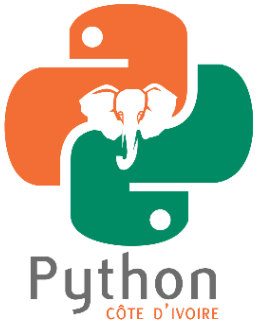
J'apprends à programmer avec Python

Initiation à la programmation Python



Zokora Elvis

degbagnon@gmail.com @ElvisZokora

- Ingénieur Génie Logiciel / Sys et Sécurité
- Consultant en systèmes d'informations
- Amoureux du langage Python
- Président de Python CI

Objectifs pédagogiques

- Connaitre Python et savoir ce qu'il fait
- Ecrire des scripts/Applications avec Python
- Donner les clés pour:
  1. Approfondir rapidement vos connaissances en toute autonomie
  2. Concevoir rapidement de vraies applications utiles

Programme de formation

1. Python c'est quoi ?
2. Variables
3. Type de variable
4. Opération de base
5. Fonctions de bases
6. Conditions et boucles
7. Listes et Tuples
8. Dictionnaires et Fonctions
9. Modules
10. Fichiers
11. La POO

C’est quoi Python ?

Un langage de programmation
Ecrit par programmeur Guido van Rossum développeur Néerlandais
https://gvanrossum.github.io//
Le langage de programmation Python a été créé en 1989 aux Pays-Bas.
Le nom Python vient d'un
hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan.

La dernière version de Python est la version 3

Caracteristiques du langage Python

- Multi-plateformes
- Interprété
- Open source , Gratuit
- Très utilisé dans l'industrie et le monde académique
- Orienté objet
- De haut niveau
- À typage dynamique
- Relativement simple à prendre en main

Python pour quoi ?

- Un langage facile à apprendre !
- Une syntaxe légère, efficace et agréable à lire
- Une syntaxe discrète, code court et facile à lire

On peut presque tout faire avec Python

- Interfaces graphiques
- Des applications Web
- Des Applications Mobiles
- Calcul scientifique (algèbre, analyse, statistiques/probas, théorie des graphs, visualisation, etc.)
- calcul formel: calculer la dérivée ou l'intégrale d'une fonction, résoudre une équation, factorisation, calcul de limites, résolution de systèmes d'équations, ...

- informatique haute performance / calcul massivement parallèle
- chimie, biologie, astronomie, psychologie, finance, etc.
- optimisation, machine Learning , traitement du signal, ...
- Musique
- 3D
- Jeux
- Traitement d'image
- systèmes embarqués
- bases de données
- Cartographie
- Réseau .....

Python est lisible

```
class CompteBancaire(object):
    "Gestion d'un compte bancaire"

    #Constructeur de la classe
    def __init__(self,nom="Dupont",solde=1000):
        self.nom=nom
        self.solde=solde

    #Methode permettant de gerer les depots
    def depot(self,montant):
        self.solde=self.solde+montant


    #Methode permettant de gerer les retrait sur le compte Bancaire
    def retrait(self,montant):
        self.solde=self.solde-montant


    #Methode permettant d'afficher le nom du Titulaire du compte et le solde du compte
    def affiche(self):
        print("le solde du compte bancaire de ",self.nom,"est de ",self.solde)
```

Exemple de code
Python:

```
l = [2, 3, 5]
for n in l:
    print(n)
```

L'équivalent en C++ :

```
#include <vector>
#include <iostream>
int main() {
    std::vector<int> v {2, 3, 5};    // Ne
marche pas pour C++98
    std::vector<int>::iterator it;
    for(it = v.begin() ; it != v.end() ;
it++) {
        std::cout << *it << std::endl;
    }
    return 0;
}
```

L'équivalent en Java:

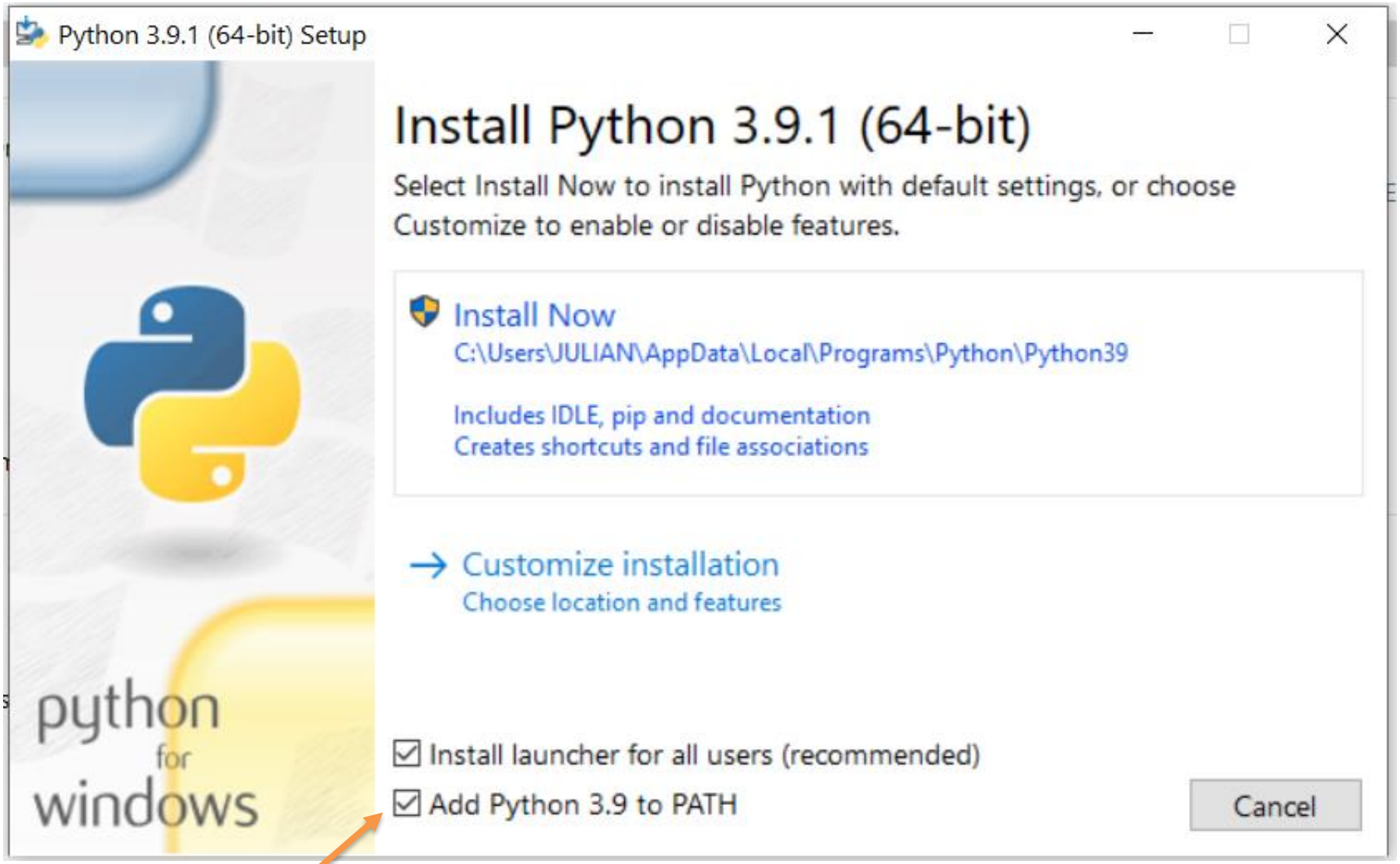
```
import java.util.ArrayList;
import java.util.Arrays;
public class Test {
    public static void main(String[]
args) {
        ArrayList<Integer> list =
new
ArrayList<Integer>(Arrays.asList(
2, 3, 5));
        for(int val : list) {
            System.out.println(val);
        }
    }
}
```

il n'y a pas que les petits qui l'utilisent

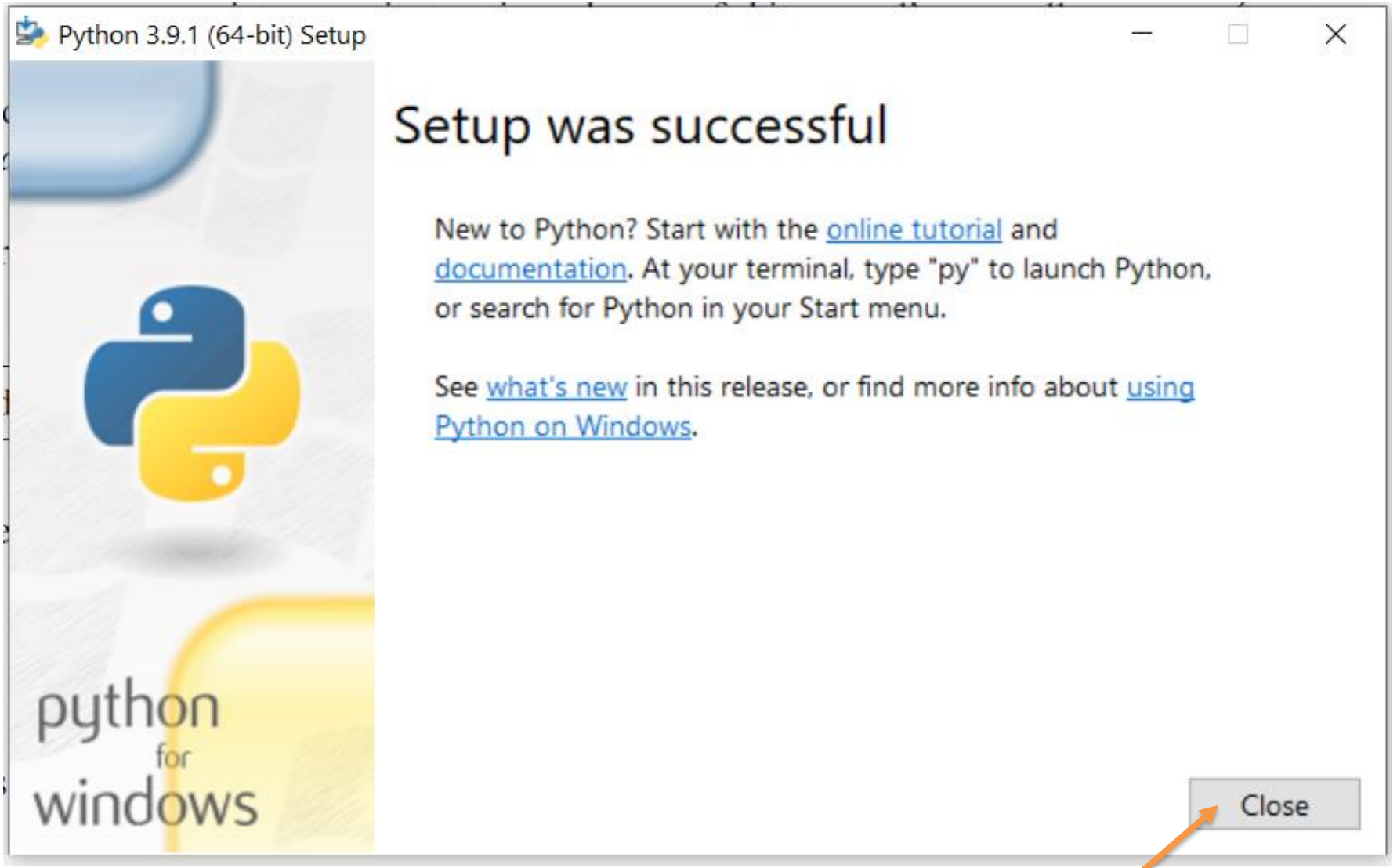
Installation des Outils et présentation de l'environnement de programmation

