Nom: Prénom: Groupe:

Université de Bordeaux — Devoir surveillé du 06 novembre 2024
Département Licence, **4TPU146U / 4TPU147U** Informatique 1 — 09h00 – 10h30

Le sujet comporte 05 exercices et 9 pages, dont une page d'annexe

Aucun document n'est autorisé – Des informations importantes comme les fonctions de manipulation d'images disponibles sont rappelées en annexe page 9. Vous pouvez détacher cette annexe pour plus de facilité.

Écrivez votre nom et prénom sur chaque feuille pour éviter qu'elles se perdent.

Exercice 1 Considérons la fonction `mystere` suivante :

```
1 def mystere(L):
2     n = len(L)
3     for i in range(n // 2):
4         if L[i] < L[n - 1 - i]:
5             return False
6     return True
```

1. Quel est le type du paramètre `L` de cette fonction et de la valeur qu'elle retourne ?

2. Simulez l'exécution de l'appel `mystere([12, 9, 6, 8, 7, 10])` en complétant le tableau suivant qui montre l'évolution des variables `n` et `i`, et les valeurs lues dans `L[i]` et `L[n-1-i]` :

`n`		
`i`		
`L[i]`		
`L[n-1-i]`		

3. Que retourne l'appel `mystere([8, 6, 3, 5])` ?

4. Que retourne l'appel `mystere([9, 4, 2, 4, 8])` ?

5. Quelle est la complexité de la fonction `mystere(L)`, en fonction de `n`, le nombre d'éléments contenu dans `L` ?
(ne cocher qu'une seule case)
☐ $\mathcal{O}(1)$ ☐ $\mathcal{O}(log(n))$ ☐ $\mathcal{O}(\sqrt{n})$ ☐ $\mathcal{O}(n)$ ☐ $\mathcal{O}(n^2)$ ☐ $\mathcal{O}(n^3)$ ☐ $\mathcal{O}(2^n)$

Exercice 2 Ecriture de code

Ecrivez une fonction `majoriteSuperieur(L:list, s:int)->bool` qui prend en entrée une liste d'entiers `L` et un entier `s` appelé seuil, et qui retourne `True` si `L` contient plus d'éléments supérieurs ou égaux à `s` que d'éléments inférieurs à `s`.

Par exemple si `L1=[5,12,7,8,10,16,3,7]` :

— l'appel à `majoriteSuperieur(L1,10)` retournera `False` car `L1` contient 3 élements supérieurs ou égaux à 10 et 5 éléments inférieurs à 10.

— l'appel à `majoriteSuperieur(L1,6)` retournera `True` car `L1` contient 6 élements supérieurs ou égaux à 6 et 2 éléments inférieurs à 6.

Exercice 3 Expressions booléennes

1. Soit le code suivant :

```
x = -1
y = 3
p = x >= 0 and y < 2
q = p or x != y
r = not (p and q)
```

Donnez les valeurs de `p`, `q`, et `r`, en précisant les résultats intermédiaires.

2. Indiquer l'ensemble des valeurs entières de `x` pour lesquelles l'expression Python suivante vaut `True`.

```
x >= 3 and x <= 9 and (x % 3 == 0 or x == 5)
```

3. Le tarif de visite d'un zoo est le suivant :
   - gratuit (0 euros) pour les enfants de moins de 3 ans ;
   - 14,50 euros pour les enfants entre 3 ans et 11 ans ;
   - 19,50 euros pour les adultes ou adolescents à partir de 12 ans ; 16,80 euros s'ils possèdent une carte jeune ;

   Écrivez la fonction `tarifZoo(age:int, carteJeune:bool)->float` qui retourne le tarif appliqué en fonction de l'âge du visiteur et d'un booléen qui vaut True si la personne possède une carte jeune et False sinon.

Exercice 4 Analyse des ventes

Vous avez une liste d'entiers représentant les valeurs des ventes sur plusieurs jours. Votre tâche consiste à créer plusieurs fonctions pour analyser ces valeurs.

1. Écrivez une fonction `nbreChangementsPositifs(L:list)->int` qui prend en argument une liste des valeurs des ventes sur plusieurs jours consécutifs et qui renvoie le nombre de jours où il y a eu une augmentation des ventes par rapport au jour précédent. Si c'est le premier jour, il n'y a pas de changement à comparer.
   - l'appel `nbreChangementsPositifs([100,80,120,120,140,150,100])` renvoie `3`. En effet en position `2` la liste indique une vente à 120 avec une augmentation de `40` par rapport au jour précédent, de même pour les positions 4 et 5 qui indiquent respectivement une valeur de `140` et `150` réalisant ainsi un bénéfice de `20` et de `10` par rapport au jour précédent.
   - l'appel `nbreChangementsPositifs([100, 80, 75, 70])` renvoie `0`.

