VADEMECUM POUR PYTHON

1 Instructions élémentaires

Affectation

```
x=2
x,y=1,12.4 # affectation en couple
x+=1       # incrémentation de 1
x,y=y,x    # permutation de variables
type(a)    # type de a
```

Opérations élémentaires numériques

```
5**2 # 5 puissance 2
5/2  # 5 divisé par 2
5//2 # quotient de 5 par 2
5%2  # reste de la division de 5 par 2
```

Affichage

Les guillemets pour le texte peuvent être simples ou doubles.

```
print('la valeur de a est ',a)
print('la valeur de a est ',a, end='')
# -> pas de retour chariot
```

Nombres complexes

```
z=3+5j
print(z.real); print(z.imag)
```

2 Listes

⚠ Les listes sont numérotées de 0 à $n-1$, n : longueur de la liste.

```
liste=[1,2.0,"trois",4,5,6,7,'huit']
liste=[]              #liste vide
liste=list('texte') #transforme texte en
    liste
liste=list(range(0,10)) #liste de 0 à 9
len(liste)            #longueur de la liste
```

Slicing

```
liste[0]    # élément 0 de la liste
liste[-1]   # dernier élément
liste[1:4]  # éléments de 1 à 4 (exclu)
liste[3:])  # éléments à partir de 3 (le 4ème
    )
liste[:4])  # éléments jusqu'à 4 (exclu)
liste[0:4:2]# en comptant de 2 en 2
liste[::-1] # liste miroir
```

Modification en place de la liste La variable de la liste est modifiée

```
liste.append(a) # ajouter élément a en fin
liste.pop()     # enlève et renvoie le
    dernier
liste.remove(a) # retire 1ère occurrence de a
liste.sort()    # ordonne la liste
liste.reverse() # retourne la liste
liste[0:2]=['a',2]# modifie les deux 1ers é
    lts
liste[2:2]=[1,2,3]# insère liste en indice 2
del liste[1]      # supprime 1er élément
```

Chercher dans les listes

```
a in liste       # teste l'appartenance
liste.index(a)   # indice 1ère occurrence de a
liste.count(a)   # nombre d'occurrences de a
```

Opérations sur les listes Ne modifie pas la liste.

```
liste+[4,5]   # ajouter une liste au bout
2*liste       # 2 fois la liste bout à bout
```

Difficultés d'affectations entre listes

```
copie=liste
#pointent vers la même liste mémoire
copie=liste.copy()  # 'vraie' copie
copie2=liste[:]     # idem
```

Listes en compréhension

```
[x**2 for x in range(1,100)]
# liste des carrés des entiers de 1 à 99
[x**2 for x in range(1,100) if x%2==1]
# liste des carrés des entiers impairs de 1 à
    99
[x*y for x in range(1,10) for y in range(3,5)
    ]
#l'ordre des for est important
```

3 Chaines de caractères

Comme les listes mais non mutables, le slicing est identique.

```
chaine="bonjour"
chaine=='bonjour'        #True
"bonjour"+" Python"       #concatène
chaine+=" Python"
phrase='J\'aime Python' #échappement

# Questionner l'utilisateur
a=input("Entrez un entier : ")
# a est une chaîne de caractères
int(a)   #transforme en nombre entier
```

Méthodes sur les chaînes

```
chaine.find('jour') #renvoie l'indice ou -1
chaine.split('o')   #coupe sur les 'o' ->
    liste
chaine.replace('jour','soir ')
chaine.upper()       #met en majuscules
chaine.lower()       #met en minuscules
```

Tests sur les chaînes

```
chaine.isnumeric()  #teste si c'est un nombre
chaine.endswith('d')#teste si termine par d
"jour" in "Bonjour"#teste l'appartenance
```

etc...et d'autres dans la documentation.

4 Structures conditionnelles et boucles

Pour l'indentation, utiliser des espaces (et non des tabulations)

Pour arrêter une boucle : **break**
Passer directement à l'itération suivante : **continue**
Pour... ne rien faire : **pass**

Structure conditionnelle
Les conditions `elif` et `else` sont facultatives

```
x=5
if x==0:
  print('x est nul')
elif x>0:
  print('x est strictt positif')
else:
  print('x est strictt négatif')
print('fin')
```

Boucles

```
x=5
while x>0:
  x-=1 # on retranche 1
  print(x)

#de 1 à 5 (exlu)
for i in range(1,5):
  print(i)

#listes et chaines sont itérables
liste=['zero','un','deux']
for i in liste:
  print(i)
for i in "bonjour"
  print(i)
```

5 Fonctions

Après `return`, le programme n'exécute plus rien dans la fonction.
⚠ `return` au milieu d'une boucle provoque son arrêt.

```
def somme(i,j=1):
  # 1= valeur par défaut de j
  return i+j # valeur renvoyée
somme(3,9)
somme(j=3,i=5) #change l'ordre
somme(3)  #utilise valeur par défaut

#incrémente la variable gloable a
def incremente(incr):
  global a
  a+=incr
```

Documenter ses fonctions par triples guillemets

```
def somme(i,j=1):
  '''somme(i,j) fait la somme de i et j
    par défaut, j=1'''
  return i+j # valeur renvoyée
help(somme) #donne l'aide
```

6 Bibliothèques mathématiques

```
from pylab import *
```

Pseudo-aléatoire Différents de ceux de la bibliothèque `random`

```
random() #nbre aléatoire entre 0 et 1 (exclu)
randrange(3,10) #entier aléatoire entre 3 et
    9
randrange(10) #entier aléatoire entre 0 et 9
randrange(0,11,2) #entier aléatoire pair
    entre 0 et 10
shuffle(liste)  #mélange une liste (sur place
    )
```

Tableaux

```
#tableau à 2 dimensions
a = array([[1, 2, 3], [4, 5, 6]])
a[0,1] #renvoie 2

#tableau avec 100 entrées réparties de 0 à pi
x=linspace(0,pi,100)
y=cos(x) # tableau des cosinus des élts de x
```

Graphiques

```
#y en fonction de x (avec x, y des tableaux)
plot(x, y)

#quelques options
plot(x, y2, 'b-o', label='sin(x)')
xlabel("abscisses")
ylabel("ordonnees")
legend()
xlim(-1, 3*pi)
ylim(-1.4, 1.4)
show() #affichage du graphique
```

7 Manipulation de fichiers

```
f=open('C:/Documents/toto.txt', 'r+')
f.read() #lit depuis le point courant
f.read(n)         # lit n caractères
f.readline()      #ligne courante
f.readlines()     #liste des lignes
f.write("texte") #écrit
f.close()         #ferme le fichier
f.tell()          #position dans le fichier
f.seek(0)         #se place en tête de fichier
```