Installation de Python : la méthode classique...

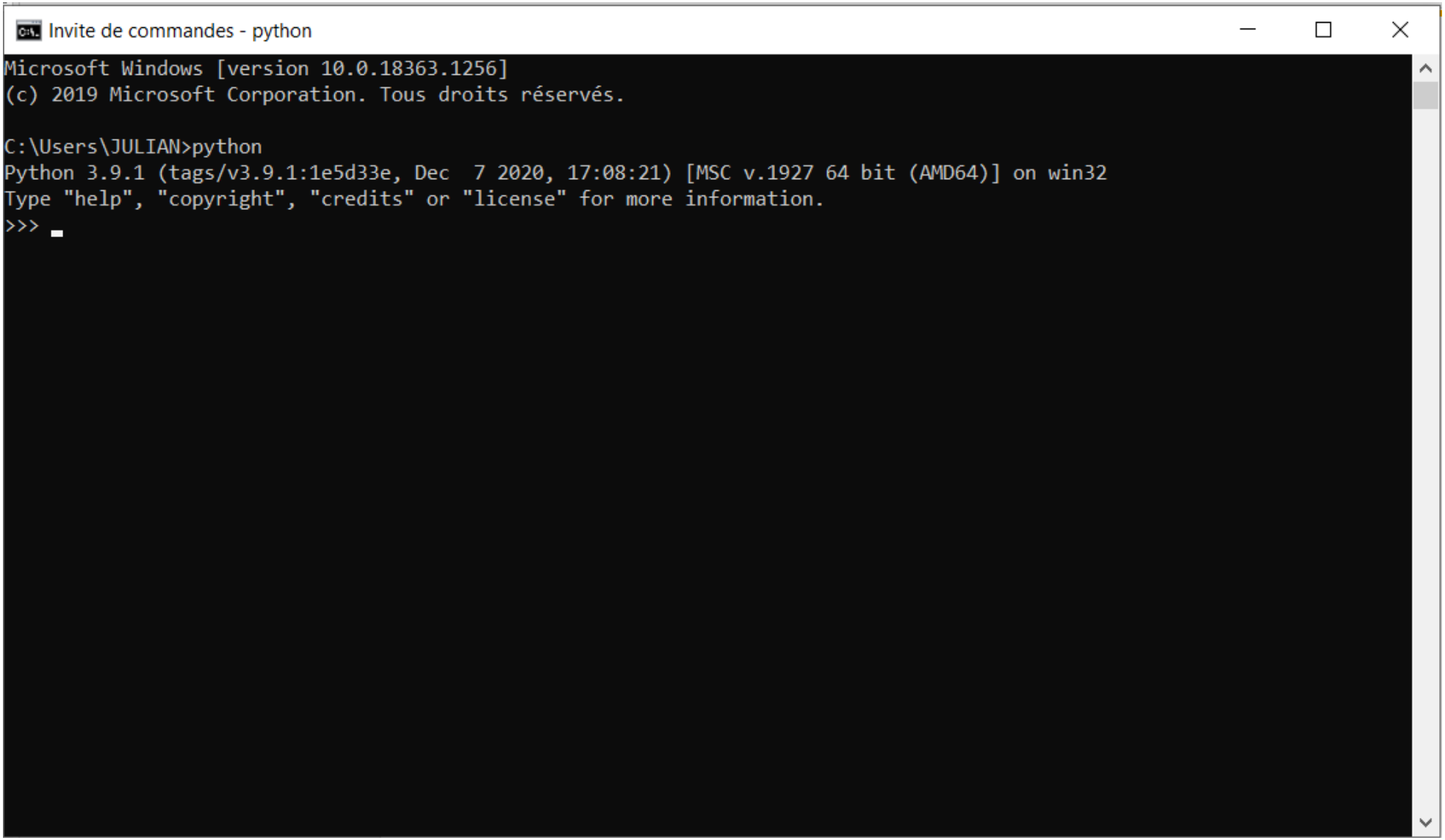
www.python.org


Python 3.9.1 (64-bit) Setup
Install Python 3.9.1 (64-bit)
Select Install Now to install Python with default settings, or choose
Customize to enable or disable features.
Install Now
C:\Users\JULIAN\AppData\Local\Programs\Python\Python39
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python
for
windows
Install launcher for all users (recommended)
Add Python 3.9 to PATH
Cancel

À Cocher



Terminer l'installation


Invite de commandes - python
Microsoft Windows [version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. Tous droits réservés.
