

TP1 : prise en main de Python

Objectif : découvrir l'environnement de travail Python.

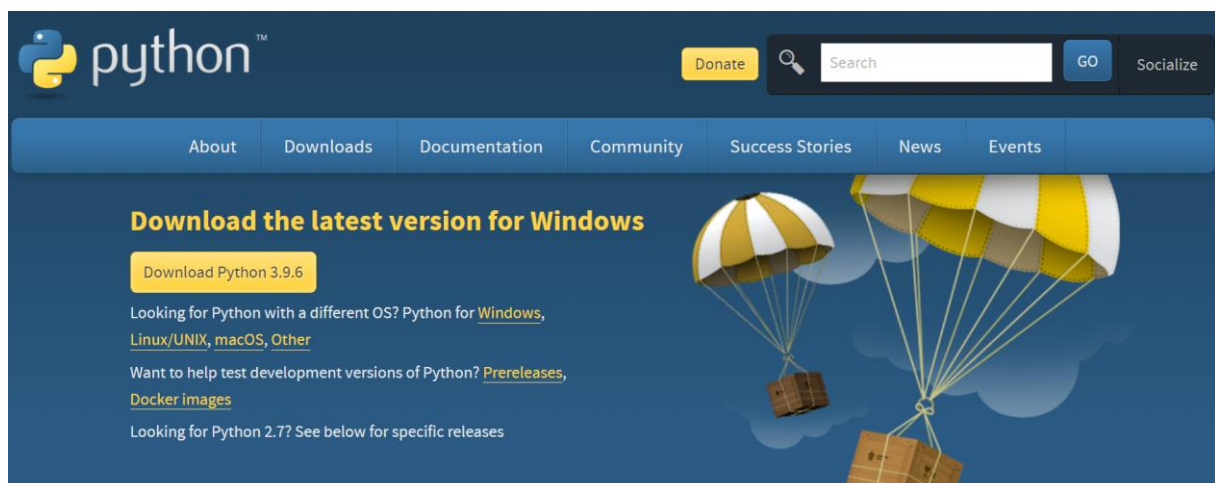
I. Environnement Python

Dans ce premier TP de la ressource R107, vous allez utiliser le langage Python en exploitant ses différents environnements d'exécution. Le premier d'entre eux consiste à charger la console python et lui faire exécuter les lignes de commande les unes après les autres.

✚ **Remarque :** *Python 3.9.7 et Python 2.7.18 sont les dernières versions stables au moment de l'édition de ce document.*

Système d'exploitation Windows

Téléchargez l'installateur python à partir du site <https://www.python.org/downloads/>



Vous pouvez choisir de télécharger les dernières versions stables de « **Python 3** » ou « **Python 2** ». Pour les TP choisir la **version 3** en cliquant sur le bouton « **Download Python 3.9.7** » puis exécutez l'installateur.

✚ L'installation est à faire uniquement si Python n'est pas présent sur la machine de la salle TP.

Une fois installé, vous disposez de trois méthodes pour charger l'environnement :

- ✓ Ouvrir une console « **cmd** » puis taper la commande « **python** »

