Les_Bases_4

July 16, 2025

1 Les Listes

[1]:
```
# Une liste est définie avec les crochets. Elles peuvent contenir ou non des␣
 ↪valeurs.


liste_0 = []                          # ----->  Cette liste ne contient aucune␣
 ↪valeur

liste_1 = [1,"Python",True,44,5.8] # ----->  Cette contient des valeurs séparées␣
 ↪par des virgules
print(liste_0)
print(liste_1)
```

```
[]
[1, 'Python', True, 44, 5.8]
```

1.1 Ajouter les éléments d'une liste

[2]:
```
# La méthode append() :
liste_0.append("premier") # ----> permet de rajouter UNE valeur dans la liste
print(liste_0)
```

```
['premier']
```

[3]:
```
# La méthode extend() :
liste_0.extend([12,6,67,"Markus", "Dembélé", 6.8])
print(liste_0)

# liste_0.extend(12,6) ----> ECHEC car extend() ne prend qu'un argument
```

```
['premier', 12, 6, 67, 'Markus', 'Dembélé', 6.8]
```

1.2 Enlever les éléments d'une liste

[4]:
```
liste_1.remove(5.8)
liste_0.remove("premier")

print(liste_0)
```

```
print(liste_1)
```

```
[12, 6, 67, 'Markus', 'Dembélé', 6.8]
[1, 'Python', True, 44]
```

1.3 Récupérer des éléments d'une liste

1.3.1 Les indices

[5]:
```
# Les indices

A = liste_0[0]  # ----> A prend le premier élément de liste_0
print(A)
# >>>> Il affichera 12

B = liste_1[-1] # ----> B prend le dernier élément de liste_1
print(B)
# >>>> Il affichera 44
```

```
12
44
```

1.3.2 Les slices

[6]:
```
liste_2 = liste_0[0:-1] # ----> Le [:-1] est EXCLUSIF
print(liste_2)
```

```
[12, 6, 67, 'Markus', 'Dembélé']
```

[7]:
```
PSG = liste_2[-1:]
PSG.extend(["Barcola", "Doué", "Hakimi",
            "Vitinha", "Kvara", "WZE",
            "Ruiz", "Lee", "Nuno Mendes",
            "Mayulu","Donnarumma"])
print(PSG)
```

```
['Dembélé', 'Barcola', 'Doué', 'Hakimi', 'Vitinha', 'Kvara', 'WZE', 'Ruiz',
'Lee', 'Nuno Mendes', 'Mayulu', 'Donnarumma']
```

[8]:
```
team_1 = PSG[::2]  # ----> On prend les éléments de PSG à partir du premier␣
 ↪jusqu'au dernier élément avec un pas de 2
team_2 = PSG[1::2] # ----> On prend les éléments de PSG à partir du deuxième␣
 ↪jusqu'au dernier élément avec un pas de 2

print(team_1)
print(team_2)
```

```
['Dembélé', 'Doué', 'Vitinha', 'WZE', 'Lee', 'Mayulu']
['Barcola', 'Hakimi', 'Kvara', 'Ruiz', 'Nuno Mendes', 'Donnarumma']
```

[9]:
```
# Inverser les éléments d'une liste

PSG_Inv = PSG[::-1]  # ----> On prend les éléments de PSG à partir du premier␣
 ↪jusqu'au dernier élément avec un pas de -1
print(PSG_Inv)
```

```
['Donnarumma', 'Mayulu', 'Nuno Mendes', 'Lee', 'Ruiz', 'WZE', 'Kvara',
'Vitinha', 'Hakimi', 'Doué', 'Barcola', 'Dembélé']
```

1.4 Autres méthodes des listes

Savoir les index des éléments

[10]:
```
# La méthode index():

index_liste_0= liste_0.index("Markus") # ----> permet de connaître l'index de␣
 ↪l'élément de la liste
print(index_liste_0)
```

```
3
```