2. Écrivez une fonction `joursEnHausse(L:list)->list` qui prend en argument une liste des valeurs des ventes sur plusieurs jours consécutifs et qui renvoie une nouvelle liste contenant les indices de position où il y a eu une augmentation des ventes par rapport au jour précédent. Si c'est le premier jour, il n'y a pas de changement à comparer.
   Attention : pour cette question vous devrez impérativement utiliser la fonction de la question précédente pour calculer la taille de la nouvelle liste. La création de cette liste se fera ensuite comme indiqué en annexe.
   - l'appel `joursEnHausse([100,80,120,120,140,150,100])` renvoie `[2,4,5]`.
   - l'appel `joursEnHausse([100, 80, 75, 70])` renvoie `[]` une liste vide.

3. Considérons la fonction `mystere` suivante :

```
1  def mystere(L):
2      hausses = joursEnHausse(L)
3      if len(hausses) > 0:
4          s = 0
5          for h in hausses:
6              s = s + (L[h] - L[h-1])
7          return s / len(hausses)
8      return 0
```

4. Que retourne l'appel `mystere([100,80,120,120,140,150,100])` ?

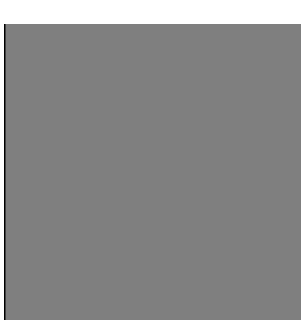
5. Quel est le rôle de la fonction `mystere` de manière générale ?

Exercice 5 Marches d'escalier

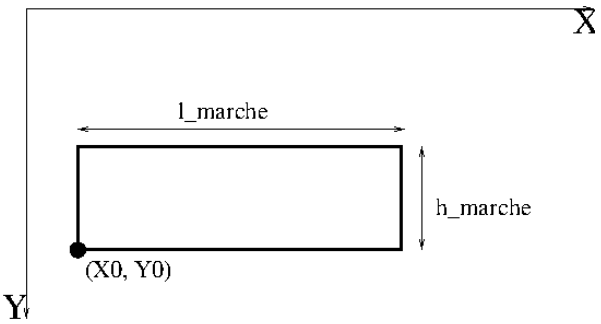
Dans cet exercice, vous allez utiliser la bibliothèque **bibimages.py** pour dessiner un escalier. Chaque marche de l'escalier sera représentée par un rectangle dont la largeur et la hauteur seront déterminées par des paramètres que vous définirez.
Les images qui accompagnent cet exercice se trouvent pages 6 et 7.

1. Écrire une fonction `colorierGris(img)` qui colorie l'image `img` avec du gris d'intensité RGB : (127, 127, 127).
   Ainsi par exemple, l'image `carreGris` illustrée sur la fig. 1(a) est une image carrée de côté `200` pixels. Elle est obtenue par le code suivant :
   ```
   carreGris= nouvelleImage(200, 200)
   colorierGris(carreGris)
   ```

2. Une marche est représentée par un rectangle défini par un point caractéristique de coordonnées `(x0, y0)` illustré sur la fig. 1(b), sa longueur `l_marche` , sa hauteur `h_marche` et sa couleur de remplissage `c`.
   Écrire une fonction `dessinerMarche(img, x0, y0, l_marche, h_marche, c)` pour dessiner une marche dans une image `img`.