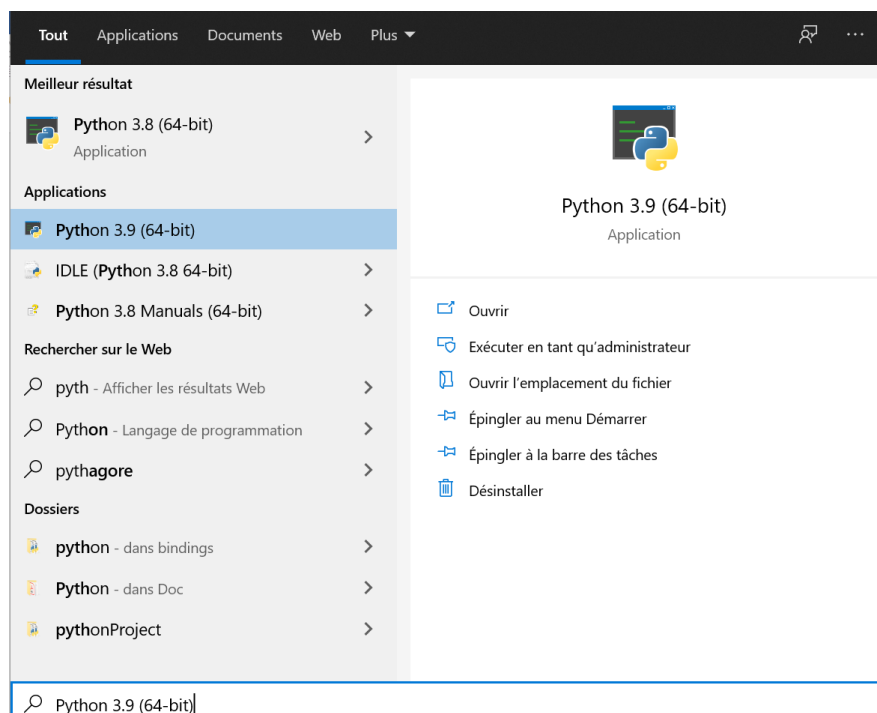
```
C:\> Invite de commandes - python

Microsoft Windows [version 10.0.19041.1165]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\IB>python -V
Python 3.9.7

C:\Users\IB>python
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

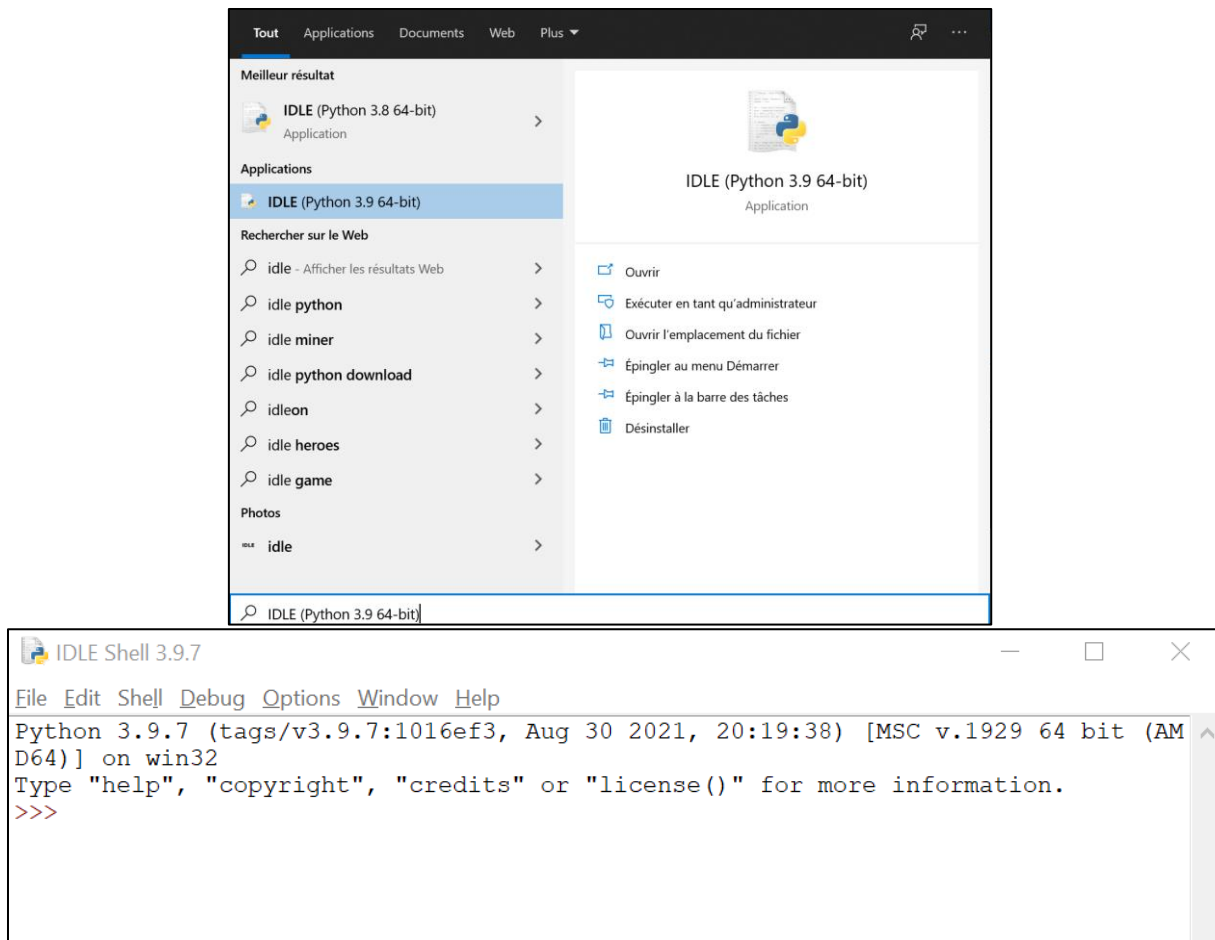
- ✓ Lancer directement un terminal python avec l'application « **Python 3.9 (64-bit)** »