1.4.1 Savoir le nombre d'occurences dans une liste

[11]:
```
# La méthode count():

liste_0.append(67) # ----> On ajoute le nb 67 pour avoir 2 mêmes éléments
count_1 = liste_0.count(67) # ----> permet de connaître l'occurence d'un␣
 ↪élément d'une liste

print(count_1)
# >>>> Il affichera le nombre d'occurences dans la liste
```

```
2
```

1.4.2 Trier une liste

[12]:
```
liste_2.sort()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 liste_2.sort()

TypeError: '<' not supported between instances of 'str' and 'int'
```

[ ]:
```
# >>>> sort() fonctionne avec des listes contenant UNIQUEMENT des éléments de␣
 ↪même type
```

[13]:
```
# La méthode sort() ne renvoie rien.

liste_3 = [True, False]
print(liste_3.sort())
```

```
None
```

[14]:
```
liste_3.sort() # ----> Elle range les éléments de la liste automatiquement␣
 ↪selon l'ordre alphabétique en <str> ou
               #       selon le nombre avec les <int>
liste_3
```

[14]:
```
[False, True]
```

[15]:
```
# Rangons la liste PSG par ordre alphabétique :
PSG
```

[15]:
```
['Dembélé',
 'Barcola',
 'Doué',
 'Hakimi',
 'Vitinha',
 'Kvara',
 'WZE',
 'Ruiz',
 'Lee',
 'Nuno Mendes',
 'Mayulu',
 'Donnarumma']
```

[16]:
```
# Méthode 1 : la fonction sorted():

PSG_trie = sorted(PSG)  # ----> la fonction sorted() inverse l'ordre de la liste
print(PSG_trie)
```

```
['Barcola', 'Dembélé', 'Donnarumma', 'Doué', 'Hakimi', 'Kvara', 'Lee', 'Mayulu',
'Nuno Mendes', 'Ruiz', 'Vitinha', 'WZE']
```

[17]:
```
# Méthode 2 : la méthode .sort():

PSG.sort()
PSG # ----> Avec sort(), pas besoin de récupérer la liste dans une variable.
```

[17]:
```
['Barcola',
 'Dembélé',
 'Donnarumma',
 'Doué',
 'Hakimi',
```

```
 'Kvara',
 'Lee',
 'Mayulu',
 'Nuno Mendes',
 'Ruiz',
 'Vitinha',
 'WZE']
```

[18]:
```
# ATTENTION !!!!
liste_3 = liste_3.sort() # ----> Comme la méthode sort() renvoie None, la liste␣
 ↪sera écrasée
print(liste_3)
```

```
None
```

[19]:
```
# La bonne méthode !
liste_3 = [True, False]
liste_3 = sorted(liste_3)
liste_3
```

[19]:
```
[False, True]
```

1.4.3 Inverser une liste avec une méthode

[20]:
```
# La méthode reverse():
print(PSG_Inv)
PSG_Inv.reverse() # ----> Inverse l'ordre de la liste
print(PSG_Inv)
```

```
['Donnarumma', 'Mayulu', 'Nuno Mendes', 'Lee', 'Ruiz', 'WZE', 'Kvara',
'Vitinha', 'Hakimi', 'Doué', 'Barcola', 'Dembélé']
['Dembélé', 'Barcola', 'Doué', 'Hakimi', 'Vitinha', 'Kvara', 'WZE', 'Ruiz',
'Lee', 'Nuno Mendes', 'Mayulu', 'Donnarumma']
```

1.4.4 Autres façons de supprimer les éléments d'une liste

[21]:
```
# La méthode pop():

indesirable = PSG_Inv.pop(-5) # ----> permet de supprmier un élément grâce à␣
 ↪son index
print(indesirable)
```

```
Ruiz
```

[22]:
```
print(PSG_Inv)
```

```
['Dembélé', 'Barcola', 'Doué', 'Hakimi', 'Vitinha', 'Kvara', 'WZE', 'Lee', 'Nuno
Mendes', 'Mayulu', 'Donnarumma']
```