(a) carreGris



(b) Marche

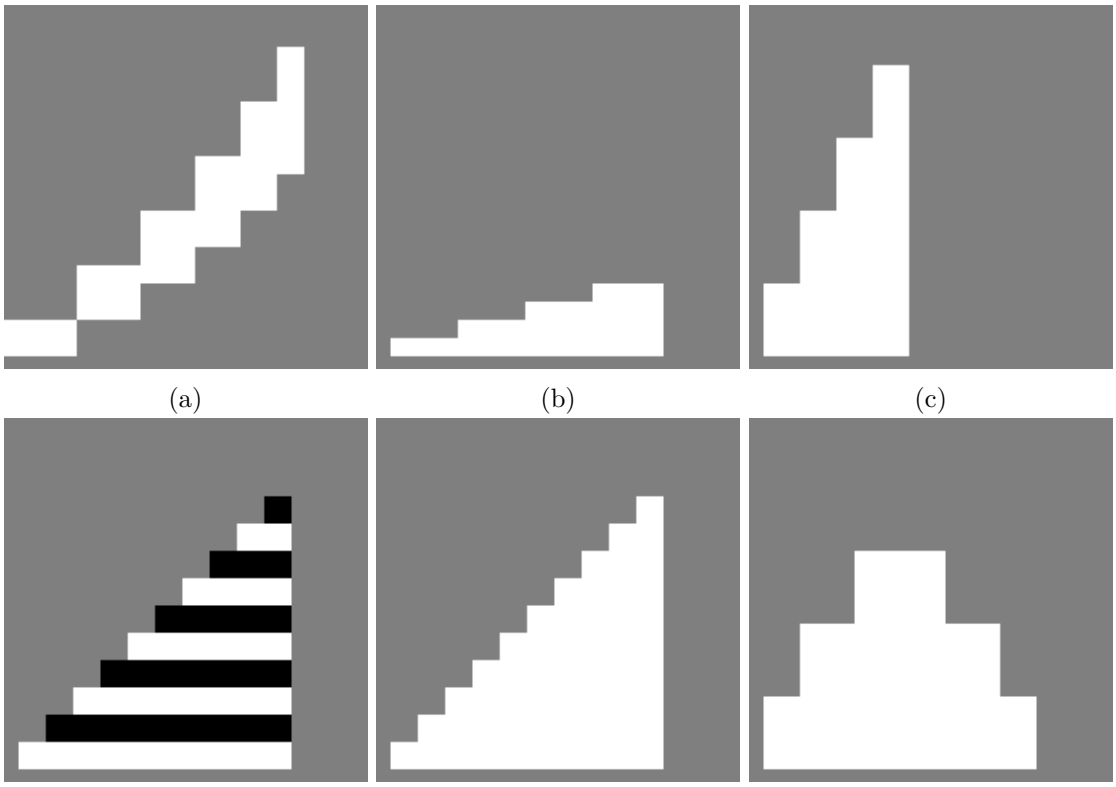
FIGURE 1

3. Un escalier est défini comme une suite de marches. La première marche est donnée par les coordonnées `(x0, y0)` de son point caractéristique, sa longueur `l_premiere_marche`, sa hauteur `h_marche` et sa couleur de remplissage `c`. La fonction `dessiner` dessine un escalier dans une image `img`.

```
1  def dessiner(img, x0, y0, l_premiere_marche, h_marche, nmb_marches, c):
2      x= x0
3      y= y0
4      decalage= l_premiere_marche//nmb_marches
5      l_marche= l_premiere_marche
6      for i in range(nmb_marches):
7          dessinerMarche(img, x, y, l_marche, h_marche, c)
8          x= x + decalage
9          y= y - h_marche
10         l_marche=  l_marche - decalage
```

Soit les images de la fig 2.
Pour chaque image choisie indiquer sur l'image ce que représentent les paramètres `x0, y0, l_premiere_marche, h_marche`.
Indiquer si elle peut être produite ou pas par un appel à `dessiner` à partir de l'image `carreGris`.
Justifier votre réponse.

Nom: Prénom: Groupe:



(a) (b) (c)

(d) (e) (f)

FIGURE 2 – Exemples d'images

Fin.

Fonctions mathématiques

Les fonctions mathématiques sont disponibles depuis le module `math`, par exemple la fonction `sqrt` calculant une racine carrée :

```
from math import sqrt
```

Certaines fonctions sont disponibles directement, par exemple l'opérateur puissance :

```
2**4
```

calcule ici 2^4.

Création d'une liste

Pour rappel, la syntaxe à utiliser pour créer une liste `L` de `n` éléments et contenant initialement que des 0 est :

```
L = [0] * n
```

Manipulation d'images

Voici un rappel des principales fonctions disponibles pour manipuler les images (à vous de voir celles qui sont utiles pour ce devoir).

Ci-dessous, `img` est une image	
`ouvrirImage(nom:str) -> image`	Ouvre le fichier *nom* et retourne l'image contenue dedans, par exemple : `img = ouvrirImage("teapot.png")`
`nouvelleImage(large:int, haut:int) -> image`	Retourne une image de taille $large \times haut$, initialement noire.
`afficherImage(img:image)`	Affiche l'image *img*.
`L = largeurImage(img)` `H = hauteurImage(img)`	Récupère la largeur de *img*. Récupère la hauteur de *img*.
`colorierPixel(img:image, x:int, y:int, (r,g,b))`	Peint le pixel (x, y) dans l'image *img* de la couleur (r, g, b)