```
Python 3.9 (64-bit)

Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- ✓ Lancer l'interface de développement IDLE avec l'application « **IDLE (Python 3.9 64-bit)** »



✚ **N.B.** Le déroulement des TP de la ressource R107 sera uniquement sous Windows. A noter que l'exécution de Python dans un environnement Linux est strictement identique à celle d'un environnement Windows.

II. Première Instruction en Python

1) Typage dynamique

Nous avons vu en cours comment initialiser une variable en python. Il suffit de déclarer un nom de variable (une étiquette en quelque sorte) en lui affectant une valeur. Python détermine alors automatiquement son type en prenant en compte la nature de la valeur affectée. On parle de **typage dynamique**.

La fonction « **type()** » prenant en argument le nom de la variable entre parenthèses permet d'afficher le type de donnée ainsi alloué.

```
>>> a=254
>>> type(a)
<class 'int'>
```

2) Affichage et formatage

La fonction « **print()** » permet d'afficher dans la console une information transmise en argument entre parenthèses. Il est possible d'afficher plusieurs informations en les séparant par des virgules. Les données à afficher peuvent être contenues dans des variables appelées comme arguments de la fonction

```
>>> data1=10
>>> data2=11
>>> print(data1, data2)
10 11
```

On peut également formater l'information à afficher en ajoutant du texte :

```
>>> age = 32
>>> name = 'Jean'
>>> print(name , 'a' , age , 'ans')
Jean a 32 ans
```

Il existe aussi une autre façon de faire pour formater le message à afficher. Cela consiste à utiliser des accolades vides {} précisent l'endroit où le contenu de la variable doit être inséré. Et d'utiliser l'instruction .format() qui indique la liste des variables à insérer.

```
>>> print("{} a {} ans".format(name, age)) # méthode format() permet
une meilleure organisation de l'affichage des variable
Jean a 32 ans
```

Toujours avec cette deuxième façon de faire, on peut définir la précision à utiliser pour l'affichage des valeurs de type float

```
>>> var = (4500 + 2575) / 14800
>>> print("Le résultat du calcul est", var)
Le résultat du calcul est 0.4780405405405405
>>> print("Le résultat du calcul est {:.2f}".format(var))
Le résultat du calcul est 0.48
```

En effet, Les deux points : indiquent que l'on veut préciser le format. Le formatage avec .xf (x étant un entier positif) renvoie un résultat **arrondi**, avec f pour float

Une autre façon de faire qui est possible à partir de la version 3.6 de Python c'est d'utiliser la syntaxe suivante avec le f-Strings : **f"phrase à afficher avec {variable}"**

```
>>> name = "Eric"
>>> age = 20
>>> f"Hello, {name}. vous avez {age} ans."
'Hello, Eric. Vous avez 20 ans.'
```

En python, il est possible d'obtenir de l'aide sur un élément en utilisant la fonction « **help()** » prenant en argument l'élément considéré entre parenthèse

Vous pouvez également obtenir de l'aide directement à partir du site officiel <https://docs.python.org/3/> En suivant le lien « **Library Reference** ». Vous obtenez toute l'aide nécessaire sur les éléments du langage.

