LYCÉE Jules Haag CPGE PT / PTSI Votre avenir avec un temps d'avance	**TP d'Informatique** Rappels des bases de la programmation en langage python	**TP1**

Partie du programme :

1. Programme du premier semestre

1.1 Rappels des notions de programmation en langage python vues au lycée

L'objectif de ce TP est de rappeler les bases indispensables dans la programmation en langage python.

Ressources **:**

- **Cours 1 : Introduction au langage de programmation Python**
- **Cours 2 : Listes et Tuples en langage Python**

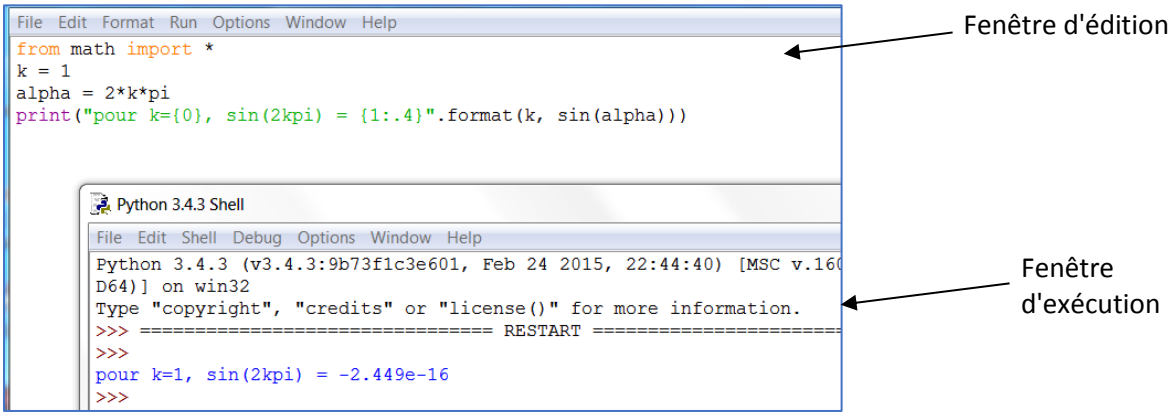
1. Rappels :

1.1. Qu'est-ce que python

- **Python est un langage de programmation interprété**, c'est-à-dire que la traduction se fait en temps réel, lors de l'exécution. Cette dernière nécessite la présence d'un interpréteur (IDLE pour nous). L'un des avantages est qu'un même script peut être exécuté sur plusieurs plateformes différentes, en revanche la traduction (interprétation) du code à chaque exécution a un impact sur les performances.
- Python favorise la programmation impérative structurée, fonctionnelle et orientée objet (définition et interaction de briques logicielles appelées objets).

1.2. Interpréteur utilisé en CPGE

IDLE :



Les principales fonctionnalités de IDLE sont :

- L'éditeur de texte avec coloration syntaxique, l'autocomplétion (l'utilisateur se voit proposer un complément qui pourrait convenir à la chaîne de caractères qu'il a commencé à taper), l'indentation (ajout de tabulations ou d'espaces dans un fichier texte) ;
- Le débogueur intégré avec avancement par étape

En pratique, dans IDLE, vous créez un nouveau script (*Menu File, puis New File*) que vous enregistrez sous un nom « nom de votre script ».py sur votre clef usb. Une fois rédigé, le programme sera exécuté en cliquant sur *Run module* dans le menu *Run* ou en appuyant sur F5. Le résultat sera automatiquement affiché dans la fenêtre de l'interpréteur.

