Devoir en classe 1

devoirs

NSI — Septembre 2023

Exercice 1 (3 points)

Voici un algorithme sous la forme d'une fonction (implémentée en Python) :

Code Python

```
def mystere(tab, element, k):
    n = len(tab)
    s = 0
    #
    i = 0
    while i < n and s < k:
        if tab[i] == element:
            s += 1
        i += 1
    #
    return s == k
```

1. Préciser des spécifications possibles (entrées, rôle, sortie)
   Point bonus : pré-conditions et post-conditions
2. Proposer en concordance avec votre réponse précédente, une série de tests unitaires.

Exercice 2 (5 points)

On veut déterminer le produit des éléments d'un tableau constitué par des entiers, on se propose de l'écrire sous la forme d'une fonction `produit`.

1. Écrire une telle fonction en langage Python en la dotant d'une docstring dans lequel vous mettrez des doctests.
2. Quelle instruction permet d'évaluer les doctests dans les docstrings ?
3. Proposer une modification du code pour prendre en compte que si on rencontre 0 dans le tableau, on ne continue pas le parcours du tableau et on renvoie 0.
   Point bonus : utilisation d'une boucle non bornée

Exercice 3 (3 points)

Voici quatre morceaux de programme en langage Python :

Code 1

Code Python

```
for _ in range(4)
    print(_, _ ** .5)
```

Code 2

Code Python

```
s = ""
for i in [1, 3, 5, 7]:
  for j in range(2, 10, 2):
    s += str(ij)
```

Code 3

Code Python

```
class Fraction:
    def init(self, num, den):
        self.numerateur = num
        self.denominateur = den
```

Code 4

Code Python

```
a, b = 12, 8
while b != 0:
    a, b = b, a % b
```

1. Parmi les codes, un seul ne comporte aucune erreur, lequel ? (aucune justification n'est attendue)
2. Pour chacun des trois autres, expliquer l'erreur (car oui... il n'y en a qu'une).

Exercice 4 (6 points)

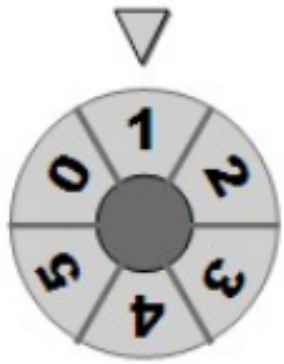
On désire modéliser, par une classe d'objets en langage Python, un sélecteur (une roue crantée) à n positions. Il suffit de tourner d'un cran dans un sens ou un autre pour changer de sélection.

On désire donc créer la classe **Selecteur** avec les contraintes suivantes :

sur un appareil photo

- le constructeur prend en paramètre un tableau des n sélections possibles ;
- la modélisation de la position se fera avec des entiers de 0 à $n-1$;
- elle disposera d'une méthode **tourner(entier)→None** qui permet de tourner une fois dans le sens direct (sens anti-horaire) si l'entier est positif et une fois dans l'autre sens si l'entier est négatif et ne change rien si l'entier est nul ;
  Point bonus : rendre cette méthode privée
- elle disposera de deux méthodes **direct()→None** et **indirect()→None** qui appellent la méthode **tourner** ;
- on permettra d'utiliser la fonction `print` sur l'objet grâce à la méthode spéciale **__str__** qui renvoie la sélection actuelle.

Note importante La sélecteur n'est jamais *en butée* : s'il est sur la «dernière position» et qu'on le fait tourner dans le sens direct, il se place alors dans la «première» position.



une modélisation visuelle

Voici un comportement attendue :

Code Python

```
1 mes_options = ["auto", "programmes", "vitesse", "ouverture",
  ↪  "manuel", "effets"]
2 roue = Selecteur(mes_options)
3 for i in range(3):
4     roue.direct()
5 roue.indirect()
6 print(roue) # cette commande affiche "vitesse"
```

Consigne : coder cette classe en langage Python.

Exercice 5 (5 points)

Cet exercice utilise la bibliothèque `PIL`. On rappelle ci-dessous quelques fonctions, attributs et méthodes :

Fonctions
- `Image.open(nom_de_fichier)` ouvre une image et renvoie un objet permettant d'y accéder.
- `Image.new(mode, taille)` crée un objet Image et renvoie l'objet.
- `ImageDraw.Draw(image)` crée un contexte graphique pour image et le renvoie.

Attribut `image.size` est un tuple contenant dans cet ordre le nombre de lignes et le nombre de colonnes de l'objet image.

Méthodes
- `image.getpixel(coordonnees)` renvoie la couleur aux coordonnées de l'image.
- `image.putpixel(coordonnees, couleur)` place la couleur aux coordonnées dans l'image, cela ne renvoie rien.
- `image.save(nom_de_fichier)` sauvegarde l'image dans un fichier, cela ne renvoie rien.
- `contexte_graphique.line(tab_coordonnees, couleur)` dessine une ligne brisée avec la couleur sur le contexte graphique en suivant le tableau de coordonnées, cela ne renvoie rien.
- `contexte_graphique.rectangle(tab_coordonnees, couleur1, couleur2)` dessine un rectangle en le remplissant avec la couleur1 et en l'entourant avec la couleur2.
  Le paramètre `tab_coordonnees` est du type `[(x0, y0), (x1, y1)]` avec `x1 >= x0` et `y1 >= y0`. Ce tableau définit une boîte englobante, les premières coordonnées définissent dans l'image un point en haut à gauche et les secondes un point en bas à droite.
  Cela ne renvoie rien.

Les questions peuvent être abordées de manière indépendante.

1. Écrire en Python un script qui importe le(s) module(s) nécessaire(s) et suffisant(s), ouvre une image contenue dans le fichier `monImage.jpg` et la sauvegarde dans un fichier nommé `maCopie.jpg`.

2. On suppose que l'on dispose d'un objet image, en mode `RGB`, nommée `img` de largeur `W`. Écrire en Python une boucle qui permet de mettre la ligne 30 de cette image en couleur blanche.

3. Créer une fonction `change_ligne(im, n, couleur)` qui prend en paramètre :

   - `im` un objet image de largeur `W` ;
   - `n` un numéro (valide) de ligne ;
   - `couleur` un tuple de couleur `RVB`.

   Cette fonction doit modifier la ligne `n` de l'image telle que :
   on remplace la couleur existante par `couleur` si la composante rouge de la couleur de `im` est plus petite que celle de `couleur`.

4. Écrire en Python un script qui crée une image en mode `RGB`, instancie un contexte graphique sur cette image, trace une grande croix (type multiplication) sur l'ensemble du contexte puis sauvegarde l'image dans un fichier nommé `testCroix.jpg`