L'objectif de ce TP est de maîtriser l'aspect interactif de Python 3.x

1 Présentation de Python

Python est un langage de programmation développé en langage C, par Guido Van Rossum (du CWI, Centrum voor Wiskunde en Informatica aux Pays Bas) en 1990. Python porte ce nom car Guido Van Rossum est fan des "Monty Python's Flying Circus" une célèbre série comique de la BBC des années 1970. Le site officiel de Python est www.python.org.

Python est libre et gratuit (sous General Public Licence). Il continue à évoluer grâce à de nombreux bénévoles. L'interpréteur principal de Python est maintenu par le créateur du langage.

Les points forts de Python sont : facilité d'apprentissage, portabilité (Unix, Linux, MacOS, MS-DOS, Windows et autres), extensible (une multitude de librairies et de modules par défaut), de haut niveau, puissant (souvent comparé à Tcl, Perl, Scheme ou Java),

Python a peu de mots clés, simple de structure, d'une syntaxe clairement définie, facile à lire et à maintenir.

Il existe des versions de Python pour Windows CE (Pocket PC), Nokia Series 60 , Psion, Nintendo, Sony PlayStation 2, iPod et autres (voir http://www.python.org/download/other/)

Python est utilisé dans de nombreux domaines : informatique (bio-informatique, scripts d'administration système ou d'analyse de fichiers textuels, développement liés au Web, scripts CGI, navigateurs Web, moteurs de recherche, agents intelligents, objets distribués, accès aux bases de données relationnelles, réalisation d'interfaces graphiques utilisateurs etc ...) industrie, sciences, commerce, etc. Il est utilisé dans google, Yahoo, NASA, Lucas/film, Red Hat, Zope etc.

1.1 Galaxie Python

De nombreuse bibliothèques sont développées pour Python dans différents domaines :

`Biopython` : bibliothèque en Python gratuite pour la biologie moléculaire sur ordinateur. (bases de données biologiques, Importation des différents formats (Fasta, GenBank, SwissProt, ...), Alignements multiples: BLAST, ClustalW etc.) voir www.biopython.org.

`Jython` : (anciennement nommé JPython), est un interpréteur Python écrit en Java. Il permet la compilation de code Python en bytecode Java. www.jython.org.

`Pygame` : est une bibliothèque libre multi-plateforme qui facilite le développement de jeux vidéos en Python. www.pygame.org.

`wxPython` : est une bibliothèque libre multi-plateforme en Python utilisée pour la création d'interfaces graphiques. www.wxpython.org.

`PyGTK` : est une bibliothèque en Python pour la gestion des interfaces graphiques d'applications s'appuyant sur la GTK+ écrite en C. voir www.pygtk.org.

1.2 Quelle version utiliser ?

La version 3.x de Python est sortie en décembre 2008. C'est une évolution majeure et une refonte assez profonde de ce langage. Des programmes écrits avec les versions antérieures de Python peuvent être incompatibles avec Python 3.x . C'est la version que nous allons utiliser en TP.

1.3 Installation

Python est disponible sur le site http://www.python.org, pour le télécharger (environ 18 Mo). Son installation sur Mac ou sous Windows est très facile. Souvent il est intégré à Linux donc pas besoin de l'installer.

Pour lancer Python il suffit de taper python à l'invite du shell de Linux. Quand Python est installé sous Windows, on le lance en allant Démarrer, programmes, python puis IDLE python (GUI).

Quand l'interpréteur fonctionne en mode interactif, son prompt principal est : >>>; pour la suite, il attend avec le prompt secondaire trois points par défaut ··· ; nécessaires lorsqu'on saisit une construction sur plusieurs lignes.

Important : Lisez ce qui suit et tester. Ne vous contentez pas uniquement de ces exemples, en essayez d'autres, soyez imaginatifs ! C'est en pratiquant qu'on retiendra les commandes et la syntaxe de Python. Comprenez et notez le rôle de chaque commande.

Erreur : Quand il y a erreur, l'interpréteur affiche un message d'erreur, qu'il faut toujours lire.

2 Affectation

Une variable sert à mémoriser une information (un nombre, une chaîne de caractères, etc.). En Python, on n'a pas besoin de déclarer une variable avant de s'en servir. Le signe égale = est utilisé pour affecter une valeur à une variable. En python, la déclaration d'une variable et son initialisation (c.à.d. la première valeur que l'on va stocker dedans) se fait en même temps. De plus Python reconnaît automatiquement le type de chaque variable.

