TD_tuple

April 1, 2021

1 Exercice 1 : niveau facile

```
t = (3, 7, 2, 2, 10)
```

1. Sans l'aide de l'ordinateur, donner le résultat des expressions ci-dessous.
   (Remarque : Certaines expressions renvoient une erreur)
2. Vérifier vos réponses à l'aide de l'ordinateur

[ ]:
```
t[3]
del(t[1])
t(2)
(1,) + t[4]
3,4 + 5,5
t[-3]
t[1:3]
```

2 Exercice 2 : niveau facile

Ecrire une fonction `quatreOperations` qui :

- prend en **paramètre** 2 nombres
- **renvoie** un tuple contenant les résultats des 4 opérations sur ces 2 nombres

Un exemple d'appel de cette fonction est donné ci-dessous

[5]:
```
monTuple = quatreOperations(8,2)

print(monTuple)

# On remarque bien que monTuple contient 4 éléments la 8+2, 8-2, 8*2 et 8/2
```

[5]:
```
(10, 6, 16, 4.0)
```

3 Exercice 3 : niveau intermédiaire

(Dans cet exercice, on utilisera l'opérateur de concaténation +)

1. Ecrire une fonction `prefixe` qui

   - prend en **paramètre** un tuple `t` et un élément `e`
   - **renvoie** un tuple dans lequel l'élément `e` a été placé en début de tuple `t`

Un exemple d'appel de cette fonction est donné ci-dessous

[15]:
```
nouveau_tuple = prefixe(4,(3,1,1,5,9))
print(nouveau_tuple)
```

```
(4, 3, 1, 1, 5, 9)
```

[18]:
```
nouveau_tuple = suffixe(4,(3,1,1,5,9))
print(nouveau_tuple)
```

```
(3, 1, 1, 5, 9, 4)
```

2. Ecrire une fonction `suffixe` qui

   - prend en **paramètre** un tuple `t` et un élément `e`
   - **renvoie** un tuple dans lequel l'élément `e` a été placé en fin de tuple `t`

Un exemple d'appel de cette fonction est donné ci-dessous

3. Ecrire une fonction `insere` qui

   - prend en **paramètre** un tuple `t` et un élément `e` et un indice `i` tel que $0 \leq i < len(t)$
   - **renvoie** un tuple dans lequel l'élément `e` a été inséré à l'indice `i` dans le tuple `t`

Ecrire une fonction `insere` qui **renvoie** avec `e`, `i` et `t` passés en **paramètre**.

Un exemple d'appel de cette fonction est donné ci-dessous

[20]:
```
# Insertion de l'entier 4 à l'indice 2
nouveau_tuple = insere(4,2,(3,1,1,5,9))
print(nouveau_tuple)
```

```
(3, 1, 4, 1, 5, 9)
```

4 Exercice 4 : niveau intermédiaire

(Dans cet exercice, on utilisera l'opérateur de concaténation `+`)

Ecrire une fonction `supprime` qui :

- prend en **paramètre** un tuple `t` et un indice `i` tel que $0 \leq i < len(t)$
- **renvoie** un tuple contenant tous les éléments de ce tuple `t` sauf celui d'un indice `i`

Un exemple d'appel de cette fonction est donné ci-dessous

[22]:
```
# Suppression de l'élément d'indice 3
nouveau_tuple = supprime(3,(3,1,1,5,9))
print(nouveau_tuple)
```

```
(3, 1, 1, 9)
```

5 Exercice 5 : niveau intermédiaire

[ ]:
```python
def ajoute(t1,t2):
    t = ()
    for i in range(len(t1)):
        # A compléter


    return t
```

Compléter la fonction `ajoute` qui :

- prend en **paramètre** 2 tuples `t1` et `t2` de même longueur et contenant des entiers
- **renvoie** un tuple dont chaque élément est la somme des éléments de `t1` et `t2`

Un exemple d'appel de cette fonction est donné ci-dessous

[3]:
```python
nouveau_tuple = ajoute( (3,1,5), (6,0,2) )
print(nouveau_tuple)
```

```
(9, 1, 7)
```

6 Exercice 6 : niveau difficile

Ecrire une fonction `zipper` qui :

- prend en **paramètre** 2 listes `liste1` et `liste2` de même longueur
- **renvoie** une liste contenant des tuples. Chaque tuple contenant un élément de `liste1` avec l'élément de `liste2` de même indice

Un exemple d'appel de cette fonction est donné ci-dessous

[14]:
```python
a = [1,2,3,4]
b =["a","b","c","d"]

zipper(a,b)
```

[14]:
```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
```