# Introduction à Python

Pascal Lafourcade

IREM Clermont, 22 mars 2024



IUT CLERMONT AUVERGNE
Aurillac - Clermont-Ferrand - Le Puy-en-Velay
Montluçon - Moulins - Vichy

# Organisation

# Team

- Emmanuel Delay
- Nicolas Desfôrets
- Cédrick Faury
- Pascal Lafourcade
- Malika More

# Programme

- Boucle
- Moyenne
- Médiane
- Palindrome
- Factorielle
- Dichotomie
- Syracuse
- Tremblement de terre : GPX
- Mastermind
- Cryptarithme
- Tkinter

# Planning

## Matin

- 9h00 - 9h30 : Tour de table
- 9h30 -10h00 : Introduction à Python
- 10h00 - 10h45 : Premiers Pas
- 10h45 -11h00 : Pause
- 11h00 - 12h00 : Les incontournables

## Après-midi

- 13h30 - 14h00 : GPX
- 14h00 - 15h00 : Mastermind, Cryptarithme
- 15h00 - 15h15 : Pause
- 15h15 - 16h30 : Tkinter

# Introduction à Pyhton

# Algorithme vs Programme

## Algorithme

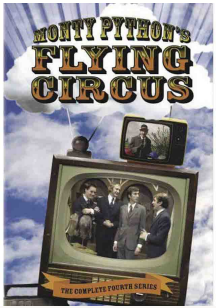
`x` reçoit le double de `y`

## Programme

```
x = y + y
x = 2 * y
```

# Python, 1991

Guido van Rossum




`www.python.org`

GPL-compatible, PSF License Agreement and the Zero-Clause BSD license

Pyhton 2 est déprécié

- Garbage collector
- Langage interprété
- Pas de déclaration
- De nombreuses bibliothèques
- Orienté objet
- Pas très rapide
- Bon pour prototyper

# Types

- Entiers : `int` — 42
- Réels : `float` — 3.14 ou 1/3
- Booléens : `bool` — `True` ou `False`
- Chaînes de caractères : `string` — "Pascal"

```
1 rayon = 3 + 2 * 5
2 aire = 3.14 * rayon ** 2
3 Chaine = "Cette" + " chaine" + " est" + " bien" + " trop" + " longue" + " pour" \
4     + " tenir" + " sur" + " une" + " ligne."
```

Surcharge de l'opérateur +

`chr(97)` vaut `'a'` et `ord('a')` vaut 97

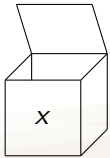
# Opérateurs

- `a+b`, `a - b`, `a * b`, `a / b`, `a ** b`,
- `not a`, `a and b`, `a or b`.
- Si $a = b.q + r$ avec $0 \leq r < b$, alors $q =$ `a // b`, et $r =$ `a % b` ou $r =$ `divmod(a,b)`.
- Conversion :

```
x = int(y)
y = str(x)
```

# Notion de variable

- Une variable désigne un **emplacement** de la **mémoire** qui porte un **nom**.
- Contient une **valeur**.
- Son contenu peut changer durant l'exécution d'un programme.



# Choix des noms des variables :

## En mathématiques

- $n, p, q$ pour des entiers
- $i, j, k$ pour des indices
- $x$ pour un réel
- $z$ pour un complexe
- $f, g, h$ pour des fonctions
- ...

# Choix des noms des variables :

## En mathématiques

- $n, p, q$ pour des entiers
- $i, j, k$ pour des indices
- $x$ pour un réel
- $z$ pour un complexe
- $f, g, h$ pour des fonctions
- ...

Il n'y a que 26 lettres !

# Choix des noms des variables :

## En mathématiques

- $n, p, q$ pour des entiers
- $i, j, k$ pour des indices
- $x$ pour un réel
- $z$ pour un complexe
- $f, g, h$ pour des fonctions
- ...

Il n'y a que 26 lettres !

$\alpha, \beta \ldots x_1, x_2 \ldots$ encore $\alpha_1, \alpha_2 \ldots$

# Choix des noms des variables :

## En informatique

- *cpt* pour un compteur
- *cote*, *somme*, *resultat*

# Choix des noms des variables :

## En informatique

- *cpt* pour un compteur
- *cote*, *somme*, *resultat*

## Conventions de nommage

- **Par définition** : Pas d'espaces, ni symboles réservés, pas d'accents
- Des "vrais" noms explicites !
- Plus de restriction de taille !
- Attention aux majuscules (CONSTANTES, Classes etc...)

