1

Partie 2

2

2

Plan

- Les listes
- Les tuples
- Les ranges
- Les slices
- Les dictionnaires
- Les sets

3

3

Les listes

4

4

Les listes

- Une liste est une collection ordonnée et modifiable d'éléments éventuellement hétérogènes (pas nécessairement du même type).
- Les éléments de la liste sont indexés, le premier élément a un index [0].
- Il est possible de spécifier une gamme d'indices en précisant où commencer et où terminer la gamme. La valeur de retour sera une nouvelle liste avec les éléments spécifiés.
- Pour déterminer si un élément est présent dans une liste. Il faut utiliser (in)

5

Les listes – Méthodes :

list= ['Java','Python','C#']

Méthode	Description
list.insert(2,"sql")	Ajoute un élément à l'index spécifié
list.append("J2ee")	Ajoute un élement à la fin de la liste
list.extend(list2)	Ajoute des éléments d'une liste (list2) à la liste actuelle (list)
list.remove("laravel")	Supprime l'élément spécifié de la liste actuelle.
list.pop(1)	Supprime l'élément à l'index spécifié
list.clear()	Vide la liste
list.sort()	Trie la liste de manière alphanumérique, en ordre croissant, par défaut. En ordre décroissant **list.sort(reverse=True)**
list.reverse()	Inverse l'ordre actuel des éléments
list2=list.copy()	Créée une copie de la liste (list) dans la liste (list2)

Fonctions: len(list) retourne la longueur de la liste (list)

6

Les listes

Opérations sur une liste

```
l1[0]
len(l1)
l1[1:]
l1[0:2]
l1[:2]
l1 += [4, 5]
del l1[3]
l1[1] = 'x'
l1.append('x')
l1.count('x')
l1.insert(1, True)
l1.remove('x')
```

Création d'une liste

```
l1 = [ 1, 2, 3]      # liste [1, 2, 3]
l2 = []              # une liste vide
```


7

Les listes

Exercices

Indiquer, après l'exécution de chaque ligne, la valeur de la liste l.

```
l = []
l.append(2)
l.insert(0, 4)
l.insert(2, 1)
l[1] = 'deux'
l[2] /= l[2]
l.count(1)
l[0], l[1] = l[1], l[0]
```


8

Les tuples

9

Les tuples

Un tuple (n-uplet) permet d'assembler plusieurs objets dans une séquence immuable.

```
t = (1, 'premier', 10.5) # t est un tuple composé de 3 objets
a, b, c = t              # déstructurer le tuple : a == 1, b == 'premier', c == 10.5

t[0]                     # 1
t[1]                     # 'premier'
t[2]                     # 10.5
t[-1]                    # 10.5

t = (1, )                # tuple avec un seul élément (virgule obligatoire)
```

On peut construire un tuple à partir d'une séquence

```
tuple('abc')             # ('a', 'b', 'c')
tuple([1, 'X'])          # (1, 'X')
```

Tuple ou liste ? Les deux sont des séquences (donc opérations communes) mais
liste modifiable et tuple immuable
tuple : les éléments ont un sens en fonction de leur position

10

Les ranges

11

Les ranges

Un range permet de définir des séquences immuables d'entiers. Il est généralement utilisé pour les répétitions (for).

Appels possibles :

```
range(stop) -> range object    # range(10) -> des chiffres de 0 à 9
range(start, stop[, step]) -> range object
```

Quelques exemples :

```
r1 = range(4)
r1                       # range(0, 4)
list(r1)                 # [0, 1, 2, 3]
tuple(r1)                # (0, 1, 2, 3)
r1[-1]                   # 3

list(range(4, 8))        # [4, 5, 6, 7]
list(range(2, 10, 2))    # [2, 4, 6, 8]
list(range(5, 2, -1))    # [5, 4, 3]
```


12

Les slices

Les slices

Motivation : Référencer une partie des objets contenus dans une séquence.
Les éléments de séquence de l'indice début inclus, à l'indice fin exclu avec un pas.
Les indices peuvent être positifs (0 pour le premier élément) ou négatifs (-1 pour le dernier).

