

PARTIE 2 : ALGORITHMIQUE & PROGRAMMATION

TP 1 – DÉCOUVERTE DE PYTHON

Objectifs

- Découvrir l'environnement Python.
- Découvrir les limites de Python.
- Manipuler différents types.

Remarque

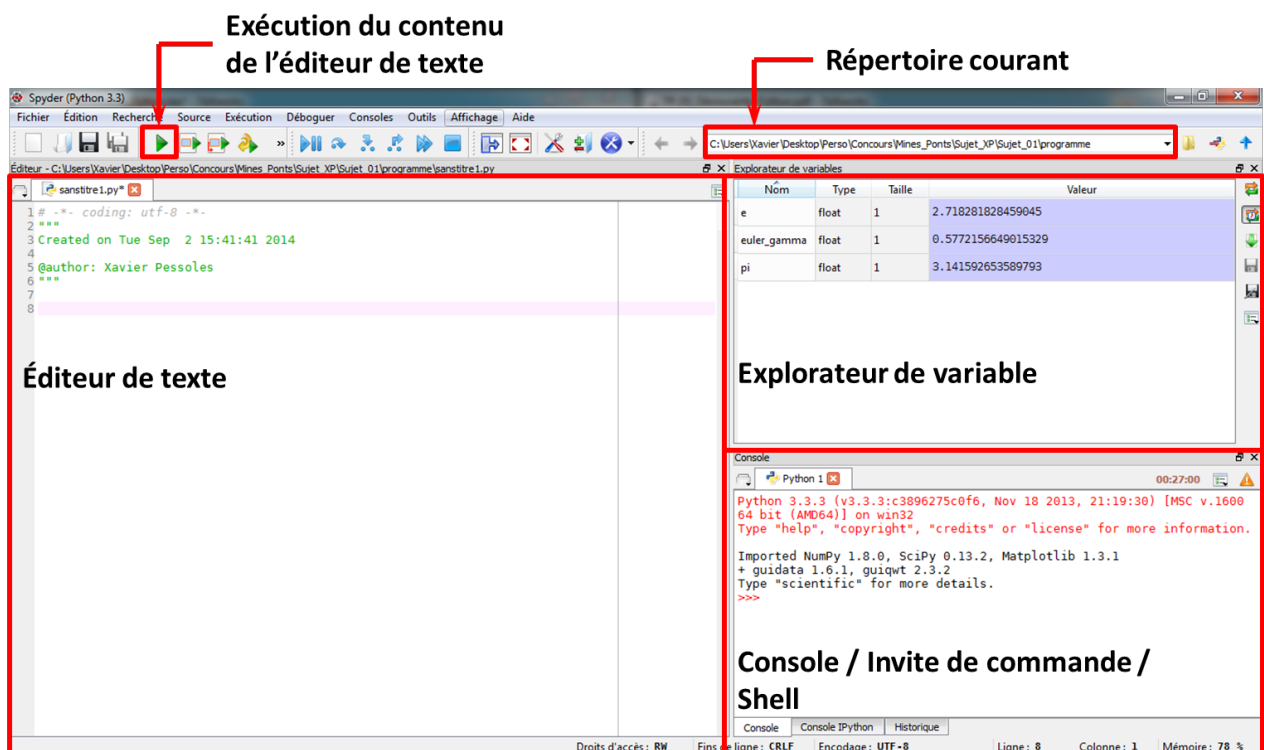
Python est un langage de programmation dont nous découvrirons les possibilités en cours. Pour exécuter un programme écrit en Python, il existe plusieurs « interpréteurs ».

Dans le cadre de ce TP, nous proposons d'utiliser Spyder qui intègre des bibliothèques scientifiques.

Commencer par copier-coller le Sujet de TP dans votre répertoire personnel.

Pour ouvrir Spyder :

1. sur le bureau, aller dans le dossier Pédagogie ;
2. lancer Spyder.



Fenêtre Spyder

Ce document évolue. Merci de signaler toutes les erreurs ou coquilles constatées.

La console (appelée aussi invite de commande, shell, interpréteur de commande ...) permet de tester des lignes de codes. Le résultat d'une commande s'affiche directement dans la console.

