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 3.14.0*** *est la dernière version stable au moment de l'édition de ce document.*

Système d'exploitation Windows

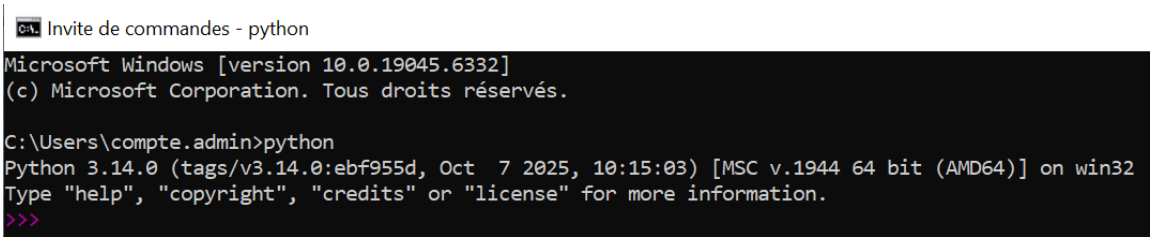
Téléchargez l'installateur python à partir du site https://www.python.org/downloads/

Vous pouvez télécharger les dernières versions stables de « **Python 3** ». Pour les TP choisir la **version 3.14** en cliquant sur le bouton « **Download Python 3.13.0** » puis exécutez l'installateur.

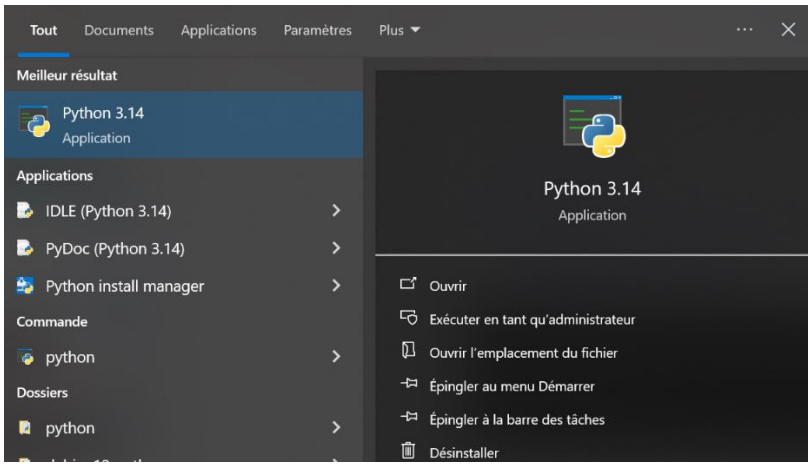
- L'installation est à faire uniquement si Python n'est pas présent sur la machine de la salle TP.

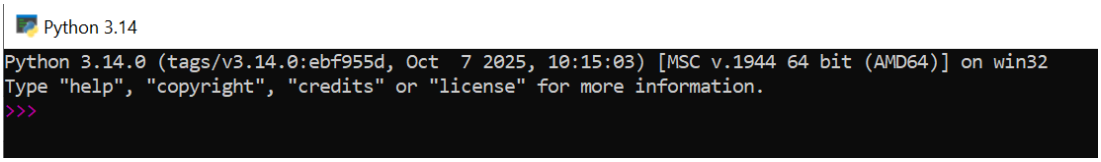
Une fois installé, vous disposez de trois méthodes pour lancer l'environnement Python :

1) Ouvrir une console « **cmd** » puis taper la commande « **python** »

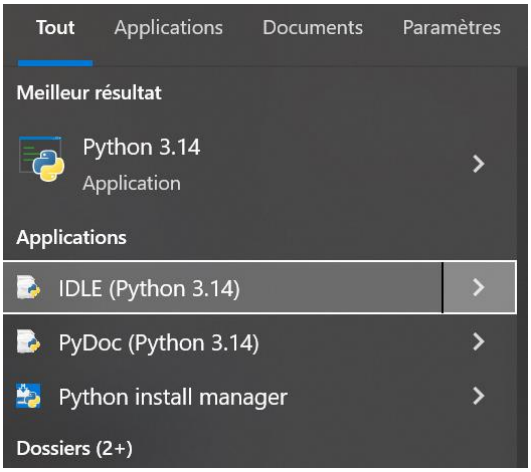


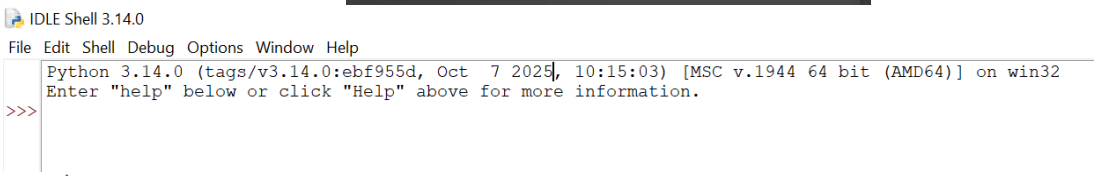
2) Lancer directement un terminal python avec l'application « **Python 3.14 (64-bit)** »





3) Lancer l'interface de développement IDLE avec l'application « **IDLE (Python 3.12 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 parti 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 et 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 « **m** » 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 **2025** a une moyenne de **12,5 »**

nom, prenom : str
math_note, anglais, info : float
promotion : int
Moyenne calculée : 13,2

Exercice 2 : Operations sur les entiers

On considère la séquence d'instructions suivante (exécution pas à pas). Sans utiliser l'invite de commande Python, essayer de compléter le tableau ci-après avec les valeurs de x et y après chaque instruction. Une fois c'est fait, vous pouvez vérifier les valeurs en exécutant les instructions.