C:\Users\JULIAN>python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

L'environnement de développement IDLE

IDLE

- L'IDE "officiel" de Python
- Intégré de base avec Python (standard): rien à installer
- Écrit en Python
- Open source

Pourquoi IDLE ?

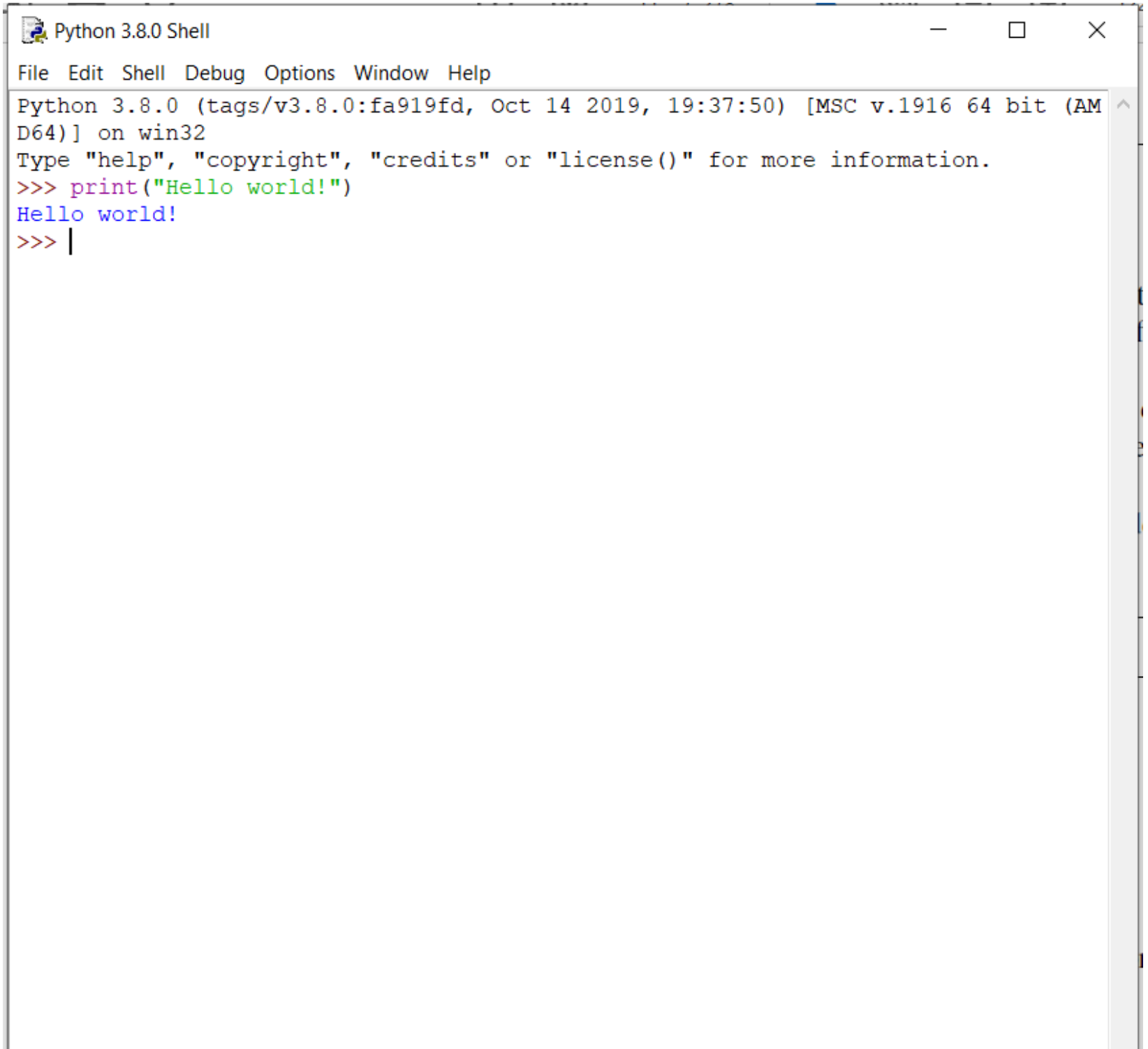
- Un classique
- Intégré de base avec Python: si vous avez Python, vous avez IDLE (sauf cas particuliers)
- Fonctionne sur Windows, MacOSX, Linux, ...
- Fonctionnalités basiques mais adaptées aux besoins de cette formation
- Ni trop simple, ni trop compliqué
- L'interprète python
- Ouvrir l'interprète en mode interactif

