

TRAVAUX PRATIQUES
La programmation en Python
T.P. n°5 : Les collections : Tuples et listes

BTS SN-IR
1 ^{ère} année
Page 1 sur 4

Les tuples :

```

1  langages = ()           # 1 : Création d'un tuple vide
2  print(type(langages))  # 2 : Affichage du type de la variable langages
3  langages = tuple()     # 3 : Autre méthode de création d'un tuple vide
4  print(type(langages))  # 4 : Affichage du type de la variable langages
5  langages = ("Python", "C++", "Java", "PHP") # 5 : Affectation d'une collection à la variable langages
6  print(type(langages))  # 6 : Affichage du type de la variable langages
7  langages_str = "Python C++ Java PHP"       # 7 : Affectation d'une collection à la variable langages
8  print(type(langages_str))                  # 8 : Affichage du type de la variable langages
9  langages = tuple(langages_str.split(" "))  # 9 : Conversion de la variable langages_str en tuple
10 print(type(langages))                     # 10 : Affichage du type de la variable langages
11 print(langages)                           # 11 : Affichage du contenu de la variable langages
12 print(langages[0])                        # 12 : Afficher le premier element de la variable langages
13 print(langages[-1])                       # 13 : Afficher le dernier element de la variable langages
14 print(langages[:2])                       # 14 : Afficher du premier jusqu'au troisième element
15 print(langages[::2])                      # 15 : Afficher le premier et le deuxième element
16 print("#####")                          # 16 : Afficher des '#' pour séparer les sections
17 for element in langages:                  # 17 : Parcourir la variable langages element par élément
18     print(element)                        # 18 : Afficher les éléments un par un
19 print("#####")                          # 19 : Afficher des '#' pour séparer les sections
20 i = 0                                     # 20 : Création et initialisation de la variable i
21 while i < len(langages):                  # 21 : Tant que i est < à la longueur de la variable langages
22     print(langages[i])                    # 22 : Afficher l'élément qui a l'indice i (1er à l'indice 0)
23     i += 1                               # 23 : Incrémenter la variable d'itération i
24 print("#####")                          # 24 : Afficher des '#' pour séparer les sections
25 langages_str = " ".join(langages)         # 25 : Conversion du tuple en str
26 print(type(langages_str))                 # 26 : Affichage du type de la variable langages_str
27 L1, L2, L3, L4 = langages_str.split(" ")  # 27 : Affectation multiple
28 print(L1, L2, L3, L4)                    # 28 : Affichage des variables L1, L2, L3, L4
29 print(L1)                                # 29 : Affichage des variable L1
30 print(L2)                                # 30 : Affichage des variable L2
31 print(langages.index("Python"))            # 31 : Affichage de l'indice de l'élément "Python"
32 print(langages.count("Java"))              # 32 : Comptage de l'ocurence de l'élément "Java"
33 print(sorted(langages))                   # 33 : Trier le tuple par ordre alphabétique
34

```

Exercice 1 : On exige l'utilisation de la boucle for

Ecrire un programme en python qui :

Demande à l'utilisateur de saisir un nombre décimal positif ex : 18.58

Affiche ce nombre en remplaçant le point par une virgule ex : 18,58

TRAVAUX PRATIQUES
La programmation en Python
T.P. n°5 : Les collections : Tuples et listes

BTS SN-IR
1 ^{ère} année
Page 2 sur 4

Les listes :

```

1  langages = []                # 1 : Création d'une liste vide
2  print(type(langages))       # 2 : Affichage du type de la variable langages
3  langages = list()           # 3 : Autre méthode de création d'une liste vide
4  print(type(langages))       # 4 : Affichage du type de la variable langages
5  langages = ["Python", "C#", "Ruby", "Java"] # 5 : Affectation d'une collection à la variable langages
6  print(type(langages))       # 6 : Affichage du type de la variable langages
7  langages_str = "Python C# Ruby Java" # 7 : Affectation d'une chaîne à la variable langages_str
8  print(type(langages_str))    # 8 : Affichage du type de la variable langages_str
9  langages = list(langages_str.split(" ")) # 9 : Conversion de la variable langages_str en liste
10 print(type(langages))        # 10 : Affichage du type de la variable langages
11 print(langages)              # 11 : Affichage du contenu de la variable langages
12 print(langages[0])           # 12 : Afficher le premier élément de la variable langages
13 print(langages[-1])          # 13 : Afficher le dernier élément de la variable langages
14 print(langages[:2])          # 14 : Afficher du premier jusqu'au troisième élément
15 print(langages[:2])          # 15 : Afficher le premier et le deuxième élément
16 print("#####")             # 16 : Afficher des '#' pour séparer les sections
17 for element in langages:      # 17 : Parcourir la variable langages élément par élément
18     print(element)            # 18 : Afficher les éléments un par un
19 print("#####")             # 19 : Afficher des '#' pour séparer les sections
20 i = 0                        # 20 : Création et initialisation de la variable i
21 while i < len(langages):      # 21 : Tant que i est < à la longueur de la variable langages
22     print(langages[i])        # 22 : Afficher l'élément qui a l'indice i (1er à l'indice 0)
23     i += 1                    # 23 : Incrémenter la variable d'itération i
24 print("#####")             # 24 : Afficher des '#' pour séparer les sections
25 langages_str = " ".join(langages) # 25 : Conversion de la liste en str
26 print(type(langages_str))     # 26 : Affichage du type de la variable langages_str
27 L1, L2, L3, L4 = langages_str.split(" ") # 27 : Affectation multiple
28 print(L1, L2, L3, L4)         # 28 : Affichage des variables L1, L2, L3, L4
29 print(L1)                    # 29 : Affichage des variable L1
30 print(L2)                    # 30 : Affichage des variable L2
31 print(langages.index("Python")) # 31 : Affichage de l'indice de l'élément "Python"
32 print(langages.count("Ruby"))  # 32 : Comptage de l'occurrence de l'élément "Ruby"
33 print(sorted(langages))        # 33 : Trier la liste par ordre alphabétique
34 print(type(L1))               # 34 : Affichage du type de la variable L1

