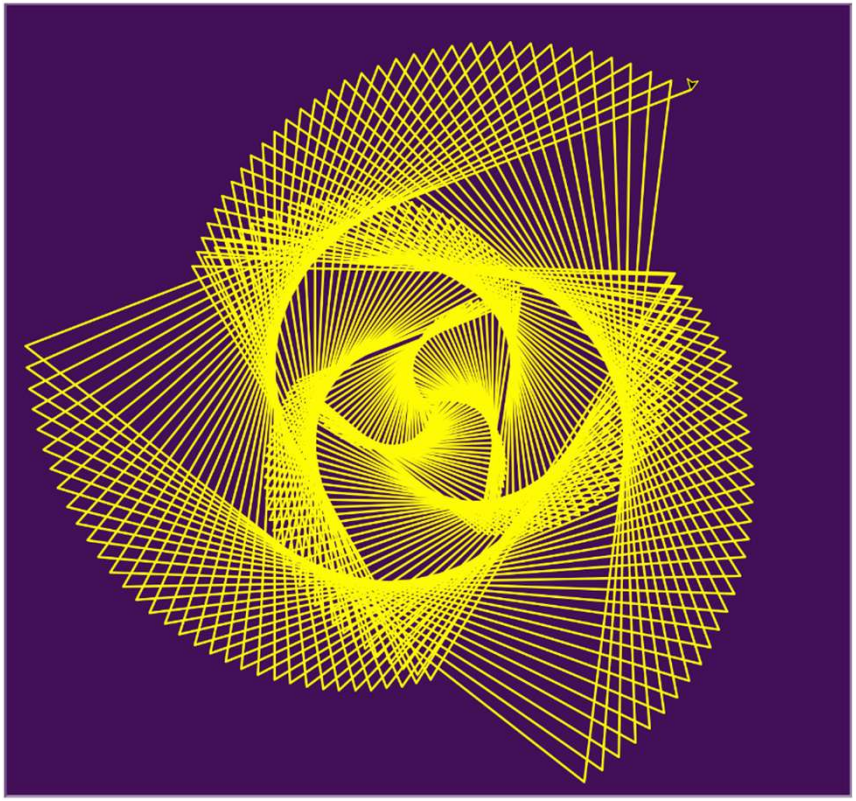
Turtle01.py

```
1
2   import turtle
3   turtle.setup()  #Initialise la fenêtre
4   print(turtle.heading())  #Affiche 0.0 : le crayon pointe vers le point bleu : Est
5   turtle.left(90)  #Pointe vers le point jaune : Nord
6   turtle.right(270)  #Pointe vers le point vert : Ouest
7   turtle.setheading(0)  #Pointe de nouveau vers le point bleu
8   turtle.setheading(-90)  #Pointe à l'opposé du point jaune : Sud
9   print(turtle.heading())  #Affiche '270.0'
10
11  longueur_cote = 100
12  for i in range(8):
13      turtle.forward(longueur_cote)  #Côté
14      turtle.left(360/8)  #Angle
15
16  turtle.circle(120)  #Trace un cercle de rayon 120px
17  turtle.circle(70, 180)  #Trace un demi-cercle de rayon 70px
18  turtle.circle(90, steps = 8)  #Trace un octogone de longueur 90px
19  turtle.circle(40, 180, 4)  #Trace la moitié d'un octogone de longueur 40px
20
21
```