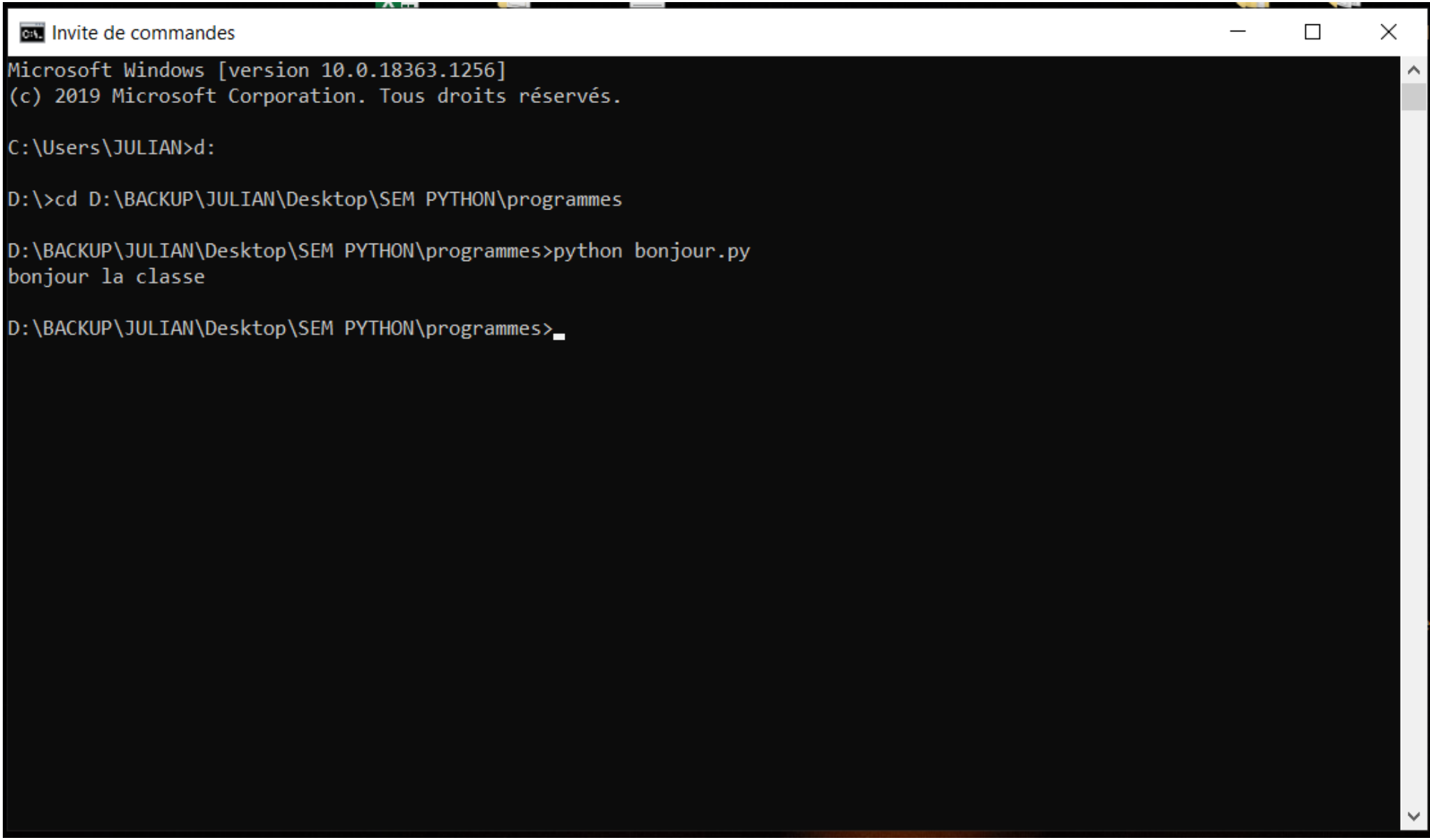
Les alternatives à IDLE

Si IDLE ne vous plaît pas il y a :

PyCharm
Spyder
Et les autres: Notepad++,Vscode , Code::Blocks, Eclipse, ...

Premier Programme


Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello world!")
Hello world!
>>>


Invite de commandes
Microsoft Windows [version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. Tous droits réservés.
C:\Users\JULIAN>d:
D:\>cd D:\BACKUP\JULIAN\Desktop\SEM PYTHON\programmes
D:\BACKUP\JULIAN\Desktop\SEM PYTHON\programmes>python bonjour.py
bonjour la classe
D:\BACKUP\JULIAN\Desktop\SEM PYTHON\programmes>

Variables et types de base

Qu'est-ce qu'une variable ?