Exercice 1 :

Dans une console IDLE, initialisez les variables « **nom** », « **prenom** », « **math** », « **anglais** », « **info** », « **promotion** » contenant respectivement une chaîne de caractères représentant le nom, une autre chaîne de caractères pour le prénom, un réel pour la note en mathématique, un réel pour la note d'anglais, un réel pour la note d'info, un entier pour l'année de première inscription à l'IUT. Initialisez-les à votre convenance en respectant les consignes précédentes. Vous rajouter aussi la variable « **moy** » qui contiendra la moyenne des trois notes que vous devez calculer automatiquement à partir des notes saisies.


- 1) Afficher le type de chaque variable
- 2) Afficher la moyenne en formatant le résultat comme suit :
« L'étudiant **toto titi** de la promotion **2021** a une moyenne de **12,5** »

Exercice 2 :

Déclarer les trois variables « **jour** », « **heure** », « **minute** » contenant le jour du mois, l'heure entre 0 et 23 et les minutes entre 0 et 59. Puis, calculer le nombre de minutes écoulées depuis le début du mois puis affichez le résultat à l'écran.

3) Interaction avec l'utilisateur – Script

Python offre la possibilité d'interagir avec l'utilisateur pour lui permettre de saisir des valeurs en cours d'exécution. Il suffit d'utiliser la fonction « **input()** » pour initialiser la variable avec la valeur qui sera saisie à la suite par l'utilisateur.

 **Attention** : Avec Python 2 (ou 2.7), le type de la variable initialisée par python est défini dynamiquement en fonction de la valeur saisie par l'utilisateur. Avec Python 3, la fonction retourne **systématiquement une chaîne de caractères**, même lorsque l'on saisit une valeur numérique.

```
>>> print("Saisir une annee : ")
>>> annee=input()
2021
```

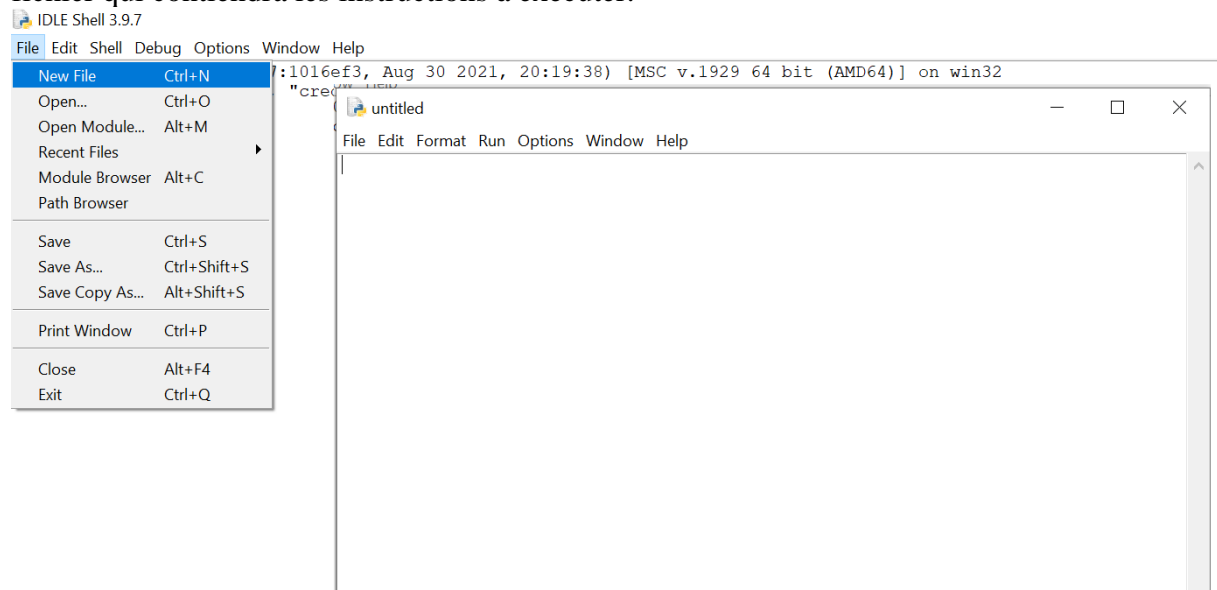
```
>>> print(annee)
2021
>>> print(annee+1)
Traceback (most recent call last):
File "<pyshell#32>", line 1, in <module>
print(annee+1)
TypeError: must be str, not int2021
```

