

NSI - Bac Blanc

- Correction -

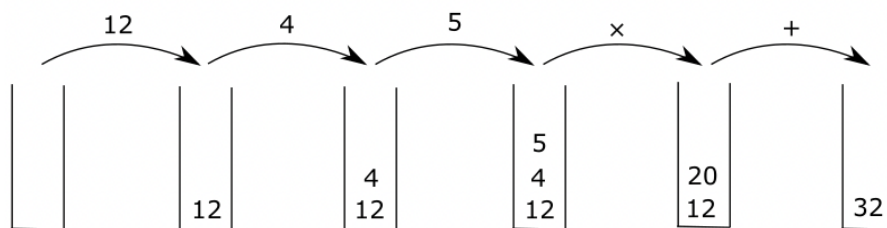
LGT Tani Malandi

2022 - 2023

Source des corrections : https://pixees.fr/informatiquelycee/term/suj_bac/index.html

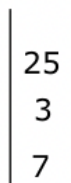
Exercice 1

1.



2.

- a. la variable *temp* contient la valeur 25
- b.



3.

```
def addition(p):  
    v1 = depiler(p)  
    v2 = depiler(p)  
    empiler(p, v1+v2)
```

4.

```
p = pile_vide()  
empiler(p, 3)  
empiler(p, 5)  
addition(p)  
empiler(p, 7)  
multiplication(p)
```

Exercice 2

1.
 - a. La fonction renvoie `[2, 6]`.
 - b. La fonction *mystere* renvoie les coordonnées du personnage après avoir parcouru le chemin passé en paramètre de la fonction (et en partant de l'origine du repère).
- 2.

```
def accessible(dep, arrivee):  
    arr = mystere(dep)  
    return arr[0]==arrivee[0] and arr[1]==arrivee[1]
```

- 3.

```
from random import randint  
def chemin(arrivee):  
    deplacement = '00000000'  
    while not accessible(deplacement, arrivee) :  
        deplacement=''  
        for k in range(8):  
            pas = str(randint(0,1))  
            deplacement = deplacement + pas  
    return deplacement
```

4. La plus grande valeur en binaire qui permet d'atteindre le point `[5, 3]` est `11100000` (il faut que les bits de poids fort soit à 1, on commence donc par monter avant de commencer à se déplacer vers la droite). En base 10 cela donne 224.

Exercice 3

1a

```
def total_hors_reduction(tab):  
    total = 0  
    for pa in tab:  
        total = total + pa  
    return total
```

on acceptera aussi un simple :

```
def total_hors_reduction(tab):  
    return sum(tab)
```

1b

```
def offre_bienvenue(tab):  
    somme = 0  
    longueur=len(tab)  
    if longueur > 0:  
        somme = tab[0]*0.8  
    if longueur > 1:  
        somme = somme + tab[1]*0.7  
    if longueur > 2:  
        for i in range(2,longueur):  
            somme=somme+tab[i]  
    return somme
```

2

```
def prix_solde(tab):  
    longueur = len(tab)  
    reduc = 1  
    if longueur == 1 :  
        reduc = 0.9  
    if longueur == 2 :  
        reduc = 0.8  
    if longueur == 3 :  
        reduc = 0.7  
    if longueur == 4 :  
        reduc = 0.6  
    if longueur >= 5 :  
        reduc = 0.5  
    return reduc*total_hors_reduction(tab)
```

autre possibilité plus "courte" :

```
def prix_soldeb(tab):  
    return total_hors_reduction(tab)*(1-0.1*min(5,len(tab)))
```

3a

Dans cette question nous partons du principe que tab n'est pas vide.

```
def minimum(tab):  
    mini = tab[0]  
    for pa in tab:  
        if pa < mini:  
            mini = pa  
    return mini
```

on acceptera aussi :

```
def minimum(tab):  
    return min(tab)
```

3b

```
def offre_bon_client(tab):  
    longueur = len(tab)  
    total = total_hors_reduction(tab)  
    if longueur >= 2:  
        total = total - minimum(tab)  
    return total
```

4a

plusieurs solutions possibles

[30.5, 20.0, 35.0, 15.0, 6.0, 5.0, 10.5] => total après promotion = 122 - 20 - 5 = 97

4b

[35, 30.5, 20.0, 15.0, 10.5, 6.0, 5.0] => total après promotion = 122 - 20 - 6 = 96

4c

Pour avoir le prix après promotion de déstockage le plus bas possible, il faut trier le tableau dans l'ordre décroissant. On peut donc utiliser un algorithme de tri (tri par sélection, tri par insertion ou tri fusion)

Exercice 4

1.

a.

```
class Carte:
    def __init__(self, val, coul):
        self.valeur = val
        self.couleur = coul
```

b.

```
c7 = Carte(7, "coeur")
```

2.

```
def initialiser() :
    jeu = []
    for c in ["coeur", "carreau", "trefle", "pique"] :
        for v in range(2,15) :
            carte_cree = Carte(v,c)
            jeu.append(carte_cree)
    return jeu
```

3.

La structure des données la plus adaptée est la file, puisque l'on a affaire à une structure de type FIFO (First IN First OUT). Le classement des cartes doit suivre la "règle FIFO", car la carte remportée (la dernière arrivée) doit être placée en dessous du tas.

4.

```
def comparer(carte1, carte2):
    if carte1.valeur > carte2.valeur :
        return 1
    elif carte1.valeur < carte2.valeur :
        return -1
    else :
        return 0
```

Exercice 5

1.

```
def somme(n) :  
    total = 0  
    for i in range(1,n) :  
        total = total + 1/i  
    return total
```

2.

a.

Au lieu d'avoir `while indice <= len(L)`, on devrait avoir `while indice < len(L)`. En effet, quand indice est égal à `len(L)`, nous allons avoir une erreur du type *list index out of range* (par exemple, pour un tableau de 5 éléments, l'indice du dernier élément est 4

b.

La fonction renvoie 0 alors qu'elle devrait renvoyer -2. Voici la version corrigée de la fonction :

```
def maxi(L) :  
    indice = 1  
    maximum = L[0]  
    while indice < len(L) :  
        if L[indice] > maximum :  
            maximum = L[indice]  
        indice = indice + 1  
    return maximum
```

3.

La variable `i` est de type nombre. Dans le *append* on essaye de concaténer une chaîne de caractère ("Joueur ") avec un nombre (`i`) ce qui va provoquer une erreur (on doit avoir 2 chaînes de caractères pour effectuer une concaténation). Il est donc nécessaire de transformer le nombre en chaîne de caractère grâce à la fonction `str`.

```
def genere(n):  
    L = []  
    for i in range(1, n+1) :  
        L.append('Joueur '+str(i))  
    return L
```

4.

a. 21

b. Dans le cas où on exécute *suite(7)*, au cours des différents appels récursifs, `n` prend les valeurs suivantes : 7, 5, 3, 2, 1, -1, -3...., nous n'aurons jamais le cas de base (`n=0`). Les appels récursifs vont avoir lieu jusqu'au moment où la pile de récursion sera pleine. Nous aurons donc une erreur : *RecursionError: maximum recursion depth exceeded*

5.

a.

(5, [10])

b.

4 [10]