# Égalités ...

= affecte la valeur d'une expression à une variable.

## Quizz

Que vaut `x` après les 2 instructions suivantes ?

```
x = 3
x = x+1
```

# Égalités ...

= affecte la valeur d'une expression à une variable.

## Quizz

Que vaut `x` après les 2 instructions suivantes ?

```
x = 3
x = x+1
```

== effectue une comparaison entre deux expressions.

## Quizz

Que vaut `x` après l'instruction suivante ?

```
x = 3 == 3.14
```

# Fonction

```
1 def double(x):
2     res = x + x
3     return res
```

Quizz : Que fait la fonction `double` sur ces entrées :

```
1 a=3
2 print(double(3))
3 print(double(3.14))
4 print(double('to'))
5 print(double([1, 2]))
6 print(double('3'+'a'))
7 print(double('3'+a))
```

# Quizz : Que fait la fonction `mystere`

```
1 def mystere(x, y):
2     x = x + y
3     y = x - y
4     x = x - y
5     return (x, y)
```

# Quizz : Que fait la fonction `mystere`

```
1  def mystere(x, y):
2      x = x + y
3      y = x - y
4      x = x - y
5      return (x, y)
```

```
1  a = 3
2  b = 4
3  print(mystere(a, b))   # affiche (4, 3)
4  print(a, b)            # affiche  3 4
```

# Quizz : Que fait la fonction `mystere`

```
1  def mystere(x, y):
2      x = x + y
3      y = x - y
4      x = x - y
5      return (x, y)
```

```
1  a = 3
2  b = 4
3  print(mystere(a, b))   # affiche (4, 3)
4  print(a, b)            # affiche 3 4
```

| | x | y |
|---|---|---|
| x ← x + y | a + b | b |
| y ← x - y | a + b | (a + b) - b = a |
| x ← x - y | (a + b) - a = b | a |

# Variables locales vs globales

```
def f(z):
    x = z + a
    return x
x = 3
a = 3
print(f(x))
a = 4
print(f(x))
```

# Variables locales vs globales

```
def f(z):
    x = z + a
    return x
x = 3
a = 3
print(f(x))
a = 4
print(f(x))
```

6 puis 7

## Globale vs locale

a est une variable globale

z est une variable locale

# Que fait ce programme ?

```
def f(z):
    y = x + z
    x = 42
    return y

x = 18
print(f(x));
```

# Que fait ce programme ?

```
def f(z):
    y = x + z
    x = 42
    return y

x = 18
print(f(x));
```

y = x + z UnboundLocalError: local variable 'x' referenced before assignment

Le `x = 42` dit que `x` est locale, mais elle existe déjà

# Variable Globale

```
def myfunc():
    global x
    x = "fantastic"

myfunc()

print("Python is " + x)
```

# Listes

```
1  L = []
2  L.append(4)
3  L.append(5)
4  L.append(6)
5  print(L)                  # [4, 5, 6]
6  L = [4,5,7]
7  print(L)                  # [4, 5, 7]
8  print(L[1])               # 5
9  print(len(L))             # affiche 3, le nombre d'elements de la liste L
10 print(L[-1])              # affiche le dernier element de la liste L
11 print(L[len(L) - 1])      # affiche le dernier element de la liste L
```

# Slicing

```
1 ch = "Bonjour!"
2 print(ch [3:7])          # affiche la chaine "jour"
3 print(ch [1:8:2])        # affiche la chaine "oju!" pas de 2, commence a 1
```

Le pas est optionnel. Si le début n'est pas indiqué alors le début de la liste est considéré par défaut, de même pour la fin.

`S == S[::-1]` teste si S est un palindrome.

La méthode `join()` assembler des chaîne de caractères.

```
1 lst = ["Bon", "jour", "!"]
2 print("".join( lst ))          # affiche "Bonjour!"
```

# Dictionnaire

```
1 d={
2   "Guenaelle": 42,
3   "Pascal": 31
4   }
5 d["Malika"]=63       # ajoute le score de Malika dans le dictionnaire
6 del d["Guenaelle"]   # supprime l'entree Guenaelle du dictionnaire
7 print(d['Pascal'])   # affiche "31"
```

```
1 print(d['Simon'])                          # affiche KeyError: 'Simon'
2 print(d.get('Simon', 'Joueur inconnu'))    # affiche "Joueur inconnu"
```

# Indentation : un rôle crucial !

Attention

Leur présence ou leur absence aux bons endroits peut modifier le comportement d'un programme, voire générer des erreurs !