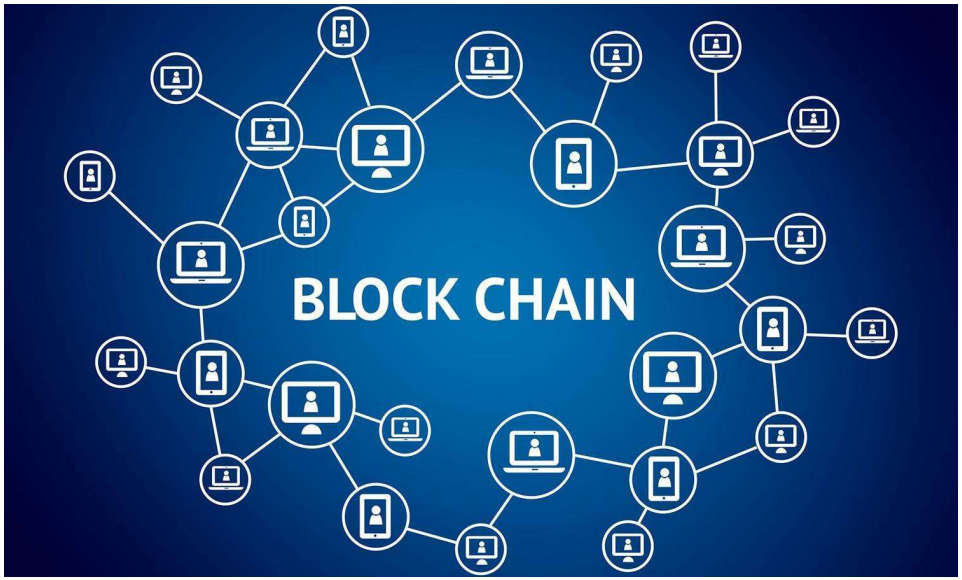
# Dessins/Jeux (2)

Turtle01.py | TurtleLogo.py

```
1   import turtle
2   import random
3
4   window = turtle.Screen()
5   arthur = turtle.Turtle()
6
7   window.colormode(255)
8   arthur.speed(0)
9   arthur.width(2)
10
11  window.bgcolor(50, 0, 70)
12  arthur.pencolor(255, 255, 0)
13
14  def shape(angle, side, limit):
15      reverseDirection = 200
16      arthur.forward(side)
17
18      if side % (reverseDirection * 2) == 0:
19          angle += 2
20          print(side)
21      elif side % reverseDirection == 0:
22          angle -= 2
23          print (side)
24
25      arthur.right(angle)
26      side += 2
27
28      if side < limit:
29          shape( angle, side, limit)
30
31  angle = 119
32  side = 0
33  limit = 600
34  shape(angle, side, limit)
35
36  turtle.done()
37
```

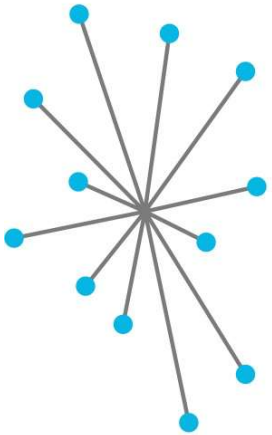


# Blockchain

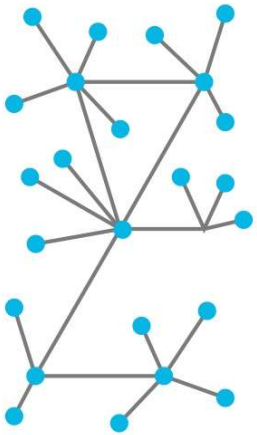


# Blockchain

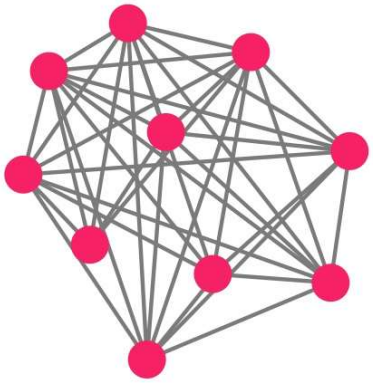
Centralized

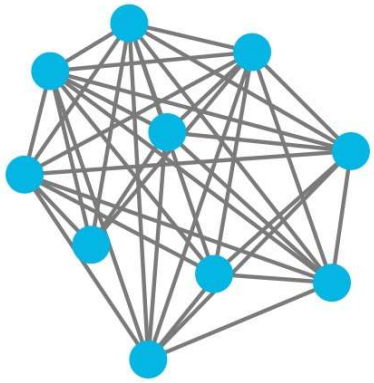


Decentralized



Distributed Ledgers





## The New Networks

Distributed ledgers can be public or private and vary in their structure and size.