Exemple

Saisir la ligne suivante dans la console :



```
>>> print('When Alexander Bell invented telephone he had 3 missed calls from Chuck Norris.')
```

Quel est le résultat ?

Exemple

Saisir la ligne suivante dans l'éditeur de texte. :



```
print('When Alexander Bell invented telephone he had 3 missed calls from Chuck Norris.')
```

Quel est le résultat ?

Suivant les éditeurs de texte, de nombreuses fonctionnalités peuvent être intégrées (coloration syntaxique, proposition de méthodes, complétion ...).

La programmation permet de stocker des variables. Ainsi, l'explorateur de variable peut être nécessaire pour visualiser leur contenu.

Exercice 1 – Découverte de l'interpréteur

Objectifs

- Découvrir l'interpréteur (shell).
- Découvrir les instructions élémentaires.
- Découvrir la notion de type.
- Découvrir la notion d'affectation.

Dans l'interpréteur python, saisir les instructions suivantes :



```
>>> 4+3
>>> 4*3
>>> 7/2.
>>> 7//2
>>> 7**2
>>> 7%2
```

Question 1 Quel est le but de chacune de ces instructions ?

Dans l'interpréteur python, saisir les instructions suivantes :



```
>>> "Abracadabra"
>>> print('abracadabra')
>>> #abracadabra
>>> abracadabra
```

Question 2 Expliquer les différences entre les instructions précédentes.

Remarque

Que se passe-t-il en saisissant l'instruction suivante ?



```
>>> "Abrac"+"adabra"
```

Remarque

De manière générale pour afficher du texte, on utilise l'instruction *print*.

On va maintenant *affecter des variables*, c'est à dire qu'on va chercher à stocker des valeurs. Dans l'interpréteur python, saisir les instructions suivantes :



```
>>> a=1
>>> b=2
>>> a=b
>>> b=a
```

Question 3 En utilisant un tableau, préciser les valeurs stockées dans *a* et dans *b* après l'exécution de chacune des commandes.

Dans l'interpréteur python, saisir les instructions suivantes :



```
>>>a=1
>>>b='1'
>>>c=1.
```

Question 4 Quelle est d'après la différence entre ces affectations ?

La fonction *type(variable)* permet de connaître... le type d'une variable.

Question 5 Quel est le type de chacune des variables *a*, *b* et *c* ? Comment peut-on traduire ces types en français ?

Dans l'interpréteur python, saisir les instructions suivantes :

python

```
>>>a=1
>>>b=2
>>>a<b
>>>a>b
>>>a==b
>>>a!=b
```

Question 6 Expliquer le but de chacune de ces opérations.

La commande `input()` permet au shell de demander à l'utilisateur de saisir une variable. En lançant cette commande le shell attend donc une saisie au clavier terminée par la touche entrée.

Saisir les instructions suivantes :

python

```
>>>input("Coucou :")
>>>input()
```

Question 7 Expliquer ce qu'il se passe.

Il est alors possible de stocker une information donnée par l'utilisateur. Saisir les instructions suivantes :

python

```
>>>a = input("Quel âge as-tu ?")
```

Question 8 Quelle valeur contient *a* ? Quel est le type de *a* ? Quelle différence peut-il y avoir entre saisir son âge en chiffres ou en lettres ? Quel problème cela peut-il poser ?

Remarque

Vous avez dû constater qu'il existe les types `int`, `str`, `float`. Lorsqu'on manipule des nombres (entiers, réels ou chaîne de caractère, il est possible de les convertir) :

- Conversion d'un nombre réel en chaîne de caractère :
– `str(2.)`
- Conversion d'une chaîne de caractère en nombre réel :
– `float("2.")`

Attention à bien manipuler des nombres.

Exercice 2 : Découverte de l'éditeur de texte

Un des avantages d'utiliser un éditeur de texte est qu'il permet de corriger les erreurs de frappe ou de syntaxe. Il permet aussi de sauvegarder le code qui a été saisi.



Saisir l'instruction suivante dans le fichier.

```
print('Chuck Norris counted to infinity - twice')
```

Question 1 Quel est le résultat lorsqu'on exécute le code ?

