Tuples

Source : Python-doctor (https://python.doctor/page-apprendre-tuples-tuple-python)

Attention : un tuple est une liste qui **ne** peut **plus** être modifiée : c'et un objet **non-mutable**.

1. Création et accès aux valeurs

Créer un tuple vide

In [ ]:

```
tuple_vide=()
print("tuple_vide est de type ",type(tuple_vide)," et contient ",tuple_vide)
```

Créer un tuple contenant des valeurs

Avec la syntaxe *rigoureuse* :

In [ ]:

```
tuple_divers=(35,"Rennes",True)
print("tuple_divers est de type ",type(tuple_divers)," et contient ",tuple_divers)
```

Sans les parenthèses :

In [ ]:

```
un_tuple=1,2,3,4
print("un_tuple est de type ",type(un_tuple)," et contient ",u
n_tuple)
```

Remarques :

- les parenthèses ne sont pas obligatoires mais elle rendent le code plus lisible
- **la virgule est obligatoire** même dans le cas d'un tuple à un seul élément.

In [ ]:

```
faux_tuple1=("non tuple")
faux_tuple2=(4.5)
vrai_tuple1=("tuple",)
vrai_tuple2=(5.4,)
# vérification des types et contenus
liste_tuples=[faux_tuple1,faux_tuple2,vrai_tuple1,vrai_tuple2]
for element in liste_tuples :
    print("cet élément est de type ",type(element)," et contie
nt ",element)
```

Conséquence : on NE peut PAS construire directement un tuple par compréhension.

Créer un tuple à partir d'une liste : mon_tuple = tuple(ma_liste)

In [ ]:

```
ma_liste = [(i+1)**2 for i in range(10)]
tuple_carres = tuple(ma_liste)
print("tuple_carres contient : ",tuple_carres)
```

Créer un tuple à partir d'une chaîne de caractères : mon_tuple=tuple(chaine)

In [ ]:

```
apple=tuple("apple")
print(apple)
```

Concaténer deux tuples : tuple1 + tuple2

In [ ]:

```
concat_tuple = un_tuple + tuple_divers
print("le tuple concaténé contient : ",concat_tuple)
```

Récupérer une valeur

Pour récupérer une valeur d'indice connu : mon_tuple[indice]

In [ ]:

```
print(tuple_divers[1]," est dans le département ",tuple_divers
[0])
```

Rappel : on ne peut pas modifier une valeur contenue dans un tuple.

In [ ]:

```
tuple_divers[0]=3500
```

Remarque : pour contourner le caractère non-mutable du tuple, on peut passer transitoirement par une liste :

In [ ]:

```
liste = list(tuple_divers)
liste[0]=35000
tuple_divers = tuple(liste)
print(tuple_divers[1]," a pour code postal ",tuple_divers[0])
```

Parcourir les valeurs

In [ ]:

```
for valeur in tuple_divers :
    print(valeur)
```

Tester si une valeur est dans un tuple

In [ ]:

```
if "Rennes" in tuple_divers :
    print("Rennes est dans le tuple")
print("45 est dans le tuple : ",45 in tuple_divers)
```

Déterminer le nombre de valeurs contenues dans un tuple : len(nom_du_tuple)

In [ ]:

```
print("tuple_divers contient ",len(tuple_divers)," valeurs.")
```

Remarque : on **NE** peut **PAS** supprimer des valeurs d'un tuple. Par contre, on peut le "détruire" avec la fonction *del()*.
Attention : le tuple n'existe plus ensuite !

In [ ]:

```
print("tuple_vide contient : ",tuple_vide)
del(tuple_vide)
print("tuple_vide contient : ",tuple_vide)
```

Compter le nombre d'occurences d'une valeur dans un tuple : nom_tuple.count(valeur)

Récupérer l'indice d'une valeur donnée dans un tuple : nom_tuple.index(valeur)

In [ ]:

```
print("Il y a ",apple.count('p')," p dans apple.")
print("Dans apple, le l est en indice ",apple.index('l'))
```

2. Utilisations

- Comme le tuple est un objet non-mutable, il peut être utilisé pour créer des **constantes**.

- Il permet des **affectations multiples** :

In [ ]:

```
a,b=1,2
print("a = ",a,"et b = ",b)
a,b=b,a
print("maintenant, a = ",a,"et b = ",b)
```

- Par conséquent, il permet à une fonction de **retourner plusieurs valeurs** :

In [ ]:

```
def division_euclidienne(dividende,diviseur) :
    quotient = dividende // diviseur
    reste = dividende % diviseur
    return quotient,reste

q,r = division_euclidienne(10,3)
print("Dans la division euclidienne de 10 par 3, le quotient e
st ",q," et le reste ",r)
```