# Python

Les bases du langage

# Présentation

- Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. La première version publique de ce langage a été publiée en 1991. Ce langage possède des caractéristiques intéressantes:

  - Il est multiplateforme. Il peut fonctionne sur: Windows, Mac OS X, Linux, Android, iOS, ..
  - Il est gratuit.
  - C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.

# Installation

- Pour installer python, vous pouvez accéder au site officiel de python:

  https://www.python.org/

# Les variables

- Dans un programme informatique, on a besoin souvent de stocker provisoirement en mémoire des valeurs utiles pour un programme. Pour cela on utilise des variables.

- Une variable est un nom qui sert à repérer un emplacement donné de la mémoire, c'est à dire que la variable ce n'est qu'une adresse de mémoire.

# Nom d'une variable

- Le nom (on dit aussi identificateur) d'une variable, peut être formé d'une ou plusieurs lettres, de chiffres et du caractère souligné.
- Le nom d'une variable ne doit pas commencer par un chiffre
- Le nom d'une variable ne doit pas contenir un espace
- Les signes de ponctuation  ne sont pas autorisés.
- Le nom d'une variable ne doit pas être un mot réservé de python.
- Python est sensible à la casse. Test n'est pas la même chose que TEST.

# Type d'une variable et valeur

- Une variable est caractérisée par sa valeur. A un instant donné, une variable ne peut contenir qu'une seule valeur.

- Une variable est caractérisée aussi par son type. Le type d'une variable définit la nature des informations qui seront représentées dans les variables (numériques, caractères..).

# Déclaration d'une variable

- En Python, la déclaration d'une variable et son initialisation (c'est-à-dire la première valeur que l'on va stocker dedans) se font en même temps.

```
x = 2
```

- Dans cette ligne, nous avons déclaré et initialisé la valeur d'une variable.
  - Python est un langage de typage dynamique.

# Les types des variables

- Le type d'une variable correspond à la nature de celle-ci. Les types primitifs :

  - bool : booléen (True ou False)
  - Int (ou integer) : entier
  - float : réel
  - str (ou string): chaîne de caractère .

# Les types des variables

- Exemples:

```
y = 3.14
z = 120
a = "bonjour"
b = 'salut'
Ok=True
```

# Opérateurs arithmétiques (algébriques)

- Soit:

```
a=11
b=4
```

| Python | nom | signification |
|---|---|---|
| a+b | addition | 11+4=15 |
| a-b | soustraction | 11-4=7 |
| a*b | multiplication | 11x4=44 |
| a/b | division | 11/4=2,75 |
| a**b | exposant | $11^4$ =14641 |
| a//b | Division entière | Quotient de la division entière de a par b : 2 |
| a%b | modulo | Reste de la division entière de a par b: 3 |

# Opérateurs arithmitiques

- si vous mélangez les types entiers et floats, le résultat est renvoyé comme un float (car ce type est plus général). Par ailleurs, l'utilisation de parenthèses permet de gérer les priorités.

- L'opérateur / effectue une division celui-ci renvoie systématiquement un float :

```
3 / 4          # 0.75
```

# Opérateurs d'affectation

- Affectation des valeurs à des variables:

| Python | En pseudo-code | Signification |
|---|---|---|
| A=11 | A←11 | A reçoit 11 ou on affecte 11 à A |

- Opérateurs d'affectation composés:

| opérateur | signification |
|---|---|
| A+=2 | A=A+2 |
| A-=2 | A=A-2 |
| A*=2 | A=A*2 |
| A/=2 | A=A/2 |
| A%=2 | A=A%2 |
| A//=2 | A=A//2 |
| A**=2 | A=A**2 |

# Opération sur les chaines de caractères

- Pour les chaînes de caractères, deux opérations sont possibles, l'addition et la multiplication :

```
chaine = "Salut"
chaine + " Python          #'Salut Python'
chaine * 3                 #SalutSalutSalut'
```

- L'opérateur d'addition + concatène (assemble) deux chaînes de caractères.

# Opérateurs de comparaison

- Les opérateurs de comparaison Python sont utilisés pour comparer deux valeurs (nombre ou chaîne de caractères) :

| Opérateur | Nom | Exemple | Résultat |
|---|---|---|---|
| == | Égal à | a == b | Retourne True si a est égal à b |
| != | Non égal à | a != b | Retourne True si a n'est pas égal à b |
| > | Supérieur à | a > b | Retourne True si a est supérieur à b |
| < | Inférieur à | a < b | Retourne True si a est inférieur à b |
| >= | Supérieur ou égal à | a >= b | Retourne True si a est supérieur ou égal à b |
| <= | Inférieur ou égal à | a <= b | Retourne True si a est inférieur ou égal à b |

# Opérateurs logiques

- Les opérateurs logiques Python sont utilisés pour combiner des instructions conditionnelles.

| Opérateur | Exemple | Résultat |
|---|---|---|
| and | a and b | True si les deux a et b sont Trues |
| or | a or b | True si a ou b est True |
| ! | !a | True si a n'est pas True |

# L'affichage

- la fonction print() affiche l'argument qu'on lui passe entre parenthèses et un retour à ligne. Ce retour à ligne supplémentaire est ajouté par défaut.

```
print("Bonjour") ; print("Mohamed")
Bonjour
Mohamed
```

- **Remarque**: Python n'a pas besoin de points-virgules pour terminer les instructions. Les points virgules peuvent être utilisés pour délimiter des instructions si vous souhaitez mettre plusieurs instructions sur la même ligne.