Question 2 En vous aidant de l'exercice précédent, on demande d'écrire un programme qui aura pour but de demander un nombre à l'utilisateur. Le programme devra alors élever ce nombre au carré et renvoyer le résultat dans une phrase (Par exemple : si l'utilisateur saisi le nombre 8, le programme devra permettre d'écrire la phrase suivante dans l'interpréteur de commande : « Le carré de 8 vaut 64 ».).

Exercice 3 : Découverte de la boucle for

Définition

Le mot algorithme vient du nom latinisé du mathématicien perse Al-Khawarizmi, surnommé « le père de l'algèbre ». Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème.

Objectifs

- Découvrir la boucle for.

On donne la syntaxe d'une boucle for :



```
for i in range(0,10):
    print(i)
```

Remarque

- La première ligne se termine par deux points.
- La seconde ligne est précédée d'une indentation. Une indentation est composée de 4 espaces.
- Pour terminer une boucle for, il faut supprimer l'indentation (c'est-à-dire revenir un début de ligne).

Question 1 Saisir le code ci-dessus. Quel est son but ?

Question 2 En substituant les lignes suivantes dans le code saisi précédemment, expliquer le rôle de la fonction range.



```
>>>range(5)
>>>range(0,6)
>>>range(0,10,2)
>>>range(10,2,-2)
```

Question 3 En utilisant une boucle *for*, réaliser un algorithme permettant de calculer la somme s des n premiers entiers, n étant laissé au choix de l'utilisateur :

$$s = \sum_{i=0}^n i$$

Question 4 Le calcul de cette boucle a-t-elle un intérêt ? Si non, pourquoi ?

Exercice 4 : Boucle while

Objectifs

Découvrir la boucle *while*.

python

La boucle *while* permet de réaliser une instruction tant qu'une condition reste vraie :

```
i=0
while i<10:
    print(i)
    i=i+1
```

Attention

Il faut prêter attention à ce que la condition évolue à chaque itération pour éviter le risque de ne jamais sortir de la boucle.

Question Traiter l'exercice précédent en utilisant une boucle *while*.

Exercice 5 : Heures, Minutes, Secondes – Structure If

python

La boucle *if* permet de réaliser une instruction en fonction de la validité d'une condition :

```
i=2
if i<10:
    print(i)
else :
    print('i>10')
```

Question Écrire un programme Python permettant de calculer le temps écoulé entre deux horaires saisis au clavier par l'utilisateur ; les horaires seront saisis au format (heures, minutes, secondes) et le résultat sera affiché en secondes et au format (heures, minutes, secondes).

Exercice 6 : Calcul de factorielle

$n!$ (factorielle n) est définie par :

$$\forall n \in \mathbb{N} \begin{cases} \text{si } n = 0, n! = 1 \\ \text{sinon } n! = \prod_{i=1}^n i \end{cases}$$

Question Écrire un programme Python permettant de calculer $n!$.

Exercice 6 : Jeu du plus ou moins

Question Écrire un programme Python permettant à l'utilisateur de trouver un nombre choisi au hasard par le programme en ayant comme unique indication « le nombre cherché est plus grand » ou « le nombre cherché est plus petit ». L'intervalle dans lequel se trouve le nombre cherché est précisé à l'utilisateur (entre 1 et 100 par exemple).

Au cours de ce programme, l'ordinateur doit choisir un nombre au hasard. On va pour cela utiliser une fonction qui n'est pas présente par défaut dans Python, la fonction `randrange` appartenant à la bibliothèque `random`.

Import de la fonction :

```
from random import randrange
```

Utilisation de la fonction :

```
randrange(n) # renvoie un nombre entier aléatoire compris entre 0 et n-1
randrange(m,n) # renvoie un nombre entier aléatoire compris entre m et n-1
```

Exercice 7 : Année bissextile

Le système de date utilisé par Excel est basé sur le calendrier grégorien, établi en 1582 par le pape Grégoire XIII. Ce calendrier a été conçu pour corriger les erreurs introduites par le calendrier julien moins précis.