Public blockchains

Require computer processing power to confirm transactions ("mining")

- Users (●) are anonymous

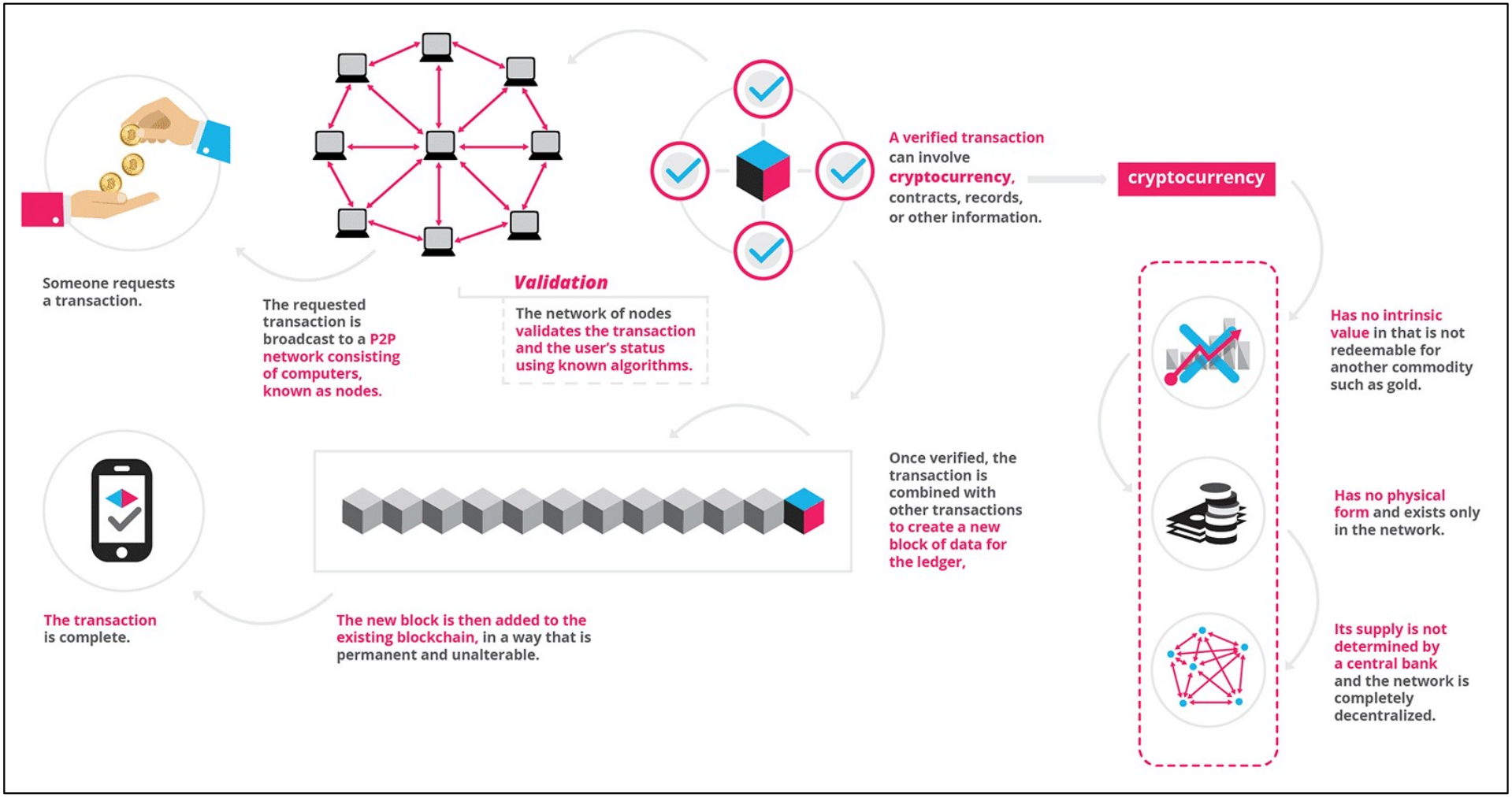
- Each user has a copy of the legder and partipates in confirming transactions independently

- Users (●) are not anonymous

- Permision is required for users to have a copy of the legder and participate in confirming transactions

