Exercices sur la mise au point de programmes

Exercice 1 On considère la fonction **commence_par_majuscule(chaine)** qui suit :

```
def commence_par_majuscule(chaine):
    initiale = chaine[0]
    if "A" <= initiale <= "Z":
        majuscule = True
    else:
        majuscule = False
    return majuscule
```

Que fait cette fonction ?

Quelle est la précondition sur la variable **chaine** ?

Quelle est la postcondition de cette fonction ?

Compléter le code de la fonction ci-dessous en écrivant sa documentation accompagnée de deux exemples / tests servant au module **doctest**, et en rajoutant la précondition et la postcondition sous la forme de deux assertions dans le code.

```
def commence_par_majuscule(chaine):
    """









    """

    initiale = chaine[0]
    if "A" <= initiale <= "Z":
        majuscule = True
    else:
        majuscule = False

    return majuscule

if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

Réécrire la fonction **commence_par_majuscule(chaine)** en le moins de lignes possible (et sans sa documentation ni les assertions).

```
def commence_par_majuscule(chaine):

```

Exercice 2 On considère la fonction **somme(tab)** suivante où tab est un tableau non vide de nombres :

```
def somme(tab):
    total = 0
    for i in range(len(tab)):
        total += tab[i]
    return total
```

Que fait cette fonction ?

Quelle est la précondition ?

Quelle est la postcondition de cette fonction ?

Quel est l'invariant de boucle ?

Compléter le code de la fonction ci-dessous en écrivant sa documentation accompagnée de deux exemples / tests servant au module **doctest**, et en rajoutant à la fois la précondition et la postcondition ainsi que l'invariant de boucle sous la forme de trois assertions dans le code. (*indication : utilisez sum() et le slicing*)

```
def somme(tab):
    """




    """

    total = 0
    for i in range(len(tab)):
        total += tab[i]


    return total

if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

Quel est le variant de boucle ?

Exercice 3 Compléter la fonction **`minimum(T)`** suivante en écrivant :

- La documentation de la fonction, sachant que **T** est un tableau de flottants.
- La précondition sur **T** et la postcondition sous forme d'assertions dans le code.

- L'invariant de boucle sous la forme d'une assertion dans le code.

```
def minimum(T):
    """   """
    # précondition

    xmin = T[0]
    for i in range(len(T)):
        if T[i] < xmin:
            xmin = T[i]
        # invariant de boucle


    # postcondition


    return xmin
```

Quel est le variant de boucle ?

Exercice 4 Trouvez le bug dans chaque programme, d'abord en le lisant, puis en l'exécutant.

```
x = 0.0
while x != 1.0:
    print(x, "au carré fait environ", x
* x)
    x += 0.1
```

```
def inverse(x):
    return 1 / x

for x in range(-4,5):
    print(x, "a pour inverse",
inverse(x))
```

```
compteur = 0

def incrementer_compteur():
    compteur = compteur + 1

incrementer_compteur()
incrementer_compteur()
incrementer_compteur()
print(compteur)
```

```
def renverser(T):
    i, j = 0, len(T)-1
    while i < j:
        T[i], T[j] = T[j], T[i]
        i, j = i+1, j-1

T = (62, 81, 53, 74, 40)
renverser(T)
print(T)
```

```
def cube(a):
    y = a * a * a

x = 5
y = cube(x)
print("Le cube de", x, "est", y)
```

```
def decaler(tableau):
    for i in range(1, len(tableau)):
        tableau[i] = tableau[i-1]
    return tableau

print(decaler([1, 2, 3, 4, 5]))
```

```
T = [19, 25, 30, 46, 78]
i = 0
while i <= len(T):
    print("T[{}] = {}".format(i, T[i]))
    i += 1
```

```
def convertir_pouce_en_cm(x):
    return 2.56 * x

mesure_pc = 85
mesure_cm =
convertir_pouce_en_cm(mesure_pc)
print(mesure_pc, "pouces", "fait
environ", \
      mesure_cm, "cm")
```

```python
def verifier_mot_de_passe(mdp):
    valide = False
    if mdp == "Deneb":
        valide = True
    return valide

while True:
    password = input("Saisir le mot de
passe : ")
    if verifier_mot_de_passe(password):
        break

print("Accès autorisé")
```

```python
def calculer_prix_reduit(tarif, age):
    prix = tarif
    if age < 10:
        reduction = 20
    elif age < 16:
        reduction = 10
    return prix - reduction

tarif = 200
age = int(input("Quel est votre âge ?
"))
prix = calculer_prix_reduit(tarif,
age)
print("Le prix à payer est de", prix,
"€")
```

```python
def somme_carres(N):
    somme = 0
    for n in range(1, N+1):
        carre = n * n
    somme += carre
    return somme

total = somme_carres(100)
print("La somme des carré des entiers",
\
      "de 1 à 100 est égale à", total)
```

```python
def reduction(tarif, age):
    if age == 18:
        tarif = tarif - 20 / 100 *
tarif
    return tarif

tarif = 25.0
age = input("Quel est votre age ?")
prix = reduction(tarif, age)
print("Vous devez payer", prix,
"euros")
```

```python
def est_present(x, T):
    for i in range(len(T)):
        if x == T[i]:
            return True
    return False

L = [6, 3, 9, 2, 7, 5, 0, 1, 8, 4]
if est_present(L, 5):
    print(5, "est présent dans la
liste")
else:
    print(5, "est absent de la liste")
```

```python
def calculer_tarif(age):
    tarif = 3.60
    if age >= 12:
        tarif = 4.20
    elif age >= 18:
        tarif = 5.80
    return tarif

print("0 à 12 ans",
calculer_tarif(10), "€")
print("12 à 18 ans",
calculer_tarif(16), "€")
print("18 ans et +",
calculer_tarif(20), "€")
```

```
def convertir_francs_en_euros(franc):
    return francs / 6.55957

somme_fr = 100
somme_eu =
convertir_francs_en_euros(somme_fr)
print(somme_fr, "F fait environ",
somme_eu, "€")
```

```
def est_present(x, liste):
    for element in liste:
        if x == element:
            return True
        else:
            return False

print(est_present(8, [5,10,15,20]))
```

```
def nom_complet(fiche):
    nom = "inconnu"
    if "prenom" in fiche:
        nom = fiche["nom"]
        prenom = fiche["prenom"]
    return prenom + " " + nom

def afficher_nom(fiche):
    nom = nom_complet(fiche)
    print(nom)

fiche1 = {"nom":"Fonce", "prenom":"Al"}
fiche2 = {"prenom":"Bil",
"ville":"Brest"}
afficher_nom(fiche1)
afficher_nom(fiche2)
```

```
def rechercher(x, T):
    debut, fin = 0, len(T)
    while fin - debut > 1:
        milieu = (debut + fin) // 2
        if T[milieu] <= x:
            debut = milieu
        else:
            fin = milieu
    if T[debut] == x:
        return debut
    return -1


L = [15, 42, 31, 36, 47, 29, 50]
i = rechercher(36, L)
print(36, "a pour indice", i)
```