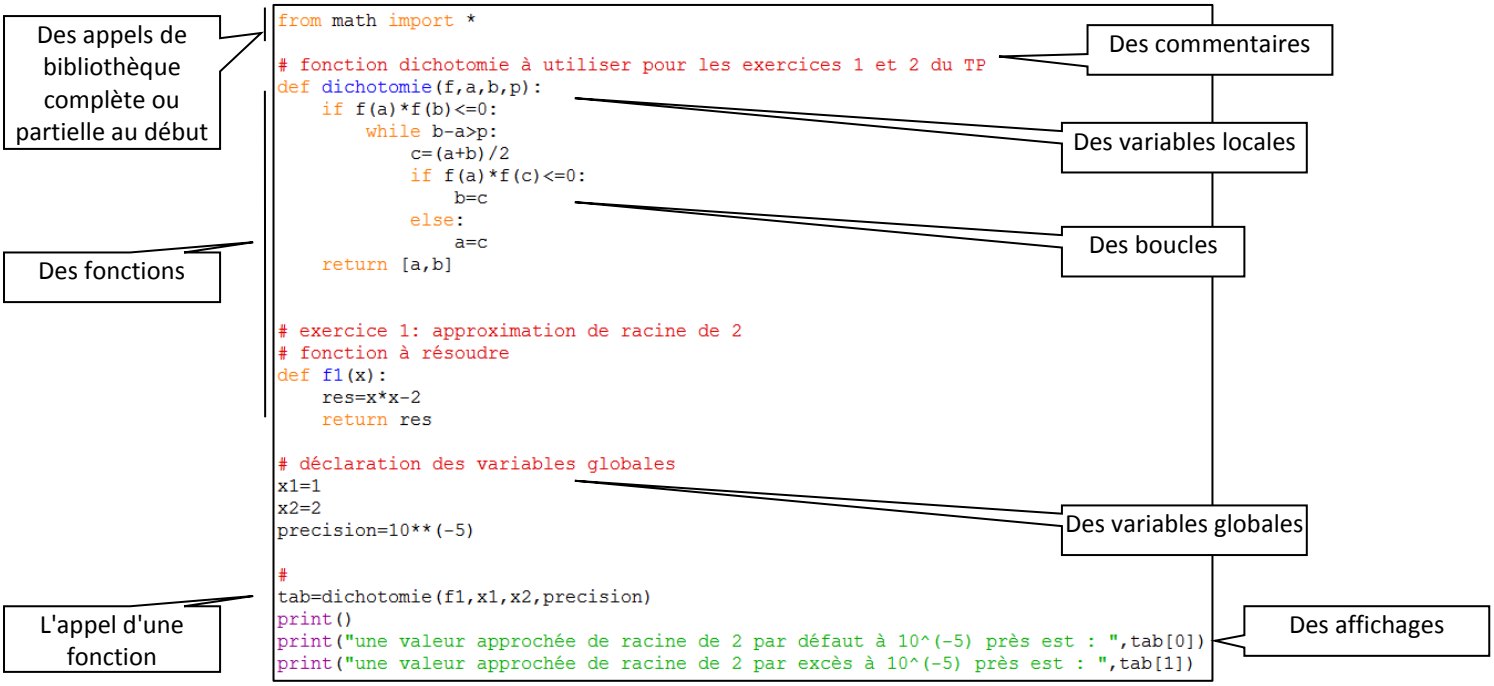
Avantages :

- Simple
- Rapide
- Coloration syntaxique

Défauts :

- Bon nombre de bibliothèques ne sont pas installées initialement.
- Interface minimaliste
- Fenêtres d'édition et d'exécution séparées

1.3. Contenu d'un programme



En pratique

Au brouillon :

- **Avant de programmer sur le logiciel, travailler au brouillon** votre problématique.
- S'il y a des itérations, écrire au brouillon les premières boucles.
- Faites des schémas de résolution au brouillon.

Sur logiciel :

- Faire des tests régulièrement : des print(….) afin d'avoir les résultats intermédiaires
- **Ne pas attendre la fin du programme pour retrouver une erreur**
- Mettre des commentaires :
  - Pour vous y retrouver
  - Pour une personne extérieure qui viendrait lire votre programme
  - Pour le concours !

2. Utilisation de Python en mode interprété

Vous trouverez un **raccourci** pour le **lancement d'IDLE** dans le **dossier commun-pedagogie** (T : pour les élèves)
T :\CPGE\Informatique\PTSI

IDLE vous propose deux fenêtres d'utilisation :

- La fenêtre édition pour taper un programme qui sera sauvegardé
- La fenêtre exécution pour exécuter un programme écrit dans le mode édition ou pour écrire des instructions exécutées en direct à chaque validation

Si vous êtes en mode édition, passez en **mode exécution** par la fonction Run (raccourci touche F5).
Le *prompt* >>> indique que l'interpréteur est en attente d'une commande python.

