Nom: Prénom: Groupe:

Université de Bordeaux — Devoir surveillé du 14 novembre 2022
Département Licence, **4TPU146U / 4TPU147U** Informatique 1 — 9h00 – 10h30

Le sujet comporte 5 exercices et 9 pages, dont une page d'annexe

Aucun document n'est autorisé – Toutes 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 :

```
def mystere(s: str) -> bool :
    n = len(s)
    for i in range(n//2):
        if s[i] != s[n-1-i] :
            return False
    return True
```

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

2. Simulez l'exécution de l'appel `mystere("ressasser")` en complétant le tableau suivant qui montre l'évolution des variables `n` et `i`, et les valeurs lues dans `s[i]` et `s[n-1-i]` :

`n`		
`i`		
`s[i]`		
`s[n-1-i]`		

3. Que retourne l'appel `mystere("ressasser")` ?

4. Que retourne l'appel `mystere("arbre")` ?

5. Quelle conjecture pouvez-vous émettre sur la valeur retournée par la fonction `mystere(s)`, pour une chaîne de caractères `s` ? On ne demande pas de prouver cette conjecture.

6. Quelle est la complexité de la fonction `mystere(s)`, en fonction de `n`, la longueur de son entrée `s` ?
(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 Expressions booléennes

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

```
1 <= x and x < 11 and (x % 3 == 0 or x == 7)
```

2. Écrire une expression Python dépendant d'une variable `x` qui vaut `True` si la valeur de `x` appartient à l'ensemble $\{1, 3, 5, 6\}$.

Exercice 3 Listes

Dans cet exercice, on considère des listes contenant des nombres entiers ou des nombres réels. Le type Python utilisé pour ces nombres est `float`.

1. Soit `L` une liste de nombres et `v` un nombre.
Écrire une fonction `valeursSuperieures(L: list, v: float) -> list` qui crée et retourne une nouvelle liste contenant les valeurs de la liste `L` qui sont supérieures ou égales à la valeur de `v`.
Exemples :

- l'appel `valeursSuperieures([12.5, 7, 16.5, 9, 12.5, 17.5], 12.5)` renvoie la liste `[12.5, 16.5, 12.5, 17.5]`

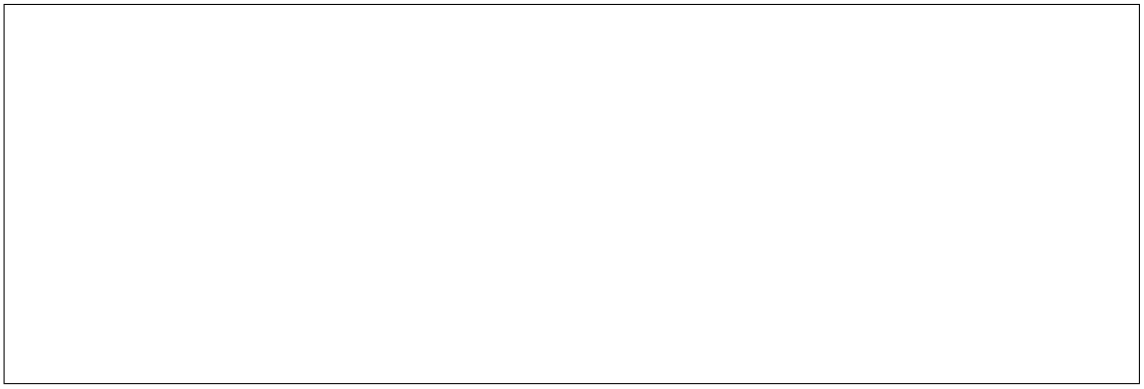
- l'appel `valeursSuperieures([12.5, 7, 16.5, 9, 12.5, 17.5], 18)` renvoie la liste vide `[]`

Indication :
une méthode pour écrire le code de la fonction `valeursSuperieures` consiste à d'abord compter le nombre de valeurs dans la liste `L` qui sont supérieures ou égales à `v`, puis à créer une liste de taille la valeur du compteur ainsi calculée et ensuite à remplir, le cas échéant, cette liste.

2. Soit la fonction `moyenneListe(L: list) -> float` qui retourne la moyenne des valeurs de la liste `L`.
Par exemple, l'appel `moyenneListe([12.5, 7, 16.5, 9, 12.5, 17.5])` retourne 12.5.

Écrire une fonction `valeursSuperieuresMoyenne(L: list) -> list` qui crée et retourne une nouvelle liste contenant les valeurs de la liste `L` qui sont supérieures ou égales à la moyenne de la liste `L`.
Par exemple, l'appel `valeursSuperieuresMoyenne([12.5, 7, 16.5, 9, 12.5, 17.5])` renvoie la liste `[12.5, 16.5, 12.5, 17.5]`

Pour cette question vous devrez impérativement utiliser les deux fonctions `moyenneListe` et `valeursSuperieures`. Il n'est pas demandé d'écrire la fonction `moyenneListe`.



