Calcul Numérique en Mécanique
Python

R. Mozul & L. Daridon

Université de Montpellier - CNRS

2021

Sommaire

Introduction

- Créé par Guido Von Rossum en 1991
- Multi-plateforme, libre et gratuit
- Géré par la Python Software Foundation
- Nombreuses contributions de la communauté

Affectation et type

- Syntaxe variable = valeur
- Le membre de droite est évalué avant celui de gauche

Affectation et type

- Syntaxe variable = valeur
- Le membre de droite est évalué avant celui de gauche
- variable peut contenir lettres, chiffres et _

Affectation et type

- Syntaxe variable = valeur
- Le membre de droite est évalué avant celui de gauche
- variable peut contenir lettres, chiffres et _
- variable ne peut pas commencer par un chiffre

Affectation et type

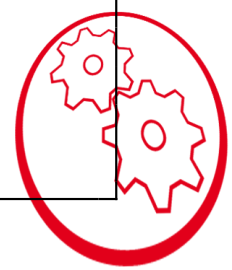
- Syntaxe variable = valeur
- Le membre de droite est évalué avant celui de gauche
- variable peut contenir lettres, chiffres et _
- variable ne peut pas commencer par un chiffre
- variable n'est pas typée

Affectation et type

- Syntaxe variable = valeur
- Le membre de droite est évalué avant celui de gauche
- variable peut contenir lettres, chiffres et _
- variable ne peut pas commencer par un chiffre
- variable n'est pas typée
- valeur est typée

Affectation et type

```
>>> a = 12
>>> a_12_b = a
>>> a = 12
>>> _12A = 12.
>>> 12a = 12
  File "<stdin>", line 1
    12a = 12
     ^
SyntaxError: invalid syntax
>>> a = 'totor'
>>> a = 12
>>> b = 12.
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
>>> isinstance(a,int)
True
>>> isinstance(b,int)
False
>>> a == b
True
>>> a is b
False
```



Type et opérateur

- Types numériques : int float complex
- Opérations classiques + - * / **

Type et opérateur

- Types numériques : int float complex
- Opérations classiques + - * / **
- Opérations courante // %

Type et opérateur

- Types numériques : int float complex
- Opérations classiques + - * / **
- Opérations courante // %
- Opérations booléennes not or and in

Type et opérateur

- Types numériques : int float complex
- Opérations classiques + - * / **
- Opérations courante // %
- Opérations booléennes not or and in
- Comparaisons == ! = < > <= >=

Type et opérateur

- Types numériques : int float complex
- Opérations classiques + - * / **
- Opérations courante // %
- Opérations booléennes not or and in
- Comparaisons == ! = < > <= >=
- Utilisation des parenthèses pour contrôler les priorités

Type et opérateur

- Types numériques : int float complex
- Opérations classiques + - * / **
- Opérations courante // %
- Opérations booléennes not or and in
- Comparaisons == ! = < > <= >=
- Utilisation des parenthèses pour contrôler les priorités
- L'expression est évaluée de gauche à droite

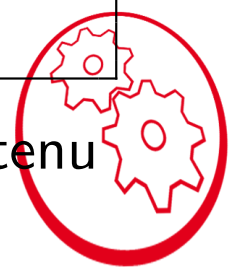
Type et opérateur

```
>>> c = 2*a**2 + 3
>>> c//a
 24
>>> c%a
 3
>>> d = (c+5) * b
>>> d < a
 False
>>> (d < a ) and ( a == b )
 False
```

Structure conditionnelle

- Exécute des instructions en fonction de la valeur d'un test
- Syntaxe avec le mot clef if test :
- Le bloc d'instructions suivant doit commencer par des espaces
- En général 4 espaces
- Le nombre d'espaces doit être constant pour toute les lignes du bloc

```
>>> if c%5 == 0:
...     print(c," est divisible par 5")
... elif c%5 == 1:
...     print(c,"-1 est divisible par 5... ")
... else:
...     print(c," nous fait suer !")
```



Exercice : se donner deux variables a et b et échanger leur contenu si $a > b$

Structure répétitive

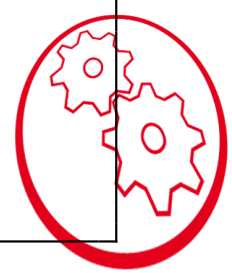
Pour répéter un bloc d'instructions il faut faire des "boucles"

- while, tant qu'une condition est vérifiée
- for, un certain nombre de fois

La boucle for va de paire avec les séquences (en fait avec les itérateurs).