2.1. Prise en main en mode interprété

1. Entrer les instructions suivantes dans l'exécuteur et observer le résultat renvoyé. En déduire la signification des opérateurs *,/,//,%, **

```
>>> 25*2          ----------> multiplication
>>> 23/4          ----------> division
>>> 23//4         ----------> division euclidienne
>>> 23%4          ----------> reste entier de la division
>>> 2**2          ----------> puissance
>>> 4**.5         ---------->
```

2. Entrer les instructions suivantes dans l'exécuteur et noter le résultat renvoyé, y compris les messages d'erreur. En déduire la signification des opérateurs =,==, <=, !=, and, or.

```
>>> 1+1=2         ---------->
>>> 2+2==4        ----------> true si le resultat est vrai
```

```
>>> 2+2<=3        ----------> true si le resultat est < ou = a 3
>>> 0!=1          ----------> a différent de b
>>> 0!=1 and 7<5  ----------> et
>>> 0==1 or 7>=5  ----------> ou
```

2.2. Déclarer et utiliser des variables

3. Saisir la suite de commande suivante et indiquer le résultat (remarque : sqrt est la fonction racine carrée):

```
>>> x=12          ----------> print(x) la variable x est mis a 12
>>> z=8           ----------> print(z) la variable z est mis a 8
>>> x             ----------> affiche la valeur de x
```

```
>>> x*z           ----------> multiplication x*z=96
>>> sqrt(x)       ----------> racine carré de 12=3.46
```

2.3. Ajouter une librairie

4. Ajouter la librairie des fonctions mathématiques puis conclure

```
>>> from math import *
>>> sqrt(x)
```

2.4. Les types de variables python

La commande python qui permet de connaitre le type d'une variable **var** est : type(var) .

5. Déclarer puis déterminer le type des variables suivantes :

```
x = 12
```

type de x : entier

```
y = sqrt(x)
```

type de y : float (nombre a virgule flotante)

```
z = 4.5
```

type de z : flaot

```
a="bonjour"
```

type de a : char

```
b='au revoir'
```

type de b : char

On retiendra qu'il existe donc deux façons de déclarer une chaîne de caractères.

```
c = True
```

type de c : bool (booléen)

---

6. Entrer les instructions suivantes et en déduire la fonction des opérateurs int, float et str.

```
>>>  int(14.3)   arrondi
>>>  float(12)   rajoute un .0
>>>  str(147)
>>>  int("42")
```

3. Utilisation de Python par l'exécution de script

Dans l'éditeur IDLE, créer un nouveau script (*Menu File, puis New File*) et enregistrer ce scipt sous le nom script2.py sur votre clef usb.
Un fois rédigé, le programme sera exécuté en cliquant sur Run module dans le menu *Run* ou en appuyant sur *F5*. Le résultat sera automatiquement affiché dansq la fenêtre de l'interpréteur.

3.1. Mise en évidence des erreurs dues aux approximations.

7. Donner la valeur théorique de $\sin(2\pi)$, et $\sin(2k\pi)$

8. Recopier le code ci-dessous et exécuter le programme

```
from math import *
k = 1
alpha = 2*k*pi
print("pour k=%d, sin(2kpi)=%.3f" %(k,sin(alpha)))
```

`%d` permet d'insérer un entier dans une chaîne de caractères, et `%.3f` un nombre à virgule flottante avec trois décimales.

9. Tester la valeur $k = 10^{15}$. Conclure

10. Pour insérer des variables dans une chaine de caractères, il est aussi possible d'utiliser la fonction ***format()***. Tester et commenter l'instruction suivantes :

```
print("pour k={0}, sin(2kpi) = {1:.3}".format(k, sin(alpha)))
```

11. On considère l'équation $y = x + (-x + 1)$. Ecrire un programme qui affiche la valeur de y (commande print(y)) pour $x = 1.0$ puis $x = 1.0 * 10^{16}$. Commenter.

4. Manipulation des chaines de caractères

Une chaîne de caractères (type str) est une suite de caractères ordonnée et non-modifable, délimitée en Python par des apostrophes ou des guillements.