Exercice 4 Extraire une sous-chaîne de caractères

Soit `texte` une chaîne de caractères et `c` un caractère de type `str`. On veut écrire une fonction `sousChaine(texte:str, c:str) -> str` capable d'extraire une sous chaîne de `texte` comprise entre la première et la dernière occurrence d'un caractère `c` donné.

1. Écrire une fonction `indicePremiereOccurrence(texte:str, c:str) -> int` qui prend en argument un texte, un caractère et qui renvoie l'indice (index) de la première occurrence de ce caractère. Cette fonction renvoie -1 si le caractère n'est pas trouvé dans le texte. Exemples :

   - l'appel `indicePremiereOccurrence("chemise", "e")` renvoie 2
   - l'appel `indicePremiereOccurrence("chemise", "z")` renvoie -1.

2. Écrire une fonction `indiceDerniereOccurrence(texte:str, c:str) -> int` qui prend en argument un texte, un caractère et qui renvoie l'indice (index) de la dernière occurrence de ce caractère. Cette fonction renvoie -1 si le caractère n'est pas trouvé dans le texte. Par exemple l'appel `indiceDerniereOccurrence("chemise", "e")` renvoie 6.

3. **Pour cette question vous devrez impérativement utiliser les deux fonctions précédentes.**
   Écrire une fonction `sousChaine(texte:str, c:str) -> str` qui prend en argument un texte, un caractère et qui renvoie la sous-chaîne comprise entre la première occurrence incluse et la dernière occurrence incluse du caractère passé en paramètre.
   Exemples
   - l'appel `sousChaine("chemise", "e")` renvoie `"emise"`.
   - l'appel `sousChaine("il fait beau", " ")` renvoie `" fait "`

   Cette fonction renvoie une chaîne vide si le caractère n'est pas trouvé dans le texte.

Exercice 5 Images

Dans cet exercice, on considère des images carrées, dont la largeur est donc égale à la hauteur.

1. Écrire une fonction `carreBlanc(img:image)` qui remplit une image carrée `img` en blanc. Ainsi par exemple, la Fig. 1(a) illustre l'image nommée `carre` produite par le code

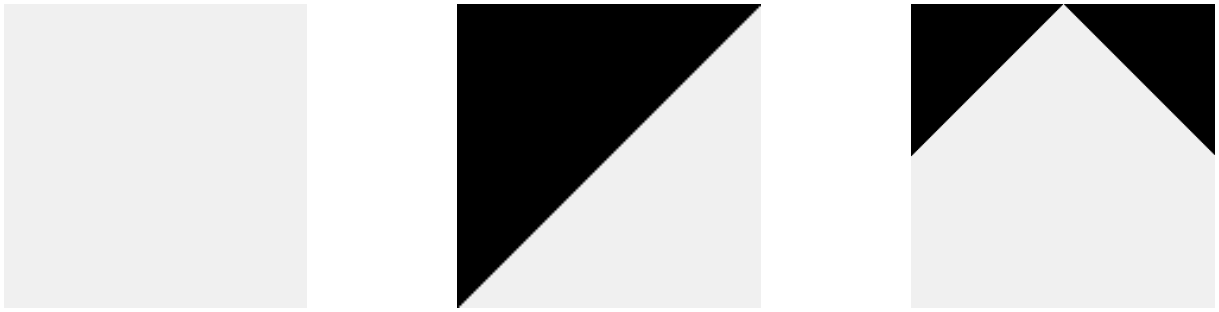
```
carre=nouvelleImage(128,128)
carreBlanc(carre)
afficherImage(carre)
```

2. Écrire une fonction `remplirTrSupGauche(img:image)` qui remplit le triangle supérieur gauche d'une image carrée `img` en noir, diagonale comprise, sans toucher au reste de l'image. Un exemple est donné sur la Fig. 1(b).

3. Écrire une fonction `remplirTrSupGaucheTrSupDroit(img: image)` qui remplit les deux triangles supérieurs, gauche et droit, d'une image carrée `img` en noir, sans toucher au reste de l'image. Une illustration est donnée sur la Fig. 1(c). Le côté des triangles est égal à la moitié de la largeur de `img`.

Nom: Prénom: Groupe:

Donner le code qui produit l'image carrée de côté `256` qui est donnée sur la Fig. 1(c).



(a) (b) (c)

FIGURE 1 – Création et transformation d'images : (a) Création d'une image carrée coloriée en blanc. (b) Remplissage en noir du triangle supérieur gauche. (c) Remplissage en noir des deux triangles supérieurs, droit et gauche.

FIN.

Création d'un tableau

Pour rappel, la syntaxe à utiliser pour créer un tableau `tab` de `n` éléments et contenant initialement que des 0 est :

```
tab = [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)