# L'affichage

- La fonction print() peut également afficher le contenu d'une variable quel que soit son type. Par exemple:

```
var = 3
print(var)          #3

Nom="Said"
print(Nom)          #Said
```

# L'affichage

- Il est également possible d'afficher le contenu de plusieurs variables (quel que soit leur type) en les séparant par des virgules :

```
x = 20
nom = "Mohamed"
print(nom, "a", x, "ans")                    #Mohamed a 20 ans
```

- **Remarque**: Python **ajoute un espace** à chaque fois que l'on utilisait le séparateur « , ».

# L'affichage

- Il existe certains caractères d'espacement utiles pour l'affichage.
  - \n: ce caractère provoque un retour à la ligne
  - \t : ce caractère provoque une tabulation
- Un commentaire est inséré dans python avec le caractère #

# L'affichage

- En python on a la possibilité de modifier le séparateur entre chaine et variable, et cela en utilisant le mot clé sep dans la fonction print()

```
>>> x = 20
>>> nom = "Mohamed"
>>> print(nom, "a", x, "ans", sep="-")
Mohamed-a-20-ans
```

# L'affichage

- En python on a la possibilité de modifier le Caractère de fin de l'affichage, et cela en utilisant le mot clé end dans la fonction print()

```
print("Bonjour mes amis", end="  ")
print("Avez-vous cours?", end=" /")
print(« Merci" )

#Bonjour mes amis  Avez-vous cours?/Merci
```

# Les formats d'affichage

- On peut appliquer un format d'affichage en utilisant ce qu'on appel les **f- string**
- **f-string** est le diminutif de formatted string.
- Les f-strings permettent une meilleure organisation de l'affichage des variables. Our cela on commence une chaine par f ou F.

```
x = 20
nom = "Mohamed"
print(f "{nom} a {x} ans")                    #Mohamed a 20 ans
```

- il est possible de mettre entre les accolades des valeurs numériques ou des chaînes de caractères.

```
print( f"J'affiche l'entier {10} et le float {3.14}" )
#J'affiche l'entier 10 et le float 3.14
print(F"J'affiche la chaine {'Python'}")
#J'affiche la chaine Python
```

# Lecture au clavier

- On a la possibilité de d'initialiser la valeur d'une variable, mais on a aussi la possibilité d'inviter l'utilisateur à saisir sa valeur avec la fonction input().

- La fonction input() prend en argument un message (sous la forme d'une chaîne de caractères), demande à l'utilisateur d'entrer une valeur et renvoie celle-ci sous forme d'une chaîne de caractères. Il faut ensuite convertir cette dernière en entier ou float en utilisant les fonction int(), float()

```
reponse = input("Entrez un entier svp: ")
x=int(reponse)
print(x)
Entrez un entier svp: 4
4
```

# La fonction type()

- Si vous ne vous souvenez plus du type d'une variable, utilisez la fonction type() qui vous le rappellera.

```
x = 2
type(x)                    #<class 'int'>

y = 2.0
type(y)                    #<class 'float'>

z = '2'
type(z)                    #<class 'str'>
```

# Conversion de type

- En programmation, on est souvent amené à convertir les types, c'est-à-dire passer d'un type numérique à une chaîne de caractères ou vice-versa. En Python, rien de plus simple avec les fonctions int(), float() et str().

```
i = 3
str(i)                  #'3'

i = '456'
int(i)                  #456

float(i)                #456.0

i = '3.1416'
float(i)                #3.1416
```

- Toute conversion d'une variable d'un type en un autre est appelé casting en anglais.

# Les conditions(1)

- Les tests ou les conditions sont un élément essentiel à tout langage informatique car ils permettent à l'ordinateur de prendre des décisions. Pour cela, Python utilise l'instruction if

# Les conditions (2)

- Exemple: condition à un seul cas.

```
x = 2
if x == 2 :
        print("Le test est vrai !")
```

- la ligne qui contient l'instruction if se termine par le caractère deux-points « : ».

- Les blocs d'instructions dans les tests doivent forcément être indentés(un espace à gauche de 4 caractères) .

# Les conditions(3)

- Exemple :test à deux cas

```
x = 3
if x == 2:
    print("Le test est vrai !")
else:
    print("Le test est faux !")
```

- La condition étant fausse lé résultat sera : Le test est **faux.**

# Les conditions(4)

- Les instructions if/elif/else vous permettent de définir des conditions multiples. Le mot-clé elif vous permet d'ajouter autant de conditions que vous voulez. Vous devez ensuite terminer avec une instruction else.
- Exemple:

```
x = 20
if x>=20 :
      print("supérieur à 20")
elif x<20 and x>=10 :
       print("entre 10 et 20")
elif x<10 and x>=0 :
       print("entre 0 et 10")
else:
         print("négatif")
```

# Les conditions(5)

- La condition dans l'instruction if peut comporter des opérateurs logiques. En python on utilise and pour le ET et or pour le OU. Ces opérateurs s'écrivent en python en minuscule.

- Exemple:

```
x = 2
y = 2
if x == 2 and y == 2:
    print("le test est vrai")
```

# Les conditions(6)

- Notez que le même résultat de l'exemple précédent serait obtenu en utilisant deux instructions if **imbriquées** :

- Exemple:

```
x = 2
y = 2
if x == 2:
    if y == 2:
        print("le test est vrai")
```