[23]:
```
# La méthode clear():

PSG_Inv.clear() # ----> Vider la liste PSG_Inv
print(PSG_Inv)  # ----> La liste sera vide
```

```
[]
```

1.4.5 Joindre les éléments d'une liste (uniquement les chaînes de caractères)

[24]:
```
# La méthode join():

PSG_trie.clear()

PSG_trie = ["La","Ligue","des","champions","le","seul","trophée","manquant",
            "n'empêchera","pas","de","dire","ceci:"]
PSG_Inv  = ["\nLe","Paris","Saint","Germain",␣
 ↪"est","le","plus","grand","club","de","France."]

phrase_1 = " ".join(PSG_trie)
phrase_2 = " ".join(PSG_Inv)

print(phrase_1,phrase_2)
```

```
La Ligue des champions le seul trophée manquant n'empêchera pas de dire ceci:
Le Paris Saint Germain est le plus grand club de France.
```

1.4.6 Transformer une chaîne de caractères en éléments d'une liste

[25]:
```
# La méthode split():

Famille = "Zoléni, Johéda, Joadanne, Joseph, Daphné"
Famille_liste = Famille.split(", ") #----> Séparation des éléments de la <str>␣
 ↪en liste en enlevant ", "
print(Famille)
print(f"Liste : {Famille_liste}")
```

```
Zoléni, Johéda, Joadanne, Joseph, Daphné
Liste : ['Zoléni', 'Johéda', 'Joadanne', 'Joseph', 'Daphné']
```

1.4.7 Les opérateurs d'appartenance

[26]:
```
# in / not in

"Zoléni" in Famille_liste # ----->  "Zoléni" appartient à Famille_liste (it's␣
 ↪True)
# >>> True
```

[26]:
```
True
```

[27]:
```
"Zoléni " in Famille_liste # ----> "Zoléni " appartient à Famille_liste (it's␣
 ↪False)
# Un simple espace peut faire la différence
# >>> False
```

[27]: False

[28]:
```
"Zoléni_" not in Famille_liste #----> "Zoléni_" n'appartient pas à␣
 ↪Famille_liste (it's True)
# >>> True
```

[28]: True

[29]:
```
if "Zoléni" in Famille_liste :
    print("Bienvenue Zoléni !")
```

```
Bienvenue Zoléni !
```

[30]:
```
# Cela fonctionne aussi avec une chaîne de caractères:

"py" in "python"
```

[30]: True

[31]:
```
"obs" in "obscidian"
```

[31]: True

2 Les listes imbriquées

[32]:
```
PSG_trie.clear()
PSG_trie.extend([["Barcola","Dembélé","Doué","Kvara"],
                 ['Lee','Mayulu','Ruiz','Vitinha',"Joao Neves",'WZE'],
                 ["Donnarumma", "Safonov"],
                 ["Hakimi",'Nuno Mendes']])

PSG_trie.sort() # ----> Rangée par ordre alphabétique par rapport au premier␣
 ↪élément de chaque liste imbriquée

PSG_trie # ----> Réorganisation de la liste PSG_trie en listes imbriquées par␣
 ↪position sur le terrain
# ----> 1ere liste : Les attaquants
# ----> 2ème liste : Les gardiens
# ----> 3ème liste : Les latéraux
# ----> 4ème liste : Les milieux de terrain
```

[32]:
```
[['Barcola', 'Dembélé', 'Doué', 'Kvara'],
 ['Donnarumma', 'Safonov'],
 ['Hakimi', 'Nuno Mendes'],
 ['Lee', 'Mayulu', 'Ruiz', 'Vitinha', 'Joao Neves', 'WZE']]
```

[33]:
```
# Je veux accéder à l'élément "Doué" :
PSG_trie[0][2] # ----> On indique la première liste avec [0] puis le troisième␣
 ↪élément de cette liste avec [2].
```