Pour afficher le contenu d'une variable x, il suffit d'évoquer x après le >>> ou d'utiliser la fonction print(x).

Question 1. Affecter 5 à x et afficher sa valeur et son type à l'aide de la fonction type().

Question 2. Affectation multiple : a=b=12. Afficher a et b et leur type.

Question 3. Affectation parallèle : a,b=2,-3. Afficher a et b. Affecter à e et pi 2.72 et 3.14 respectivement et afficher leur type.

3 Nombres

3.1 Nombres réels

Les nombres réels sont de types 'float'. S'écrivent 2.34, 1.034E-4, 5.67E32, 8.1e+232 jusqu'à approximativement $1e \pm 308$.

Question 4. Tester ces écritures.

Question 5. Affecter à x et y respectivement 5 et 5. ou 5.0 . Afficher les types de x et de y.

Question 6. Tester et expliquer int() et float().

Question 7. Tester les opérateurs arithmétiques : +, -, *, **, /, //, %

Question 8. Tester les opérateurs de comparaison : ==, >, <, <=, >=. Par exemple : 4==5.

3.2 Nombres complexes

Question 9. Les nombres complexes sont représentés sous la forme : a+bj ou a+bJ : $z = 2 + 1j$ ou $z = 1 + 1J$, $z.real$ et $z.imag$ pour extraire la partie réelle et imaginaire de z respectivement. La fonction abs() calcule le module de z ou la valeur absolue de

nombres réelle.

Question 10. Affecter à z un nombre complexe. Afficher le type de z. Tester sur z , .real, .imag et abs().

3.3 Nombres binaires

Les nombres en base binaire commencent par `0b` ou `0B`. `0B1110` vaut en base 10 : 14.

Question 11. Convertir en base 10 les nombres binaires : 11011, 11101 et 110011,

3.4 Nombres octaux

Les nombres en base octale commencent par `0o` ou `0O` et ne contiennent que des chiffres inférieurs à 8. `0o33` vaut en base 10 : $3 * 8 + 3 = 27$.

Question 12. Convertir en base 10 les nombres octaux : 1237, 705, 602.

3.5 Nombres hexadécimaux

Les nombres en base hexadécimale commencent par `0x` ou `0X` . Par exemple 0x2a vaut en base 10 : $2 \times 16 + 10 = 42$.

Question 13. Convertir en base 10 les nombres hexadécimaux : A, B, F, FF00FF,
Fonctions :
bin(n) : convertit le nombre entier n (écrit en base 10) en base binaire
oct(n) : convertit le nombre entier n (écrit en base 10) en base octale.
hex(n) : convertit le nombre entier n (écrit en base 10) en base hexadécimale.

Question 14. Écrire en base binaire, octale et hexadécimale les nombres 23, 65, 75, 15987.

3.6 Fonction round()

round(x) : arrondit un nombre réel (vers l'entier supérieur à partir de .5).
round(x,n) : arrondit un nombre réel à la décimale n (précision à la 15e décimale).

Question 15. Tester ces 2 fonctions.

4 Chaîne de caractères

Une chaîne de caractères (type string) est toujours déclarée entre apostrophe ' ' ou entre guillemets anglaises double " ". On ne peut jamais utiliser un apostrophe entre apostrophes ou une guillemet entre des guillemets.

c1 = " " : chaîne vide - ne contient rien
c2 = "spam's" : utilisation des guillemets - permet l'apostrophe
bloc = """...""" : triple guillemets - texte sur plusieurs lignes

4.1 Opérations sur les chaînes

L'opérateur + permet de concaténer 2 chaînes de caractères, et l'opérateur * permet de dupliquer n fois une chaîne.

Question 16. Concaténer 'extra' et 'ordinaire'.

Question 17. Afficher 1000 fois 'Python est génial'.

Question 18. Tester """...""".

Question 19. Affecter 'Python est facile !' à C et tester C[i], C[i:j], C[i:], C[:j] pour i, j des entiers nuls, positifs ou négatifs.

Question 20. Tester :
'y' in C : retourne vrai ou faux.
'x' not in C : vrai si C ne contient pas 'x'.