```

Exercice 2 : On exige l'utilisation de la boucle for

Ecrire un programme en python qui :

Calcule la somme des nombres de cette liste : 5 + 2 + 4 + 7 + 3

```
ma_liste = ["5",2,4,"7",3]
```

Opérations sur les listes :

```

1  maListe =list()                # 1 : Création d'une liste vide
2  maListe.append("python")       # 2 : Ajouter un élément à la liste
3  print(maListe)                 # 3 : Affichage du contenu de la variable maListe
4  langages=['Django','PySide']   # 4 : Création d'une liste qui contient deux éléments
5  maListe.extend(langages)       # 5 : Etendre la maListe avec les éléments de langages
6  print(maListe)                 # 6 : Affichage du contenu de la variable maListe
7  autre =('html','css','sql')    # 7 : Création d'un tuple
8  maListe.extend(autre)          # 8 : Ajout d'un tuple à une liste
9  print(maListe)                 # 9 : Affichage du contenu de la variable maListe
10 # Attention pas de valeur de retour pour les methodes sur les listes elles renvoient None
11 # mais modifie l'objet d'origine
12 # on ne stocke pas la modification sur une autre variable de type list
13 # c'est la liste d'origine qui est modifiée
14 maListe2 = maListe.append('Java') # Création d'une liste et ajout d'un élément
15 maListe2 = maListe.append('PHP') #Ajout d'un deuxième élément
16 print("\nmaListe: ",maListe)     # 9 : Affichage du contenu de la variable maListe
17 print("\nmaListe2: ",maListe2)   # ne contient rien c'est la liste d'origine qui est modifiée: maListe2
18 print("#####")                  # 16 : Afficher des '#' pour séparer les sections
19 frameworks_liste=['Django','PySide']
20 Langages_tuple = ('c#','Ruby')
21 autreFrameworks_tuple = ('QT', 'Samfony')
22 print("\nConcatenation des listes ",maListe , " + ",frameworks_liste," : ")
23 maListe3 = maListe + frameworks_liste # on peut concatener les listes ensemble
24 print(maListe3)
25 print("\nOn ne peut concatener liste avec tuple:")
26 # maListe4 = maListe3 + autreLangages # on ne peut pas concatener liste avec tuple
27 print("\nConcatenation des tuples",Langages_tuple," + ",autreFrameworks_tuple, " : ")
28 monTuple = Langages_tuple + autreFrameworks_tuple# On peut concatener les tuples ensemble
29 print(monTuple)
30 del maListe3[3] # Supprime le quatrième élément qui a l'indice '3'
31 maListe3.remove('css') # Attention C: est en majuscule!!
32 print("\nMa nouvelle liste après suppression: ",maListe3)

34  ▾ autre_liste = [
35      [1, 'a'],
36      [4, 'd'],
37      [7, 'g'],
38      [26, 'z'],
39  ] # J'ai étalé la liste sur plusieurs lignes
40  ▾ for nb, lettre in autre_liste:
41      print("La lettre {} est la {}e de l'alphabet.".format(lettre, nb))
42      print("")
43      ma_liste = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
44  ▾ for i, elt in enumerate(ma_liste):
45      print("À l'indice {} se trouve {}".format(i, elt))
46

```

TRAVAUX PRATIQUES
La programmation en Python
T.P. n°5 : Les collections : Tuples et listes

BTS SN-IR
1 ^{ère} année
Page 4 sur 4

Exercice 3 : On exige l'utilisation de la boucle for

Ecrire un programme en python qui vous permet de :

Constituer une liste semaine contenant les 7 jours de la semaine.

À partir de cette liste, comment récupérez-vous seulement les 5 premiers jours de la semaine d'une part, et ceux du week-end d'autre part ? Utiliser pour cela l'indilage.

Chercher un autre moyen pour arriver au même résultat (en utilisant un autre indilage).

Trouver deux manières pour accéder au dernier jour de la semaine.

Inverser les jours de la semaine en une commande.

Exercice 4 : Saisons

Créez 4 listes hiver, printemps, ete et automne contenant les mois correspondants à ces saisons. Créez ensuite une liste saisons contenant les listes hiver, printemps, ete et automne. Prévoyez ce que renvoient les instructions suivantes,

puis vérifiez-le dans l'interpréteur :

saisons[2]

saisons[1][0]

saisons[1:2]

saisons[:][1]. Comment expliquez-vous ce dernier résultat ?

Exercice 5 : Table de multiplication par 9

Afficher la table de multiplication par 9 en une seule commande avec les instructions range() et list().

Exercice 6 : Nombres pairs

Répondre à la question suivante en une seule commande. Combien y a-t-il de nombres pairs dans l'intervalle [2, 10000] inclus ?