Mémento Python

Types de variables

a = True	*# type bool, vaut True ou False*
b = 2	*# type int (entier)*
c = 3.14	*# type float (décimal)*
d = "Bonjour"	# type str (chaîne de caractères)
e = ["un",2 ,3]	# type list (liste)

Opérations sur les int et float

a // b	*quotient de la division euclidienne*
a % b	*reste de la division euclidienne*
ab**	*a exposant b*
abs(a)	*valeur absolue*
round(a,n)	*valeur approchée de a à 10^{-n} près.*

Affectation

Affecter, c'est associer une valeur à une variable.

1. *évaluation du membre de droite .*
2. *affectation de cette valeur à la variable indiquée à gauche du symbole =.*

Score = 10	*# On associe la valeur 10 à score*
x = 1.2*sin(y)	*# Python évalue d'abord y puis sin(y) puis le produit par 1.2*
a, b = 2, 3	*# veut dire a ← 2 et b ← 3*
a += 1	*# veut dire a ← a + 1*
b /= 2	*# veut dire b ← b / 2*

Fonctions

def nom_de_fonction(arguments) :

- La fonction est indentée,
- le résultat est retourné avec **return**.
- On parle d'arguments ou paramètres.

```
def  f(a, b):
    return   a*b
```

On appelle la fonction naturellement :

```
1. p = f(1,2)
2. print(f(1 ,2))
```

Les chaînes (str) et les listes (list)

En python une chaîne de caractères peut être considérée comme étant une liste de caractères.

Le premier élément d'une liste à pour indice 0

Soit a une variable de type str ou list :

n = **len(a)**	*# donne la longueur de a*

*On accède aux éléments de a par : **a[0], … , a[n-1]***

a[0]	*# premier élément*
a[n-1] ou a[-1]	*# dernier élément*
a[p:q]	*retourne a[p],… , a[q-1]*
a[p:]	*retourne a[p], …, a[n-1]*
a[:p]	*retourne a[0], … , a[p-1]*

Modification des listes

a = [] **a = list()**	*# crée une liste vide*
a = a + [elt] **a.append(elt)**	*# ajoute elt à a*
a.insert(i,x)	*# insertion de x à la position i*
del a[k]	*# supprime l'elt d'indice k de a*
a.remove(elt)	*# supprime la première* *# occurence de elt dans a*
a=str.split()	*# converti la chaîne str* *# en liste de mots*
str =" ".join(a)	*# converti a en une chaîne avec* *# des espaces comme séparateurs.*
a.sort()	*# Tri la liste a*

Recherche dans une liste

elt in a	*# test si elem se trouve dans a*
a.count(elt)	*# renvoie le nombre d'elt dans a*
a.index(elt)	*# Renvoie l'index de la première* *# occurence de elt dans a*

Listes en compréhension

```
lst = [2*i for i in range(10)]
lst = [i for i in range(20) if i%2 == 0]
```

résultat → lst = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

Concaténation de chaînes

La concatenation de chaîne consiste à réunir les chaînes concaténées en une seule.

s = "un"+" et "+deux"	*# s = "un et deux"*

t-uples

un tuple est une liste non modifiable.
On peut , par exemple, l'utiliser dans une fonction qui doit retourner plusieurs valeurs.

a = (1,2,3)	*# défini le t-uple (1,2,3)*

```
def min_max(a,b):
    if a < b:
        return (a,b)
    else:
        return (b,a)
```

Dictionnaires

Un dictionnaire est une liste de (clé,valeur)
d = {"nom":"Dupont" , "age":10 , "sexe":"F" }

d["nom"]	*# donne "Dupont"*
d.keys()	*# renvoie une liste des clés*
d.values()	*# renvoie une liste des valeurs*
d.items()	*# renvoie une liste de tuple* *# (clé,valeur)*
d.get("nom")	*# renvoie "Dupont" ou None si la clé* *# n'existe pas*

d.get("nom","inconnue") *#permet de retourner une valeur spécifique à la place de none, ici "inconnue"*

Mémento Python

Import de librairies (modules)

```
from math import *          # librairie math
```
Principales fonctions : **sqrt, cos, pi...**

```
from random import *        # librairie random :
randint(a,b)                # entier dans [a, b ]
choice(maListe)             # élément de la liste
Shuffe(a)                   # mélange de la liste a
```

Tests conditionnels (Conditions)

Python évalue si une affirmation est vraie ou fausse il lui associe la valeur booléenne True ou False. Pour écrire un test on dispose des opérateurs :

```
<,>,<= et >=        # comparer
==                  # tester l'égalité
!=                  # tester la différence
in                  # tester l'appartenance à une liste ou une chaîne.
and, or et not      # les opérateurs logiques
```

Quelques exemples :

```
3 <=4                   # True
"Paul" == " Georges"    # False
"e" in "Fred"           # True
1 not in [1 ,2 ,4]      # False
```

Instruction conditionnelle (alternative)

Executer une ou des actions suivant qu'un test conditionnel est validée ou pas.

Différentes variantes existes :

(1)
```
if <condition>:
    <instruction(s)>
```

(2)
```
if <condition>:
    <instruction(s)>
else:
    <instruction(s)>
```

(3)
```
if <condition>:
    <instruction(s)>
elif <condition>:          # else if
    <instruction(s)>
elif <condition>:          # else if
    <instruction(s)>
else:                      # le else n'est pas
    <instruction(s)>       # obligatoire
```

Exemple :

```
def mention(note):
    if note<12:
        return "passable"
    elif note<14:
        return "assez bien"
    elif note<16:
        return "bien"
    else:
        return "très bien"
```

Boucle for (bornée)

Boucle « classique » :

```
for i in range(101):  # i parcourt {0;...;100}
    print(i)
```

Boucle énumérative :

```
liste = [1 ,2 ,1 ,3 ,1 ,4]
for nombre in liste:        # nombre parcourt la liste
    print(nombre)
```

Boucle énumérative avec index:

```
liste = [1 ,2 ,1 ,3 ,1 ,4]
for idx, val in enumerate(liste):
    print( idx,val)         # val parcourt les valeurs
                            # idx les indices associés
```

Boucle avec range inversé :

```
somme = 0
for i in reversed(range(10)):  # i parcourt {9;... ;0}
    somme = somme + i
print(somme)                   # 5050
```

range(a) tous les entiers de [0 ; a[
range(a,b) tous les entiers de [a ; b[
range(a,b,p) tous les entiers de [a ; b[de p en p

Boucle while (non bornée)

On répète un bloc d'instructions tant qu'une condition est vérifiée.

```
u = 5
while u < 1000:
    u = 2 * u - 1
    print(u)
```

Affichage à l'écran

```
a = 18                          # a prend la valeur 18
print("Vous avez ", a, "ans.")  # permet d' afficher un message composé.
```

```
print("left = {}, right = {}".format(left, right))
```
permet d'afficher les valeurs des variables left et right aux endroits de la chaîne indiqués par les symboles { }

Saisie de données

```
a = input("Entrez votre nom :")
```
 # a est par défaut de type chaine de caractères

```
b = int(input("Entrez votre âge :"))
```
 # pour que b soit un entier

```
c = float(input(" hauteur de l'arbre :"))
```
 # pour que c soit un decimal