C'est un espace de stockage dans la mémoire de l'ordinateur (une "case") caractérisé par:

- Un nom
- Une valeur
- Un type
- Une portée

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.

Règles de nommage

Avec Python, les noms de variables doivent en outre obéir à quelques règles simples :

- Un nom de variable est une séquence de lettres (a → z , A → Z) et de chiffres (0 → 9), qui doit toujours commencer par une lettre.
- Seules les lettres ordinaires sont autorisées. Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que $, #, @, etc. sont interdits, à l'exception du caractère _ (souligné).
- La casse est significative (les caractères majuscules et minuscules sont distingués)
- De plus, il faut absolument éviter d'utiliser un mot « réservé » par Python comme nom de variable (par exemple : print, range, for, from, etc.).

Exemples de noms corrects:

x = 3

AbC = 3

x8 = 3

une_variable = 3

Exemples de noms incorrects:

une variable = 7

supervariable! = 19

8x = 19

une-variable = 3

Qu'est ce qu'une affectation ?

une affectation, aussi appelée assignation par anglicisme, est une structure qui permet d'attribuer une valeur à une variable.
En Python comme dans de nombreux autres langages, l'opération d'affectation est représentée par le signe égale =

Exemple:
x = 3
y = 18

- x est le nom de la 1ère variable et 3 sa valeur
- y est le nom de la 2e variable et 18 sa valeur

Le type d'une variable correspond à la nature de celle-ci.
Les trois principaux types dont nous aurons besoin dans un premier temps sont les entiers (integer ou int), les nombres décimaux que nous appellerons floats et les chaînes de caractères (string ou str). Bien sûr, il existe de nombreux autres types (par exemple, les booléens, les nombres complexes, etc,,,).

Integer : 10 , 20 , 60
Float : 12.2 , 85.6 , 13.00
String : « Bonjour » , « alerte »

Les Commentaires

Qu'est ce qu'une affectation ?

Dans un script, tout ce qui suit le caractère # est ignoré par Python jusqu'à la fin de la ligne et est considéré comme un commentaire.

Les commentaires doivent expliquer votre code dans un langage humain.

Voici un exemple :

```
# Votre premier commentaire en Python .
print (' Hello world !')
# D' autres commandes plus utiles pourraient suivre .
```

Les opérations

Sur les types numériques

\+ : Addition
\- : Soustraction
* : Multiplication
/ : Division
** : l'exposant (Puissance)
// : Division Entière
% : Modulo

Les symboles +, -, *, /, **, // et % sont appelés opérateurs, car ils réalisent des opérations sur les variables.

Sur les types numériques

Enfin, il existe des opérateurs « combinés » qui effectue une opération et une affectation en une seule étape :

```
>>> i=0
>>> i=i+1
>>> i
1
>>> i +=1
>>> i
2
>>> i +=2
>>> i
4
>>>
```

L'opérateur += effectue une addition puis affecte le résultat à la même variable. Cette opération s'appelle une « incrémentation ».

Les opérateurs -=, *= et /= se comportent de manière similaire pour la soustraction, la multiplication et la division

Opérations sur les chaînes de caractères

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

```
>>> chaine="bonjour"
>>> chaine
'bonjour'
>>> chaine + "Python"
'bonjourPython'
>>> chaine * 3
'bonjourbonjourbonjour'
>>>
```

L'opérateur d'addition + concatène (assemble) deux chaînes de caractères.
L'opérateur de multiplication * entre un nombre entier et une chaîne de caractères duplique (répète) plusieurs fois une chaîne de caractères.

Les fonctions de bases

C’est quoi une fonction ?

Une fonction est un sous-programme qui à partir de Données produit un résultat.

Pourquoi utiliser des fonctions ?

- Eviter la répétition
- Rendre le code agréable à lire et bien structuré

Il existe deux type de fonction :

- Originale
- Prédéfinies

Afficher la valeur d'une variable: la fonction print()

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

```
var = "bonjour"
print(var)
```

Obtenir une valeur saisie par un utilisateur : la fonction input()

```
x = input()
print(x)
```

Attention à la fonction input() elle retourne toujours un type string .....

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=0.2
>>> 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().

```
>>> x=3
>>> b=5
>>> type(x)
<class 'int'>
>>> type(b)
<class 'int'>
>>> x=str(x)
>>> b=str(b)
>>> type(x)
<class 'str'>
>>> type(b)
<class 'str'>
>>> x+b
'35'
```

Exercices

Prédire le résultat : opérations

Essayez de prédire le résultat de chacune des instructions suivantes, puis vérifiez-le dans l'interpréteur Python :

— (1+2)**3

— "Da" * 4

— "Da" + 3

— ("Pa"+"La") * 2

— ("Da"*4) / 2

— 5 / 2

— 5 // 2

— 5 % 2

Prédire le résultat : opérations et conversions de types

Essayez de prédire le résultat de chacune des instructions suivantes, puis vérifiez-le dans l'interpréteur Python :