4.2 Fonctions sur les chaînes

syntaxe : `fonction(argument)`
len(ch) : renvoie la longueur de la chaîne ch.
int(ch) : convertit la chaîne ch en un nombre entier quand c'est possible.
float(ch) : convertit la chaîne ch en un nombre réel quand c'est possible.

Question 21. Donner des exemples avec ces fonctions.

4.3 Méthodes sur les chaînes

syntaxe : `chaîne.méthode()`

Question 22. Tester les méthodes suivantes.

rstrip([chars]) : enlève les caractères 'chars' à la fin de la chaîne. Si on ne définit pas de caractères, les espaces blancs éventuels seront enlevés.
lstrip([chars]) : enlève les caractères 'chars' au début de la chaîne. Si on ne définit pas de caractères, les espaces blancs seront enlevés.
isdigit() : vraie si ch ne contient que des chiffres.
isalpha() : vraie si ch ne contient que de des lettres.
isalphanum() : vraie si ch ne contient que des chiffres et des lettres.
count(ch,i,j) : compte le nombre de fois que ch est répété dans l'intervalle [i,j]
find(ch,i,j) : retourne la position de la première occurrence de ch dans l'intervalle [i,j].
find(sch) : cherche la position d'une sous-chaîne sch dans une chaîne.
count(sch) : compte le nombre de sous-chaînes sch dans une chaîne.
lower() : convertit une chaîne en minuscule.
upper() : convertit une chaîne en majuscule.
capitalize() : convertit en majuscule la première lettre d'une chaîne.
swapcase() : convertit toutes les majuscules en minuscules et vice-versa.
replace(c1, c2) : remplace tous les caractères c1 par des caractères c2 dans la chaîne.
index(c) : retrouve l'index de la première occurrence du caractère c dans la chaîne.

4.4 Caractères spéciaux

Question 23. Tester les caractères spéciaux suivants en utilisant la fonction print().

+ : concatène de deux chaînes.
\' : permet d'insérer une apostrophe dans une chaîne délimitée par des apostrophes.
\" : permet d'insérer " dans une chaîne délimitée par des ". \n : provoque un saut à la ligne dans une chaîne.
\0 : pour un caractère vide.
\t : pour une tabulation.
\b : pour revenir en arrière d'un caractère.
\r : pour revenir en début de ligne sans passer à la ligne suivante.
\v : pour passer à la ligne suivante sans revenir en début de ligne.
\\ : permet de coder le caractère \.

5 Fonctions print() et input()

5.1 Fonctions print()

La fonction print() permet l'affichage sur écran.

```
>>> nom='Ali'
>>> age=23
>>> print("l'age de", nom, "est", age)
```

Question 24. Expliquez la différence entre l'affichage par la fonction print() et par la frappe du nom d'une variable suivi de la touche entrée.

5.2 Fonction input()

provoque une interruption du programme et invite l'utilisateur à entrer des données au clavier et à terminer avec Entrée. Entre parenthèses on explique à l'utilisateur ce qu'il doit faire. Elle renvoie une valeur dont le type est celui saisi par l'utilisateur.

```
>>> prenom = input('Entrez votre Prénom (entre guille
Entrez votre prénom (entre guillemets) : ali
>>> print('votre prénom est', prenom)
votre prénom est ali
```

6 Listes

Une liste est une structure de données qui contient une série de valeurs séparées par virgules et imitée par des crochets. Par exemple : filieres = ['SM','SMI','SVI','STU','SMP','SMC']

On peut rappeler ses éléments par leur indices (numéro de position). l'indice d'une liste de n éléments commence à 0 et se termine à n-1.

Une liste peut également être indexée avec des nombres négatifs selon le modèle suivant :

```
liste         : ['a', 'b', 'c', 'd']
index positif :   0    1    2    3
index négatif :  -4   -3   -2   -1
```

Question 25. Tester : type(filieres) , len(filieres), , filieres[i] , filieres[:j] , filieres[i:] , filieres[i:j] où i, j sont des entiers relatifs.

A la différence des chaînes, qui sont non modifiables, il est possible de changer les éléments individuels d'une liste.

Question 26. Pour modifier le i eme terme d'une liste, il suffit de lui affecter une nouvelle valeur : filieres[2]='SVT'.

Question 27. concaténer deux listes, reproduire 20 fois la liste L=[1,2,3].

6.1 Opérations sur les listes

Question 28. Lire la description et appliquer chacune des opérations suivantes sur plusieurs exemples de votre choix.