- break permet de quitter la boucle en cours
- continue permet de passer à l'itération suivante
- range(n) permet de générer la séquence des n premiers entiers naturels

```
n = 12
for i in range(n):
    print(i)

s = range(n//2)
for k in s:
    if k%2 == 0:
        print(f"{k} est pair")
```



Exercice : Trouver si un nombre n est premier

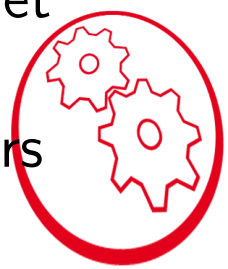
Fonctions et portées des variables

- Réutiliser des blocs d'instructions de manière paramétrées
- Attention à la portée des variables

```
def func(args):
    c = args+args
    return c

val = func('bla')

def f2(args):
    return args+val

val = f2('blu')
```

Exercice : Faire une fonction qui test si un nombre est premier et la tester pour plusieurs valeurs

Exercice : Essayer de faire une fonction qui échange deux valeurs

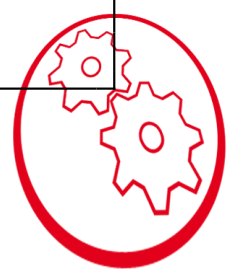


Séquences

Séquence indexées par des entiers naturels:

- les chaînes de caractères du type str
- les listes du type list
- les objets de la classe tuple ($\sim$ liste figée)

```
s1 = "totor"
s2 = 'Supercalifragilisticexpialidocious'
for c in s2:
  print(c)

for i in range(len(s1)):
  print(i, s1[i])

l1 = [1, 3, 5]
l2 = []
l2.append(13)
l2.extend(l1)
l2[2] = 3.14
print(l2)
```



Remarque : les listes et tuples peuvent contenir des données hétérogènes

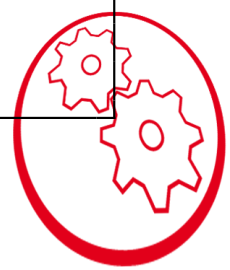
Exercice : Ecrire une fonction qui trie une liste d'entiers

Séquences

Les dictionnaires sont

- des listes
- non ordonnées
- indéxées par autre chose qu'une liste d'entiers consécutifs

```
annuaire = {1234:'Remy' ,
            6222:'Loic',
           }
annuaire[2] = 'Loic'
for i in annuaire:
  print(i)

for n in annuaire.values():
  print(n)

annuaire[1234] = 3.14
```

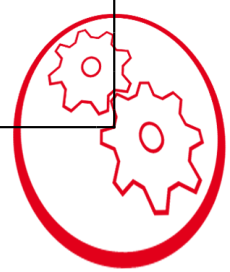


Notions d'objets

Les objets/instances sont :

- d'un type complexe appelé une classe
- souvent composés d'autres objets appelés attributs
- fournit des fonctions appelées méthodes qui travaillent sur ses attributs
- l'opérateur de résolution est .

```
>>> l = [i for i in range(12)]
>>> type(l)
<class 'list'>
>>> l.reverse()
>>> print(l)
>>> l.sort()
>>> print(l)
```



Importation de Modules

Un module renferme la définition:

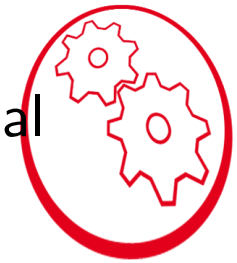
- de classe
- de variables
- de fonctions

que l'on souhaite utiliser dans l'espace de nommage courant.

```
import math
print(math.pi)
print(math.sin(math.pi/4.))
import numpy as np
a = np.array([1,2,3,4])
from matplotlib import pyplot as plt
help(plt)
```

Exercice :

- Mettre les fonctions dans tri de liste et test de primalité dans un fichier
- Utiliser ce fichier comme un module dans un script principal qui:
  - Génère une liste d'entier aléatoire
  - Extrait les nombres premiers de cette liste



Fichiers

Il faut 'ouvrir' un fichier pour le récupérer comme un objet.
Le mode d'ouverture peut être:

- en lecture : 'r'
- en écriture : 'w' (en écrasant le contenu précédent)
- en ajout : 'a' (en ajoutant à la fin du fichier)

```
with open('loop.py','r') as f:
    for l in f.readlines():
      print(l,end=''):
```