Python met à disposition des fonctions intégrées permettant de manipuler les chaînes de caractères. Vous devez convertir la chaîne saisie dans le format numérique souhaité en utilisant la fonction adaptée :

- ✓ **len(ch)** : renvoie la longueur de la chaîne **ch**.
- ✓ **float(ch)** : convertit la chaîne **ch** en un nombre réel (**float**) (**ch** représente un nombre, réel ou entier)
- ✓ **int(ch)** : convertit la chaîne **ch** en un nombre entier (mêmes restrictions)
- ✓ **str(obj)** : convertit (ou représente) l'objet **obj** en une chaîne de caractères

```
>>> annee=int(annee)
>>> print(annee+1)
2022
```

Il peut être utile d'enregistrer une séquence d'instructions dans un fichier pour la réutiliser. On parle de **script** python. Il se caractérise par un nom de fichier portant nécessairement l'extension « **py** ». Sous IDLE, il est possible de créer des scripts en créant un nouveau fichier qui contiendra les instructions à exécuter.



Pour lancer le script, il faut le sauvegarder tout d'abord puis de l'exécuter en appuyant sur F5.

Vous pouvez également écrire votre script avec un Notepad et l'enregistrer avec l'extension .py puis le lancer depuis l'invite de commandes Windows (cmd) avec la commande : **python nom_du_fichier.py**

N.B Sous Linux, afin de pouvoir lancer directement dans un terminal « **bash** » le script crée, il faut insérer le « **shebang** » en tête de fichier définissant le chemin vers l'interpréteur python (« **# !usr/bin/python3.6**») et il faut aussi rendre le fichier exécutable.

Exercice 3 :

Reprenez l'exercice 2 pour le transformer en script que vous nommerez « **tp1exo3.py** ». Prévoir une saisie interactive en affichant un message d'invite personnalisé pour chaque variable « **jour** », « **heure** » et « **minute** ». Le script doit calculer le nombre de minutes écoulées depuis le début du mois, puis il affiche le résultat à l'écran.

Exercice 4 :

Ecrire un deuxième script nommé « **tp1exo4.py** » qui permet à l'utilisateur de saisir un nombre représentant les minutes écoulées depuis le début du mois (le mois en cours par exemple) et affiche la date associée (le jour du mois).

III. Import des modules

Le langage python intègre peu de fonctions – voir la liste des fonctions disponibles ici : <https://docs.python.org/3/library/functions.html>. Il est parfois nécessaire dans un programme de faire appel à des fonctions existantes déployées dans d'autres fichiers. On parle de module. Un module est un fichier regroupant des fonctions. Pour importer un module il suffit d'utiliser l'instruction « **import** »

```
>>> import nom_du_module
```

Il est possible de n'importer que certaine fonction d'un module en utilisant la clause « **from** » dans l'instruction « **import** »

```
>>> from nom_du_module import nom_fonction1, nom_fonction2
```

Si vous souhaitez importer toutes les fonctions sans le module lui-même utiliser le caractère de remplacement « ***** »

```
>>> from nom_du_module import *
```

Pour les exercices suivants, nous allons générer des valeurs aléatoirement. Pour cela, nous allons importer le module « **random** » qui est un « **module standard** » livré avec le langage. Vous pouvez consulter l'aide relative au module **random** et repérer les différents types de données qui peuvent être générés avec les fonctions du module.

Exercice 5 :

Ecrire un script nommé « **tp1exo5.py** » qui permet de générer un nombre aléatoire entre 0 et 100 et de l'afficher.

Exercice 6 :

Ecrire un script nommé « **tp1exo6.py** » qui simule le lancer d'une pièce avec un nombre aléatoire entre 0 et 100, si le nombre est plus petit que 50, le résultat à afficher est « Pile ! » sinon c'est « Face ! ».

Exercice 7 :

Ecrire un script nommé « **tp1exo7.py** » qui simule une pièce truquée qui donnera « Pile » en moyenne deux fois sur trois.

🎉 Si vous avez tout fini avant l'heure bravo ! vous pouvez maintenant lire le guide PyCharm présent sur moodle