— str(4) * int("3")

— int("3") + float("3.2")

— str(3) * float("3.2")

— str(3/4) * 2

Écrire dans un fichier Python un programme qui:

Affiche un message pour demander à l'utilisateur de saisir un nombre

Récupère le nombre saisi

Affiche sa valeur multipliée par 2

Les structures de contrôle

Opérateurs de comparaisons

Python est capable d'effectuer toute une série de comparaisons entre le contenu de deux variables, telles que :

```
x == y          # x est égal à y
x != y          # x est différent de y
x > y           # x est plus grand que y
x < y           # x est plus petit que y
x >= y          # x est plus grand que, ou égal à y
x <= y          # x est plus petit que, ou égal à y
```

```
>>>
>>> x=5
>>> y=10
>>>
>>> x==b
False
>>> x!=b
True
>>> x>b
False
>>> x<b
True
>>> x>=b
False
>>> x<=b
True
>>>
```

If/Else

Qu'est-ce que c'est et à quoi ça sert ?

Exécuter ou non certaines instructions suivant une condition

Comment ça s'écrit ?

```
if condition :
    # 1ère instruction executée si var est vraie (ie si "var==True")
    # 2e instruction executée si var est vraie (ie si "var==True")
    # 3e instruction executée si var est vraie (ie si "var==True")
    # ...
else:
    # 1ère instruction executée si var est faux (ie si "var==False")
    # 2e instruction executée si var est faux (ie si "var==False")
    # 3e instruction executée si var est faux (ie si "var==False")
```

Exemple

```
>>>
>>> x=3
>>> if x>0 :
        print(x, "est positif")
else :
        print(x, "est negatif")


3 est positif
>>>
```

La partie else n'est pas obligatoire mais recommandée

On peut très bien écrire

```
x = 3
if x > 0:
    print("x est positif")
```

Exemple sans else

```
print("Quel est la capitale de la France ?")
reponse = input()
if reponse != "Paris":
    print("Perdu!")
```

Et avec else

```
print("Quel est la capitale de la France  ?")
reponse = input()
if reponse != "Paris":
    print("Perdu!")
else:
    print("Bravo!")
```

Les if/else imbriqués :

```
1
2
3  print("Quel est ta note sur 20 ?")
4  note = input()
5  if note > 15:
6      print("Très bien!")
7  else:
8      if note < 10:
9          print("Tu feras mieux la prochaine fois")
0      else:
1          print("Pas mal...")
2
```

Les if/else imbriqués :

```
1
2
3  print("Quel est ta note sur 20 ?")
4  note = input()
5  note =int(note)
6  if note > 15:
7      print("Très bien!")
8  else:
9      if note < 10:
10         print("Tu feras mieux la prochaine fois")
11     else:
12         print("Pas mal...")
13
```

Qu'on peut aussi écrire comme ça:

```
1  print("Quel est ta note sur 20 ?")
2  note = input()
3  note =int(note)
4  if note > 15:
5      print("Très bien!")
6  elif note < 10:
7      print("Tu feras mieux la prochaine fois")
8  else:
9      print("Pas mal...")
10
```

Test Multiple :

Les tests multiples permettent de tester plusieurs conditions en même temps en utilisant des opérateurs booléens. Les deux opérateurs les plus couramment utilisés sont le OU et le ET.
Voici un petit rappel sur le fonctionnement de l'opérateur OU :

Condition 1	Opérateur	Condition 2	Résultat
Vrai	OU	Vrai	Vrai
Vrai	OU	Faux	Vrai
Faux	OU	Vrai	Vrai
Faux	OU	Faux	Faux

et de l'opérateur ET :

Condition 1	Opérateur	Condition 2	Résultat
Vrai	ET	Vrai	Vrai
Vrai	ET	Faux	Faux
Faux	ET	Vrai	Faux
Faux	ET	Faux	Faux

Exercices

Notes et mention d'un étudiant

Voici les notes d'un étudiant : 14, 9, 13, 15 et 12. Créez un script qui affiche la note maximum,la note minimum et qui calcule la moyenne.

Affichez la valeur de la moyenne. Affichez aussi la mention obtenue sachant que la mention est «

passable » si la moyenne est entre 10 inclus et 12 exclus, « assez bien » entre 12 inclus et 14 exclus et « bien » au-delà de 14.

Les structures répétitives

Qu'est-ce que c'est et à quoi ça sert ?

En programmation, on appelle boucle un système d'instructions qui permet de répéter un certain nombre de fois (voire indéfiniment) toute une série d'opérations. Python propose deux instructions particulières pour construire des boucles : l'instruction for ... in ... , très puissante, que nous étudierons plus tard , et l'instruction while que nous allons découvrir tout de suite.

Comment ça s'écrit ?

```
while condition :
    # 1ère instruction executée si condition Varie
```

Dès que Condition devient False (var == False), on sort de la boucle

```
1 a = 0
2 while (a < 7): # (n'oubliez pas le double point !)
3     a = a + 1  # (n'oubliez pas l'indentation !)
4     print(a)
5
```

Avec l'instruction while, Python commence par évaluer la validité de la condition fournie entre

parenthèses (celles-ci sont optionnelles, nous ne les avons utilisées que pour clarifier notre

explication).