# Instruction conditionnelle : `if`

```
1 if x > 3 :
2     res = 3
3 elif x == 0 :
4     res = 0
5 else :
6     res = 1
```

# Indentation !

```
1 x = 1
2 res = 5
3 if x > 3 :
4     res = 3
5     res = 0
```

```
1 x = 1
2 res = 5
3 if x > 3 :
4     res = 3
5 res = 0
```

# Boucle `while`

```
1 i=0
2 while (i<10) :
3         print(i)          # 0 1 2 3 4 5 6 7 8 9
4         i=i+1
```

# Boucle `while`

```
1 i=0
2 while (i<10) :
3         print(i)          # 0 1 2 3 4 5 6 7 8 9
4         i=i+1
```

```
1 i=0
2 while (i<10) :
3         print(i)          # Boucle infinie
4 i=i+1
```

# Boucle `for`

```
1 for i in range(10):
2     print(i)          # 0 1 2 3 4 5 6 7 8 9
```

`range(i)` retourne l'ensemble des valeurs entières entre 0 et `i-1`

`range(start, stop[, step])`

# Bibliothèques

```
1 import time
2 import random
3
4 debut = time.time()                    # donne l'heure courante
5 r = random.randint(10,100)    # tire au hasard un entier entre 10 et 100
6 fin = time.time()
7 duree = fin - debut
```

```
python3 -m pip install turtle
```

```
1 import numpy as np
2
3 zeros = np.zeros([10,10])      # construit une matrice de taille 10 x 10 remplie de 0
4 uns = np.ones([10,10])         # construit une matrice de taille 10 x 10 remplie de 1
5 t = np.linspace(3, 4, 11)      # construit une liste de 11 valeurs allant de 3 a 4
6 print(t)                       # [3. 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.]
```

# Lecture dans un fichier

```
1 fichier = open('f.txt','r')
2 for line in fichier :
3     for word in line.split() :
4         print(word)
5 fichier.close()
```

```
1 fichier = open('f.txt','r')
2 caractere = fichier.read(1)
3 while caractere :
4     print(caractere)
5     caractere = fichier.read(1)
6 fichier.close()
```

# Écriture dans un fichier

```
1  import json
2
3  d = {
4    "Guenaelle": 42,
5    "Pascal": 31
6  }
7
8  # ecrire un dictionnaire dans un fichier JSON
9  with open('fichier.txt', mode='w', encoding='utf-8') as fd:
10   json.dump(d, fd)
11
12 # lire un fichier JSON dans un dictionnaire
13 with open('fichier.txt', mode='r', encoding='utf-8') as fd:
14   dictionnaire = json.load(fd)
```

# Erreurs

```
1  une_liste = [0, 1, 2]
2  une_liste[0] = 3
3  print(une_liste)
4  # affiche [3, 1, 2]
5
6  un_n_uplet = (0, 1, 2)
7  un_n_uplet[0] = 3
8  # genere une erreur :
9  # TypeError: 'tuple' object does not support item assignment
10
11 une_chaine = 'chaine de caracteres'
12 une_chaine[0] = 'C'
13 # genere une erreur :
14 # TypeError: 'str' object does not support item assignment
```

# Mutable vs non-mutable

## Objets non-mutables

```
1 # types immuables
2 a = 2          # une reference vers l'objet 2 est assignee pour a
3 b = a          # une reference vers l'objet 2 est assignee pour b
4 a = a + 1      # une reference vers l'objet 3 est assignee pour a,
5                # mais l'objet reference par b n'est pas impacte
6 print(a, b)    # affiche 3, 2
```

## Objets mutables

```
1 # types muables
2 a = [0, 1, 2] # une reference vers l'objet liste [0,1,2] est assignee pour a
3 b = a         # la meme reference est assignee pour a
4 a[0] = 3      # une reference vers l'objet 3 est assignee au premier element de l'objet liste
5 print(a, b)   # affiche [3, 1, 2] [3, 1, 2]
```

# Exemple

```
1 def ajouter_dans_liste( liste , x):
2     liste .append(x)
3
4 L = [1, 2, 3]
5 print(L)                    # affiche [1, 2, 3]
6 ajouter_dans_liste(L, 4)
7 print(L)                    # affiche [1, 2, 3, 4]
```

# Conclusion

# Conclusion

- Pratiquons !

Questions ?

pascal.lafourcade@uca.fr

Merci