[33]:
```
'Doué'
```

2.1 Différences entre une méthode et une fonction

[34]:
```
# Une méthode est une fonction qui appartient à un objet

# On souhaite trier la liste Famille_liste selon l'ordre alphabétique :

sorted(Famille_liste)
# La fontion sorte() retourne la liste triée
# La fonction sorted() n'agit pas directement sur la liste.

print(Famille_liste)
# >>> C'est pour cela que l'ordre de la liste reste inchangée.
```

```
['Zoléni', 'Johéda', 'Joadanne', 'Joseph', 'Daphné']
```

[35]:
```
# Il faut donc écraser l'ancienne liste en affectant ce que nous retourne la␣
 ↪fonction sorted() :

Famille_liste = sorted(Famille_liste)  # ----> Réaffectation effectuée
print(Famille_liste)
```

```
['Daphné', 'Joadanne', 'Johéda', 'Joseph', 'Zoléni']
```

[36]:
```
# La méthode reverse() est propre à une liste.
# C'est pour cela qu'on peut l'utiliser à la liste Famille_liste.

Famille_liste.reverse()
print(Famille_liste)

# Même chose pour la méthode sort() :
Famille_liste.sort()
print(Famille_liste)
```

```
['Zoléni', 'Joseph', 'Johéda', 'Joadanne', 'Daphné']
['Daphné', 'Joadanne', 'Johéda', 'Joseph', 'Zoléni']
```

3 Projet: Arsenal VS PSG

[37]:
```
# Essayons de deviner le onze titulaire contre Arsenal en demi-finale de LDC␣
 ↪avec les listes imbriquées:

PSG_squad = []
PSG_squad.extend([["Donnarumma", "Arnau Tenas", "Safonov"],                    ␣
 ↪     # Les gardiens
                  ["Hakimi", "Kimpembe", "Marquinhos", "L.Hernandez", "Nuno␣
 ↪Mendes",
                   "Beraldo", "Pacho"],                                        ␣
 ↪     # Les défenseurs
                  ["Joao Neves", "Vitinha", "WZE", "Mayulu", "Lee Kang-In",␣
 ↪"Ruiz"], # Les milieux de terrain
                  ["Barcola", "Dembélé", "Kvara", "Mbaye", "Doué", "G.Ramos"]])␣
 ↪     # Les attaquants

# Sélection des joueurs du onze de Luis Enrique

PSG_gk = [PSG_squad[0][0]]
# >>>  = [Donnarumma]

PSG_defs = [PSG_squad[1][0], PSG_squad[1][-3], PSG_squad[1][-1],␣
 ↪PSG_squad[1][2]]
# >>>    =  ['Hakimi',"Nuno Mendes", "Pacho", "Marquinhos"]

PSG_mds = [PSG_squad[2][0], PSG_squad[2][1], PSG_squad[2][-1]]
# >>>   = ["Joao Neves", "Vitinha", "Ruiz"]

PSG_atk = [PSG_squad[-1][1], PSG_squad[-1][-2], PSG_squad[-1][2]]
# >>>   = ["Dembélé", "Doué", "Kvara"]

onze_titulaire = [PSG_gk,PSG_defs,PSG_mds,PSG_atk]
# >>>          = [[Donnarumma],['Hakimi',"Nuno Mendes", "Pacho", "Marquinhos"],
#                 ["Joao Neves", "Vitinha", "Ruiz"],
#                 ["Dembélé", "Doué", "Kvara"]]
```