Exemples :

```
s = list(range(10))
s[2:4]
s[2:]
s[:-4]
s[2::4]
s[8:2:-2]
del s[2::2]
```

En particulier, s[::-1] renvoie la séquence complète inversée.

Les slices

Exercices

1. Étant donnée une liste l, comment désigner la liste de tous les éléments sauf le premier et le dernier ?
   Exemple : l = [2, 3, 4, 5] donne [3, 4]
2. Étant donnée une liste l, comment obtenir deux listes, la première qui contient les éléments d'indice pair et la seconde les éléments d'indice impair ?
   Exemple : l = [-5, 2, 1, 18, 0] donne [-5, 1, 0] et [2, 18]

Solution

1)

```
liste[1:-1]
```

2)

```
liste_indices_pairs = liste[0::2]
liste_indices_impairs = liste[1::2]
```


Les dictionnaires

Les dictionnaires

- Collection d'objets s'appuyant sur le mécanisme associatif clé:valeur
- Les dictionnaires Python sont des objets modifiables qui peuvent changer leurs valeurs
- Pour déterminer si une clé est présente dans un dictionnaire. Il faut utiliser (in) s

```
for key, value in dict.items():
    print(key, value)
for key in dict:
    print(key)
for value in dict.values():
    print(value)
```

```
moy = {'python': 17.5, 'anglais': 15.8}
print(moy)
#acces
print(moy['anglais'])
print(moy.get('anglais'))
#return liste des clés
print(moy.keys())
#return liste des valeurs
print(moy.values())
#return nombre d'élement
print(len(moy))
#detecter présence d'une clé
if 'français' in moy:
    print("c'est une clé")
else:
    print("cette clé n'existe pas")
#ajouter un élement
moy['français']=14
#modification
moy['anglais']=13.5
```

17

17

Les dictionnaires

- moy = {"python": 17.5, "anglais": 15.8}

Méthode	Description
moy.update({'anglais':16})	Elle met à jour le dictionnaire avec les éléments spécifié comme argument . Si l'élément n'existe pas, il sera ajouté.
moy.pop('anglais')	Supprime l'élément à la clé spécifié.
moy.popitem()	Supprime le dernier élément inséré
moy.clear()	Vide le dictionnaire.
moy2=moy.copy()	Créée une copie du dictionnaire (moy) dans le dictionnaire (moy2)

- Fonctions: **len(moy)**retourne la longueur du dictionnaire(moy)

18

18

Les sets

19

19

Les sets

Lorsque vous utilisez des accolades {} pour créer un objet en Python, cela peut représenter soit un dictionnaire soit un ensemble (set), selon la syntaxe que vous utilisez.

20

20

Les sets

Un set en Python est une collection qui est à la fois non ordonnée et non indexée. Les sets ne permettent pas les doublons, ce qui signifie qu'un set ne peut pas avoir deux éléments identiques.

```
# Création d'un set
mon_set = {1, 2, 3, 4, 5}
print(mon_set) # Affiche {1, 2, 3, 4, 5}
# Ajout d'un élément
mon_set.add(6)
print(mon_set) # Affiche {1, 2, 3, 4, 5, 6}
# Tentative d'ajouter un doublon
mon_set.add(3)
print(mon_set) # Affiche toujours {1, 2, 3, 4, 5, 6} car 3 est déjà dans le set
# Suppression d'un élément
mon_set.remove(4)
print(mon_set) # Affiche {1, 2, 3, 5, 6}
# Vérification de l'appartenance
print(3 in mon_set) # Affiche True
print(10 in mon_set) # Affiche False
```

liste=[5,2,6,5,8,9]

21

21

Fin Partie 2

22

22