12. La fonction ***len*** permet d'obtenir le nombre d'éléments contenus dans une chaîne de caractères. Dans votre script, déclarer la chaine de caractères "mot" correspondant au mot "grenouille" et afficher sa longueur à l'aide de l'instruction suivante :

```
print(len(mot))
```

13. Chaque caractère d'une chaîne de longueur n peut être désigné pat un index compris entre 0 et $n - 1$. Déclarer les variables suivantes et indiquer le résultat porduit par leur affichage.

```
x=mot[0]
y=mot[2:6]
```

14. Il est possible d'assembler (**concaténer**) plusieurs chaînes de caractères afin d'en construire une plus grande à l'aide de l'opérateur **+**. Concaténer les chaînes x et y définies précédemment et afficher le résultat.

5. Introduction aux listes

Une liste est une collection ordonnée et modifiable d'éléments, qui peuvent être de même type (liste d'entiers, liste de réels, liste de chaîne de caractères...) ou pas. Les éléments sont séparés par des virgules, et l'ensemble est encadré par des crochets.

15. Déclarer les listes suivantes dans votre script, et les afficher avec la fonction **print**, puis utiliser la fonction **len** pour obtenir leur longueur

```
liste1=[1,6,78,42,5]
liste2=['jour', 27, 'lundi', 1]
liste3=[]
```

16. Comme pour les chaînes de caractères, chaque élément d'une liste de longueur n est désigné par un index allant de 0 à n désignant sa place dans la liste. Écrire l'instruction permettant d'afficher le premier et dernier élément de liste1, en utilisant les résultats du paragraphe précédent.

17. Il est possible d'afficher les derniers éléments rapidement sans connaitre le nombre d'éléments de la liste, en utilisant l'instruction liste[-1] pour le dernier, liste[-2] pour l'avant dernier, etc. Ecrire l'instruction permettant d'afficher le dernier élément de liste1, et l'avant dernier de la liste2.

18. Il est possible d'attribuer une nouvelle valeur à un ou plusieurs éléments d'une liste. Attribuer la valeur 0 au premier élément de liste1 puis afficher à nouveau la liste.

19. Il est possible d'extraire simultanément plusieurs éléments d'une liste appelé encore tranche. Déclarer et afficher l'objet suivant et déterminer son type

```
sousliste1=liste1[2:4]
```

20. Afficher la tranche du 2ième élément inclus au 4ième inclus de liste1

21. Il est possible de créer une nouvelle liste ayant un nombre fini de valeurs identiques fixées (utile lors de la création d'une liste de longueur connue dont on changera par la suite ces valeurs). Créer la liste suivante et afficher son contenu

```
n=10
liste4=n*[0]
```

22. Déclarer la liste 3 comme étant le résultat de la concaténation de liste 1 et liste2

23. Il est possible d'ajouter un élément dans une liste par la fonction **append()** comme suit : liste.append(élément à ajouter à la fin). Ajouter à liste2 l'élément "mois" puis l'afficher.

24. Il est possible d'insérer un élément dans une liste à une position précise par la fonction **insert**() en indiquant en premier argument le rang où l'élément doit être inséré : liste.insert(rang, élément à ajouter). Ajouter à liste1 le nombre 18 après 78 puis l'afficher.

25. Il est possible de supprimer un élément de rang p dans une liste par la fonction **del()** comme suit : del(liste[p]).
    - Supprimer à la liste2 l'élément "mois" puis l'afficher.
    - Supprimer la tranche avant le rang 2 de la liste3 puis l'afficher.
    - Vider la liste3.

26. Il est possible de supprimer l'élément dont on connaît la valeur par la fonction **remove()** comme suit : liste.remove(élément à supprimer). Supprimer à la liste1 le chiffre 42 puis l'afficher.

Copie d'une liste :

27. Déclarer liste4=liste1 puis l'afficher.
28. Supprimer de la liste4 le chiffre 6 puis afficher la liste1, conclure.

---

29. Déclarer maintenant liste4=list(liste1)
30. Supprimer de la liste4 le chiffre 78 puis afficher la liste1, puis la liste4. Conclure.

Remarque : il est également possible de copier la liste1 par liste4=liste1[:]