


Paradigmes de programmation (Exercices)

S1 - Langages et programmation

Les exercices précédés du symbole  sont à faire sur machine, en sauvegardant le fichier si nécessaire.

Les exercices précédés du symbole  doivent être résolus par écrit.

Exercice 1

Voici différents algorithmes permettant l'affichage des 10 chiffres entiers dans l'ordre décroissant.

Préciser pour chacun des algorithmes le type de paradigme auquel il correspond.

Algorithme 1 :

```
def decompter(n:int)->None:
    if n>=0:
        print(n)
        decompter(n-1)
decompter(9)
```

Algorithme 2 :

```
for i in range(10):
    print(9-i)
```

Algorithme 3 :

```
class Nombres():

    def __init__(self,valeur):
        self.valeur = valeur

    def diminuer(self):
        self.valeur -= 1

    def __str__(self):
        return str(self.valeur)

n = Nombres(9)
while n.valeur >= 0 :
    print(n)
    n.diminuer()
```

Exercice 2

Voici deux versions d'une fonction `teste_ordre_liste` dont l'objectif est de savoir si une liste est ordonnée par ordre croissant. Indiquer quel paradigme est utilisé dans chacune des deux versions et expliquer votre réponse.

Version 1

```
# -*- coding: utf-8 -*-

def test_ordre(a,b) :
    if a < b :
        return True
    else :
        return False

def test_ordre_liste(liste) :
    if len(liste) < 2 :
        return True
    return test_ordre(liste[0],liste[1]) and test_ordre_liste(liste[1:])

test_ordre_liste([2,3,2])
```

Version 2

```
# -*- coding: utf-8 -*-

def test_ordre(a,b) :
    if a < b :
        return True
    else :
        return False

def test_ordre_liste(liste) :
    if len(liste) < 2 :
        return True
    else :
        if test_ordre(liste[0],liste[1]) == False :
            return False
        else :
            del liste[0]
            return test_ordre_liste(liste)

test_ordre_liste([2,3,2])
```

Exercice 3

Le programme ci-dessous ne respecte pas le paradigme fonctionnel. Pourquoi ?

```
i = 5

def fct():
    if i > 5:
        return True
    else :
        return False

fct()
```

Modifier le programme pour qu'il respecte le paradigme fonctionnel.

Exercise 4

Même exercice avec le programme ci-dessous :

```
l = [4,7,3]

def ajout(i):
    l.append(i)
```

Complément pour les curieux

Pour ceux qui voudraient découvrir un langage fonctionnel, [cette page](#) fournit une introduction pas à pas aux bases de OCaml. Ce langage est utilisé en CPGE scientifiques dans le cadre de l'option informatique. Une autre introduction, pour la prépa, est disponible [ici](#). On peut programmer en OCaml en ligne [ici](#).