Memento Python

Manipulation de listes

list1 = [ ] : créé une liste vide *list1*

list2 =[1,'a',[1,2]] : créé *list2* une liste à 3 éléments

list3=[1]*10 : créé *list3* qui contient 10 éléments égaux à 1

list4 = [i for i in range(5)] équivalent à **list4=[0,1,2,3,4]**

list1[i] : renvoie l'élément de *list1* à l'index i (i>=0). Si i<0, on part de la fin de la liste (list1[-1] pour le dernier, list1[-2] pour l'avant dernier…)

len(list1) : renvoie le nombre d'élément de *list1*

Méthodes sur les listes

list1.append(x) : ajoute l'élément x à la fin de la liste *list1*

list1.append(list2) : ajoute tous les éléments de *list2* à la fin de *list1*

list1.pop(i) : enlève de *list1* l'élément situé à la position indiquée et le renvoie en valeur de retour.

list1.insert(i, x) : insère un élément à la position indiquée (i est la position de l'élément avant lequel l'insertion doit s'effectuer).

list1.remove(x) : supprime de *list1* le premier élément dont la valeur est égale à x.

list1.clear() : supprime tous les éléments de *list1*.

list1.index(x[, start[, end]]) : renvoie la position du premier élément de *list1* dont la valeur égale x (en commençant par zéro).

list1.count(x) : renvoie le nombre d'éléments ayant la valeur x dans *list1*.

list1.extend(list2) : concatène *list2* à la suite de *list1*.

list1.copy() : renvoie une copie superficielle de *list1*

list(list1) : renvoie une copie profonde de *list1* (clone)

Dictionnaires

dico1 = { } : Créé un dictionnaire vide

dico2= {'a':1,2:'oui','c':[1,2]} : créé un dictionnaire à 3 clés 'a', 2 et 'c'. Ces clés permettant d'accéder aux valeurs du dictionnaire : 1, 'oui' et [1,2].

dico2['a'] : renvoie la valeur correspondante (ici 1) ou une erreur si la clé 'a' n'est pas dans le dictionnaire.

L'ajout d'une nouvelle clé et de sa valeur, ou la modification de la valeur d'une clé se fait par simple affectation :

- **dico1['a']=3 donnerait {'a':3} pour dico1**
- **dico1['a']=[2,1] donnerait {'a':[2,1]} pour dico1**

del dico2['a'] : supprime une association dans *dico2* (erreur si la clé n'existe pas)

'a' in dico2 : vérifie si la clé 'a' se trouve (True) ou non (False) dans *dico2*

list(dico2) : renvoie la liste des clés de *dico2*, ici ['a',2,'c']

Types

type() : pour connaître le type d'une variable
int() : permet la transformation en un entier
float() : permet la transformation en flottant
str() : permet la transformation en flottant

Procédure - Fonction

Fonction : renvoie un résultat par l'intstruction return

```
def nom_fonction(paramètres)
    Instructions
    return résultat
```

Procédure : ne renvoie pas de résultat (pas de return)

```
def nom_proc(paramètres)
    Instructions
```

Valeurs particulières

True : vrai
False : faux

float('inf') : +∞

Instructions conditionnelles

```
if condition:
    instructions

if condition:
    instructions
else
    instructions
```

```
if condition:
    instructions
elif condition2:
    instructions
else:
    instructions
```

Itération for

```
for i in list_ou_dico:
    instructions
```

Parcours des couples d'un dictionnaire (items)

```
for (i,j) in dic.items():
    print(i,j)
for var in range (deb, fin, pas):
    instructions
```

Boucle Tant que

```
While condition :
    instructions
```

Range

range(a) : séquence des valeurs [0, a[
range (b,c) : séquence des valeurs [b, c[(pas de 1)
range (b, c, g) : idem avec un pas de g
range(b,c,-1) : valeurs de b(incl.) à c (excl.) , pas -1

Arithmétiques

+ : addition **-**: soustraction
* : multiplication ******: puissance
/ : division
//: division euclidienne(quotient),
% : reste de la division euclicienne

Logiques

and : et
or : ou
not : négation

Comparaison

== égalité **!=** différence
< inférieur **<=** inférieur ou égal
> supérieur **>=** supérieur ou égal

Ensemblistes

in : appartient
not in : n'appartient pas

Texte

len(text) : renvoie la longueur de la chaîne *text*

text1 + text2 : concatène les chaînes *text1* et *text2*

text* n : répétition de n fois de la chaîne *text*