- Si la condition se révèle fausse, alors tout le bloc qui suit est ignoré et l'exécution du programme

se termine14.

- Si la codition est vraie, alors Python exécute tout le bloc d'instructions constituant le corps de la

boucle, c'est-à-dire :

– l'instruction a = a + 1 qui incrémente d'une unité le contenu de la variable a (ce qui signifie que

l'on affecte à la variable a une nouvelle valeur, qui est égale à la valeur précédente augmentée

d'une unité).

– l'appel de la fonction print() pour afficher la valeur courante de la variable a.

• lorsque ces deux instructions ont été exécutées, nous avons assisté à une première itération, et le

programme boucle, c'est-à-dire que l'exécution reprend à la ligne contenant l'instruction while.

La condition qui s'y trouve est à nouveau évaluée, et ainsi de suite.

Dans notre exemple, si la condition a < 7 est encore vraie, le corps de la boucle est exécuté une

nouvelle fois et le bouclage se poursuit.

Exercices

Ecrire un programme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

7 x 0 = 0

7 x 1 = 7

……

7 x 20=140

Écrire un programme qui affiche les 20 premiers termes de la table de multiplication par 7, en signalant au passage (à l'aide d'un astérisque) ceux qui sont des multiples de 3.

Exemple : 7 14 21 * 28 35 42 * 49 ...

Les listes python

C’est quoi une liste ?

Une liste est structure de données qui contient une série de valeurs .

Déclaration d’une liste :

```
maliste= []     #Déclaration d’une liste de nom « maliste »

print(type(maliste))  #Afficher le type de la variable maliste
```

Comment insère t'on des informations dans une liste?

Méthode 1 :

>>> jour = ['lundi', 'mardi', 'mercredi', 1800, 20.357, 'jeudi', 'vendredi']

>>> print(jour)

Dans cet exemple, la valeur de la variable jour est une liste.

Comme on peut le constater dans le même exemple, les éléments individuels qui constituent une liste

peuvent être de types variés. Dans cet exemple, en effet, les trois premiers éléments sont des chaînes

de caractères, le quatrième élément est un entier, le cinquième un réel, etc.

Comment insère t'on des informations dans une liste?

Methode 2 :

avec la méthode append (qui signifie "ajouter" en anglais):

```
>>> jour = []
>>> jour
>>> jour.append("lundi")
>>> jour
```

Opération sur les listes :

Tout comme les chaînes de caractères, les listes supportent l'opérateur + de concaténation, ainsi que l'opérateur * pour la duplication :

```
>>> a=["bonjour",15,96,45,75,12,45]
>>> b=["Janvier","Février",75,45,56]
>>>
>>> c=a+b
>>> c
['bonjour', 15, 96, 45, 75, 12, 45, 'Janvier', 'Février', 75, 45, 56]
>>> d=a*3
>>> d
['bonjour', 15, 96, 45, 75, 12, 45, 'bonjour', 15, 96, 45, 75, 12, 45, 'bonjour'
, 15, 96, 45, 75, 12, 45]
>>> e=b*5
>>> e
['Janvier', 'Février', 75, 45, 56, 'Janvier', 'Février', 75, 45, 56, 'Janvier',
'Février', 75, 45, 56, 'Janvier', 'Février', 75, 45, 56, 'Janvier', 'Février', 7
5, 45, 56]
>>>
```

Comment Afficher un item(élément) d'une liste?

Pour lire une liste, on peut demander à voir l'index de la valeur qui nous intéresse:

```
>>> jour = ['lundi', 'mardi', 'mercredi', 1800, 20.357, 'jeudi', 'vendredi']
>>> jour[0]
>>> jour[1]
>>> jour[2]
>>> jour[3]
>>> jour[4]
```

Le premier item commence toujours avec l'index 0. Pour lire la premier item on utilise la valeur 0, le deuxième on utilise la valeur 1, etc.

Comment Afficher l'index d'un item?

Pour trouver l'indice d'un élément donné d'une liste avec python il existe la méthode index.

Exemple d'utilisation:

```
>>> l = ['Homer','Bart','Marge']
>>> l.index('Marge')
2
>>> l.index('Bart')
1
```

Comment Supprimer un item(element) d'une liste?

Il est parfois nécessaire de supprimer une entrée de la liste. Pour cela vous pouvez utiliser la fonction del.

```
>>> liste = ["a", "b", "c"]
>>> del liste[1]
>>> liste
```

Il est possible de supprimer une entrée d'une liste avec sa valeur avec la méthode remove.

```
>>> liste = ["a", "b", "c"]
>>> liste.remove("a")
>>> liste
```

Fonction len()

L'instruction len() vous permet de connaître la longueur d'une liste, c'est-à-dire le nombre d'éléments que contient la liste.

un exemple d'utilisation :

```
>>> test= [1,2,3,5,10]
>>> len(test)
```

Questions?

Merci !