	Instructions	x	y
1)	**x = 3**	3	0
2)	**y = 0**	3	0
3)	**y = 2*x**	3	6
4)	**y = 5*y + x**	3	33
5)	**y = y%7**	3	5
6)	**y= 4 + x**	3	7

7)	**x += 4**	7	7
8)	**y += x**	7	14
9)	**y %= x + 1**	7	6
10)	**y = 5 * (x + y)**	7	65

Exercice 3

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. Valeurs finales : x = 7, y = 65

3) Interaction avec l'utilisateur

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()
2025
>>> print(annee)
2025
>>> print(annee+1)
Traceback (most recent call last):
File "<pyshell#32>", line 1, in <module>
print(annee+1)
TypeError: must be str, not int2025
```

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

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

```
>>> annee=int(annee)
```

```
>>> print(annee+1)
2026
```

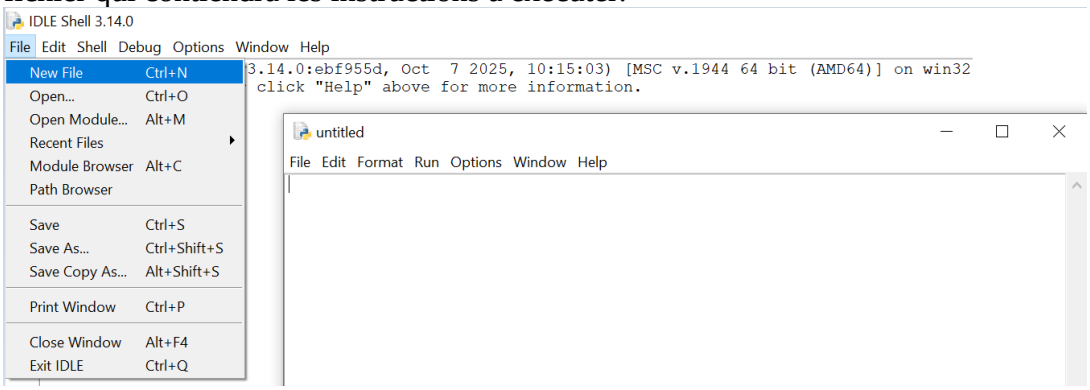
Exercice 4 : Operations sur les chaines de caractères

On considère la séquence d'instructions suivante (exécution pas à pas). En utilisant l'invite de commande Python, compléter le tableau ci-dessous avec les valeurs de x et y après chaque instruction, puis de l'affichage obtenu de la variable y.

	Instructions	x	y	Print(y)
1)	x = "Sa"	Sa		
2)	y = x+"lut"	Sa	Salut	Salut
3)	y = y+"\n\tCesar"	Sa	Salut\n\tCesar	Salut Cesar
4)	x+= "lut "+"Jules"	Salut Jules	Salut\n\tCesar	Salut Cesar
5)	y = len(x)	Salut Jules	11	11
6)	y = len("BonjourJuju")	Salut Jules	11	11
7)	y = "Salut! "*2	Salut Jules	Salut! Salut!	Salut! Salut
8)	x = len(y)	14	Salut! Salut!	Salut! Salut
9)	y.strip()	14	Salut! Salut!	Salut! Salut
10)	y= y.strip()	14	Salut! Salut!	Salut! Salut

4) Les scripts

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 scriptes en créant un nouveau fichier qui contiendra les instructions à exécuter.



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

Vous pouvez également écrire votre scripte avec un n'importe quel éditeur de texte (ex. 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/python3.14**») et il faut aussi rendre le fichier exécutable.

Exercice 5 :

Reprenez l'exercice 2 pour le transformer en script que vous nommerez «**tp1exo5.py** ». Adapter votre script afin de rajouter 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 6 :

Ecrire un deuxième script nommé « **tp1exo6.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).

Exercice 7 : évaluation d'expression booléennes

On considère le script suivant :

```
print("Donner les trois nombres entiers x1, x2 et x3 : ")
x1 = int(input("valeur de x1"))
x2 = int(input("valeur de x2"))
x3 = int(input("valeur de x3"))
condition = expression_booleenne
if(condition) :
      print(str(condition)+" = VRAI")
else :
      print(str(condition)+" = FAUX")
```

Dans le tableau ci-après, on vous donne la formulation de l' « expression_booleenne » et les valeurs utilisées pour x1, x2 et x3. Sans tester le script Python, essayer de compléter le tableau ci-après par le résultat qui sera affiché par le programme pour chaque expression. Une fois c'est fait, vous pouvez vérifier vos réponses en exécutant le script avec l'expression booléenne concernée.

	Expression booléenne	x1 x2 x3 5 8 2	x1 x2 x3 3 3 9	x1 x2 x3 5 3 7
1)	**not(x1 == x2) and not(x1 == x2)**	Vrai	Faux	Vrai
2)	**(x1 > x2) or (x3 < x1)**	Vrai	Faux	Vrai
3)	**(x1 < x2) and (x2 < x3)**	Faux	Faux	Faux
4)	**not((x1 > x2) and (x1 < x3))**	Vrai	Vrai	Faux
5)	**(x1 == x2) or (x3 == x1) or (x2 == x3)**	Faux	Vrai	Faux

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 l'exercice suivant, 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 générer avec les fonctions du module.

Exercice 8 :

Ecrire un script nommé « **tp1exo8.py** » qui permet de générer un nombre aléatoire entre 0 et 100 et de l'afficher.

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