Blockgeeks

# Fonctionnement



# Définitions

- Introduction au Blockchain
- API et Outils Python

# Blockchain - démo

Blockchain.py

```python
import hashlib
import time
import csv
import random


class Block:
    # A basic block contains, index (blockheight), the previous hash, a timestamp,
    # tx information, a nonce, and the current hash
    def __init__(self, index, previousHash, timestamp, data, proof, currentHash):
        self.index = index
        self.previousHash = previousHash
        self.timestamp = timestamp
        self.data = data
        self.currentHash = currentHash
        self.proof = proof


def getGenesisBlock():
    return Block(0, '0', '1496518102.896031', "My very first block :)", 0,
                 '02d779570304667b4c28ba1dbfd4428844a7cab89023205c66858a40937557f8')


def calculateHash(index, previousHash, timestamp, data, proof):
    value = str(index) + str(previousHash) + str(timestamp) + str(data) + str(proof)
    sha = hashlib.sha256(value.encode('utf-8'))
    return str(sha.hexdigest())


def calculateHashForBlock(block):
    return calculateHash(block.index, block.previousHash, block.timestamp, block.data, block.proof)


def getLatestBlock(blockchain):
    return blockchain[len(blockchain) - 1]


def generateNextBlock(blockchain, blockData, timestamp, proof):
    previousBlock = getLatestBlock(blockchain)
    nextIndex = int(previousBlock.index) + 1
    nextTimestamp = timestamp
    nextHash = calculateHash(nextIndex, previousBlock.currentHash, nextTimestamp, proof, blockData)
    return Block(nextIndex, previousBlock.currentHash, nextTimestamp, blockData, proof, nextHash)


```

# Blockchain – démo (2)

Blockchain.py

```
def writeBlockchain(blockchain):
    blockchainList = []
    for block in blockchain:
        blockList = [block.index, block.previousHash, str(block.timestamp), block.data, block.proof, block.currentHash]
        blockchainList.append(blockList)

    with open("blockchain.csv", "w") as file:
        writer = csv.writer(file)
        writer.writerows(blockchainList)
    print('Blockchain written to blockchain.csv.')


def readBlockchain(blockchainFilePath):
    importedBlockchain = []
    try:
        with open(blockchainFilePath, 'r') as file:
            blockReader = csv.reader(file)
            for line in blockReader:
                block = Block(line[0], line[1], line[2], line[3], line[4], line[5])
                importedBlockchain.append(block)
        print("Pulling blockchain from csv...")
        return importedBlockchain
    except:
        print('No blockchain located. Generating new genesis block...')
        return [getGenesisBlock()]


def getTxData():
    txData = ''
    for _ in range(5):
        txTo, txFrom, amount = random.randrange(0, 1000), random.randrange(0, 1000), random.randrange(0, 100)
        transaction = 'User ' + str(txFrom) + " sent " + str(amount) + ' tokens to user ' + str(txTo) + ". "
        txData += transaction
    return txData
```

# Blockchain – démo (3)

```
80
81
82  def mineNewBlock(difficulty=5, blockchainPath='blockchain.csv'):
83      blockchain = readBlockchain(blockchainPath)
84      txData = getTxData()
85      timestamp = time.time()
86      proof = 0
87      newBlockFound = False
88      print('Mining a block...')
89      while not newBlockFound:
90          # print("Trying new block proof...")
91          newBlockAttempt = generateNextBlock(blockchain, txData, timestamp, proof)
92          if newBlockAttempt.currentHash[0:difficulty] == '0' * difficulty:
93              stopTime = time.time()
94              timer = stopTime - timestamp
95              print('New block found with proof', proof, 'in', round(timer, 2), 'seconds.')
96
97              newBlockFound = True
98          else:
99              proof += 1
100     blockchain.append(newBlockAttempt)
101     writeBlockchain(blockchain)
102
103
104 def mine(blocksToMine=5):
105     for _ in range(blocksToMine):
106         mineNewBlock()
107
108 blockchain = [getGenesisBlock()]
109
```

# Divers

# Divers

- Debugging avec PyCharm
- Cohabitation Interprêteurs Python 2 et Python 3
- Passage de paramètres à un programme
- Contrôle d'accès au code: fichier *__init__.py*

# Résumé et QA

# Reference

- https://docs.python.org/3/
- https://docs.python.org/3/tutorial/index.html
- https://docs.python.org/3/howto/
- *https://inforef.be/swi/download/apprendre_python3_5.pdf*
- *https://www.**tutorialspoint**.com/**python3**/*
- *https://matplotlib.org/tutorials/index.html#introductory*
- *https://www.programcreek.com/python/example/4890/matplotlib*

# Merci