Contrôle 1 – éléments de correction

devoirs

Exercice 1

1. **Spécifications**

   Entrées **T** un tableau, **e** un élement du même type que les éléments de T, **k** un entier

   Rôle Déterminer s'il y a au moins **k** occurences de **e** dans **T**

   Sortie booléen

   Point bonus

   Préconditions k est strictement positif

   Postconditions VRAI si **e** apparaît au moins **k** fois dans **T**
   FAUX si **e** apparaît moins que **k** fois dans **T**

2.

Code Python

```
# en supposant que le tableau T soit un tableau d'entiers
assert mystere([], 0, 0)
assert mystere([1, 1, 1, 1], 1, 2)
assert not mystere([1, 1, 1, 1], 1, 5)
```

Exercice 2

1.

Code Python

```
def produit(tab):
    """Effectue le produit des éléments du tableau d'entiers tab.

    Entrée :
        tab de type list(int) : une liste d'entiers

    Sortie :
        type int : un entier

    >>> produit([1, 2, 3, 4, 5])
    120
    >>> produit([0, -2, 7, 9])
    0
    """
    p = 1

    for i in range(len(tab)):
        p = p * tab[i]

    return p
```

2. En ayant au préalable importer le module `doctest`, il s'agit d'utiliser l'instruction `doctest.testmod()`

3.

Code Python

```python
def produit(tab):
    p = 1

    for i in range(len(tab)):
        elt = tab[i]
        if elt == 0:
            return 0
        p = p * tab[i]

    return p
```

Point bonus

Code Python

```python
def produit(tab):
    n = len(tab)
    p = 1
    i = 0

    while i < n and tab[i] != 0:
        p = p * tab[i]
        i = i + 1

    return p if i == n else 0
```

Exercice 3

1. le code 4 est valide (algorithme d'Euclide sur 12 et 8)
2. **Code 1** il manque (deux-points) dans la définition de boucle ;
   Code 2 il manque une opération entre i et j (`str` ne prend qu'un seul argument) ;
   Code 3 il manque des *underscores* autour du nom du constructeur.

Exercice 4

Class complète avec le **point bonus** :

Code Python

```python
class Selecteur:

    def __init__(self, liste):
        self.liste = liste
        self.selection = 0

    def _tourner(self, n):
        if n > 0:
            self.selection += 1
            self.selection %= len(self.liste)
        elif n < 0:
            self.selection -= 1
            self.selection %= len(self.liste)

    def direct(self):
        self._tourner(1)

    def indirect(self):
        self._tourner(-1)

    def __str__(self):
        return str(self.liste[self.selection])

    def __repr__(self):
        return self.__str__()


if __name__ == "__main__":
    mes_options = ["auto", "programmes", "vitesse",
                   "ouverture", "manuel", "effets"]
    roue = Selecteur(mes_options)
    for _ in range(3):
        roue.direct()
    roue.indirect()
    print(roue)
```

Exercice 5

1.

Code Python

```python
from PIL import Image


img = Image.open("monImage.jpg")
img.save("maCopie.jpg")
```

2.

Code Python

```python
for i in range(W):
    img.putpixel((i, 30), (255, 255, 255))
```

3.

Code Python

```python
def change_ligne(im, n, couleur):
    for i in range(W):
        pxl = (i, n)
        c = im.getpixel(pxl)
        if c[0] < couleur[0]:
            im.putpixel(pxl, couleur)
```

4.

Code Python

```python
from PIL import Image, ImageDraw


# création de l'image et du calque
img = Image.new("RGB", (800, 800))
ctx = ImageDraw.Draw(img)
# tracé de la croix
ctx.line([(0, 0), (799, 799)])
ctx.line([(799, 0), (0, 799)])
# sauvegarde
img.save("testCroix.jpg")
```