[38]:
```
# Enregistrement du contenu dans un fichier txt

def afficher_texte(file_contenu):
    with open(f'{file_contenu}', mode= 'w', encoding='utf-8') as file_contenu:
        file_contenu.write(f"\t\tPrédictions du Onze titulaire contre Arsenal␣
 ↪en 1/2 finale de LDC\n\n"
                           f"Le squad du PSG pour le déplacement à Londres est␣
 ↪composé de ces joueurs: \n\n"
                           f"Gardiens: {"; ".join(PSG_squad[0])}.\n"
```

```
                        f"Défenseurs: {"; ".join(PSG_squad[1])}\n"
                        f"Milieux: {"; ".join(PSG_squad[2])}\n"
                        f"Attaquants: {"; ".join(PSG_squad[3])}\n")

        file_contenu.write(f"Je prédis le onze titulaire du Paris Saint Germain.
↪\n"
                           f"Les joueurs sélectionnés pour le Onze du départ:
↪\n\n"
                           f"Gardien : {onze_titulaire[0][0]}\n"
                           f"Défenseurs: {"; ".join(onze_titulaire[1])}\n"
                           f"Milieux: {"; ".join(onze_titulaire[2])}\n"
                           f"Attaquants: {"; ".join(onze_titulaire[3])}\n")
```

[39]:
```
# Enregistrement du fichier txt :

def SaveToFile(file):
    afficher_texte("Onze_titulaire.txt")
    with open(f"{file}",mode= 'a', encoding='utf-8') as file:
        file.write(f"Copyright : Zoléni KOKOLO ZASSI\n"
                   f"29 Avril 2025")
    print("Voici mon Onze titulaire contre Arsenal en Ligue des champions :␣
↪Allez sur {txt}".format(txt="Onze_titulaire.txt"))

SaveToFile("Onze_titulaire.txt")
```

```
Voici mon Onze titulaire contre Arsenal en Ligue des champions : Allez sur
Onze_titulaire.txt
```

4 Objet muable et objet immuable

Les objets muables sont des objets que l'on peut modifier comme les listes, les dictionnaires ou les sets. Les chaînes de caractères et les nombres sont immuables. On ne peut pas modifier une chaîne de caractères. C'est pour cela que lorqu'on utlise une méthode sur une liste, elle est directement modifiée. Alors que si on utlise une méthode sur une chaîne de carctères, elle ne retourne aucun résultat mais il faut l'affecter à une variable.

4.1 Exemple

[40]:
```
Famille_liste.append("Caroline")
print(Famille_liste)
```

```
['Daphné', 'Joadanne', 'Johéda', 'Joseph', 'Zoléni', 'Caroline']
```

[41]:
```
film = "kung fu panda".title()
print(film)
```

```
Kung Fu Panda
```

5 Quelques fonctions supplémentaires

[42]:
```
# La fonction len():
# ----> Compte le nombre d'éléments dans une liste

print(len(Famille_liste)) # ----> 6
print(len("Zoléni")) # ----> 6
```

```
6
6
```

[43]:
```
# La fonction max():
# ----> Renvoie le plus grand élément d'une liste
print(max([1,2,3]))
print(max(["a","b","c"]))
```

```
3
c
```

[44]:
```
# La fonction min():
# ----> Renvoie le plus petit élément d'une liste
print(min([1,2,3]))
print(min(["a","b","c"]))
```

```
1
a
```

[45]:
```
# la fonction sum():
# ----> Renvoie la somme des éléments d'une liste
print(sum([1,2,3])) # ----> 6
print(sum([1,2,3,4,5])) # ----> 15
```

```
6
15
```

[46]:
```
# La fonction range():
# ----> Renvoie une liste d'entiers de 0 à n-1
print(list(range(10))) # ----> [0,1,2,3,4,5,6,7,8,9]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

6 Projet 2 : Le vérificateur de mot de passe

[47]:
```
mdp = input("Entrez un mot de passe (min 8 carctères): ")
mdp_trop_court  = "votre mot de passe est trop court."
if len(mdp) == 0:
    print(mdp_trop_court.upper())
elif len(mdp) < 8:
    print(mdp_trop_court.capitalize())
```

```python
elif mdp.isdigit():
    print("Votre mot de passe ne contient que des nombres.")
else:
    print("Inscription terminée.")
```

```
Inscription terminée.
```

© Zoléni KOKOLO ZASSI

28 mars 2025, mis à jour le 15 juillet 2025