Exercice 5 On souhaite réaliser une fonction qui renvoie la liste des mots d'une phrase. On considère la proposition de code source suivante.

```
def decouper(phrase):
    L = []
    debut, fin = 0, 0
    while fin < len(phrase):
        while phrase[fin] != " ":
            fin += 1
        mot = phrase[debut:fin]
        L.append(mot)
        debut, fin = fin + 1, fin + 1
    return L
```

Donnez la signification des variables :

- **L**
- **debut**
- **fin**
- **mot**

À quoi sert la boucle **while** la plus imbriquée ?

Appliquez à la main la fonction **decouper()** sur la chaîne **"Un bel été"** et notez les valeurs des variables juste après l'exécution de la ligne **L.append(mot)**.

n° d'itération	debut	fin	mot	L
1				
2				
3				

Quel bug constate-t-on et quelle en est l'origine ?

Quelle correction faut-il apporter à la fonction ?

```

```

Testez votre modification en exécutant le programme suivant.

```
proverbe = "Lorsqu'on s'occupe d'informatique il faut \
faire comme les canards... Paraître calme en surface \
et pédaler comme un forcené par en dessous."
mots = decouper(proverbe)
print(mots)
```

Exercice 6 Dans cet exercice vous devez réaliser un module sur les tableaux.

Pour cela, créez un fichier nommé **`tableau.py`** puis définissez-y les fonctions suivantes :

- **`minimum(T)`** qui renvoie le plus petit élément du tableau non vide **`T`**.
- **`maximum(T)`** qui renvoie le plus grand élément du tableau non vide **`T`**.
- **`trier(T)`** qui renvoie une copie du tableau **`T`**, triée dans l'ordre croissant en appliquant l'algorithme de tri par sélection.
- **`est_present(x, T)`** qui renvoie **`True`** ou **`False`** suivant que **`x`** est ou non dans le tableau **`T`**.
- **`indice(x, T)`** qui renvoie l'indice de l'élément **`x`** dans le tableau **`T`** s'il s'y trouve et qui renvoie **`-1`** lorsqu'il ne s'y trouve pas.
- **`indice_dichotomique(x, T)`** qui renvoie l'indice de l'élément **`x`** dans le tableau trié dans l'ordre croissant **`T`** en appliquant l'algorithme de recherche dichotomique, et qui renvoie **`-1`** si **`x`** est absent.
- **`moyenne(T)`** qui renvoie la moyenne du tableau de nombres **`T`**, non vide.
- **`mediane(T)`** qui renvoie la médiane du tableau de nombres **`T`**, non vide.

Vous accompagnerez chaque fonction de sa documentation. Vous placerez dans celle-ci des exemples ainsi que des tests unitaires **`doctest`**.

Vous compléterez enfin le code par les préconditions et postconditions, sous la forme d'assertions.