L = [16, 'mot',['3','ab']] : Crée une liste d'éléments mixtes
L[i] : Retourne le ie élément de la liste après le 1er élément
L[i][j] : Retourne une élément d'une liste multidimensionelle
L[i:j] : Retourne la tranche d'éléments jusqu'au je élément
len(L) : Retourne le nombre d'éléments de la liste
L + l : Concaténation de deux listes
3 *L : Concaténation multiple de la même liste
x in L : Opération logique retourne true si x est un élément de la liste, sinon false.
L.append(x) : ajoute l'élément x à la fin de la liste
del L[j] : Élimine l'élément j de la liste
L[i:j]=[] : Élimine la tranche i:j de la liste
L.insert(2,x) : insère x en 2eme position
L.sort() : Ordonne les éléments en ordre ascendant
L.index(x) : Retourne la position du premier élément de valeur x
L.reverse() : Réorganise les éléments en commençant par la fin
L[i]='ceci' : Assigne une valeur à l'élément i
L[i:j]=[2,3,7] : Assigne une valeur à la tranche i:j
range(n) : Crée la liste de n éléments [0,1,2, ..., n-1], pour l'afficher list(range()) .
split() : : décompose une chaîne contenant des espaces en une liste de sous-chaînes.
'X'.join(liste) : : rassemble les éléments de liste en les séparant par la chaîne X.

La fonction list() Transforme une chaîne de caractère en liste.

Question 29. Tester range(n), range(n,m), range(n,m,s) où m, n et s sont des entiers. Pour afficher une liste créée par range() on écrit list(range()).

7 Tuples

Un tuple est une collection d'éléments **ordonnés**, séparés par des virgules, et entourés par des parenthèses (ou non). Il est semblable aux listes mais **il n'est pas modifiable**. Les tuples n'ont pas de méthodes. On ne peut alors ajouter ou enlever un élément dans un tuple. Les tuples sont plus rapides que les listes. Sont utiles pour définir un ensemble constant de valeurs.

Question 30. Définir un tuple et afficher son type.

Les tuples peuvent être convertis en listes et vice-versa. La fonction tuple() : transforme une séquence (liste, chaîne) en tuple.
La fonction list() : transforme un tuple en une liste.

Question 31. Tester list() et tuple().

Question 32. Tester

```
>>> t= 'a', 'b', 'c', 'd', 2
>>> t.append("e")
>>> t.remove("b")
>>> t.index("d")
>>> 'c' in t
```

Les autres opérations sur les tuples sont syntaxiquement similaires à celles que l'on effectue sur les listes, (testez les sur les tuples).

```
>>> print(t[1:3])
>>> T = ('e',) + t[1:]
```

8 Dictionnaires

Les dictionnaires sont très pratiques lorsque l'on a des structures complexes à décrire. Ils sont des

collections d'objets **non ordonnées**. On accède aux valeurs d'un dictionnaire par des clés de type chaîne de caractère. (alors qu'on accède aux valeurs d'une liste par indices de type numérique). Un dictionnaire est de la forme
`D = { clé1 : valeur1, clé2 : valeur2, ... }` avec la contrainte que les clés soient uniques. Exemples :

```
>>> d={"tomate":5, "pomme":12}
ou
>>> D = {}
>>> D['annee']=2003
>>> D['style'] = 'fiction'
>>> D['titre'] = 'Cosmos 2030'
>>> D['auteur'] = 'E. S. Sami'
>>> D[120] = 'adresse'
>>> D
>>> D['auteur']
```

Question 33. Affichez le type de D.

8.1 Méthodes

Question 34. Tester les méthodes suivantes.

keys() : renvoie les clés d'un dictionnaire.
values() : renvoie les valeurs d'un dictionnaire.
clear() : supprime le contenu du dictionnaire.
get(cle) : renvoie la valeur associée à la clé, `None` si la clé n'existe pas.
get(cle, exp) : renvoie l'expression si la valeur n'existe pas.
items() : crée une liste de tuples (clé,valeur).
vars() : est un dictionnaire des variables: valeurs.
globals() : renvoie un dictionnaire des variables globales uniquement.
locals() : renvoie un dictionnaire des variables locales uniquement.

8.2 Fonctions

Question 35. Tester les fonctions suivantes.

del : supprimer une entrée.
len : renvoie la longueur du dictionnaire.

Question 36. Découvrir et comprendre l'utilisation du Help de Python.