Dans le calendrier grégorien, une année normale comprend 365 jours. Étant donné que la durée réelle d'une année est en fait de 365.242198 jours, une « année bissextile » de 366 jours est utilisée tous les quatre ans pour éliminer l'erreur provoquée par trois années consécutives de 365 jours. N'importe quelle année divisible par 4 est une année bissextile : par exemple, 1988, 1992 et 1996 sont des années bissextiles.

Toutefois, il reste encore une petite erreur qui doit être prise en compte. Pour éliminer cette erreur, le calendrier grégorien prévoit qu'une année qui est divisible par 100 (par exemple, 1900) est une année bissextile uniquement si elle est également divisible par 400.

Pour cette raison, les années suivantes ne sont pas des années bissextiles : 1700, 1800, 1900, 2100, 2200, 2300, 2500, 2600. C'est parce qu'elles sont également divisibles par 100, mais pas par 400.

Les années suivantes sont des années bissextiles : 1600, 2000, 2400. C'est parce qu'elles sont divisibles par 100 et 400.

Remarque

En Python, `5%4` affiche le reste de la division euclidienne de 5 par 4.

Question Écrire un programme capable de déterminer si l'année entrée par l'utilisateur est une année bissextile ou pas.

Exercice 8 : Algorithmes de recherche élémentaires

Question Écrire un programme Python permettant à l'utilisateur de saisir deux entiers n et m tels que $n < m$ et de rechercher parmi tous les entiers contenus dans l'intervalle $[n, m]$ ceux qui :

- sont impairs ;
- sont multiples de 5 ;
- contiennent le chiffre 2.

Exercice 9 : La suite de Syracuse ou conjecture de Collatz

On appelle fonction de Collatz la fonction $f : \mathbb{N}^* \rightarrow \mathbb{N}^*$ qui à n associe $n/2$ si n est pair et $3n + 1$ sinon. On appelle la suite de Collatz une suite récurrente telle que $u_0 \in \mathbb{N}^*$ et pour tout k , $u_{k+1} = f(u_k)$.

Question 1 Pour $u_0 = 1$, $u_0 = 2$, $u_0 = 3$, calculer les premiers éléments de la suite. Que remarquez-vous ?

Remarque

Vous devez constater que lorsqu'un terme de la suite vaut 4, la suite devient périodique : les termes de la suite valant successivement 4, 2, 1, 4, 2, 1, etc.

La conjecture de Collatz est une hypothèse selon laquelle, toute suite de Collatz a un terme valant 4.

Cette conjecture n'a jamais été démontrée ou infirmée. Elle est mise à prix à 1 millions de dollars.

Question 2 Écrire un programme permettant de calculer et d'afficher les termes de la suite. Les constats effectués précédemment sont-ils vérifiés ?

On appelle vol de l'entier n la liste des termes de la suite de Collatz initialisée avec $u_0 = n$ jusqu'à l'apparition du premier 4.

Question 3 Écrire un programme permettant de calculer le vol d'un entier.

Exercice 10 : Triangle de Pascal

Le triangle de Pascal peut prendre la forme suivante :

		Colonne p							
Ligne n	1								
	1	1							
	1	2	1						
	1	3	3	1					
	1	4	6	4	1				
	1	5	10	10	5	1			
	1	6	15	20	15	6	1		
	1	7	21	35	35	21	7	1	
	1	8	28	56	70	56	28	8	1

On rappelle que le coefficient $\binom{n}{p}$ de la ligne n et de la colonne p s'obtient en ajoutant le coefficient de la ligne $n - 1$ et de la colonne p au coefficient de la ligne $n - 1$ et de la colonne $p - 1$:

$$\binom{n}{p} = \binom{n-1}{p} + \binom{n-1}{p-1}$$

Par exemple, le coefficient 10 (sixième ligne, troisième colonne) est la somme de 4 (cinquième ligne, deuxième colonne) et de 6 (cinquième ligne, troisième colonne).

Le coefficient $\binom{n}{p}$ peut aussi se calculer à partir de la formule du binôme de Newton : $\binom{n}{p} = n! / (p!(n-p)!)$

Question Réaliser un programme permettant d'afficher les différents coefficients du triangle de Pascal.