STRUCTURES DE DONNEES – CORRECTION

EXERCICE 1: LA POO EN QUELQUES QUESTIONS

1) On définit la classe Identite **de la manière suivante:**

```
1 class Identite:
2     def __init__(self, nom, prenom, an):
3         self.nom=nom
4         self.prenom=prenom
5         self.an=an
6     def age(self, a):
7         return a-self.an
```

Qu´affichera la console à la suite de l´exécution des instructions suivantes? Justifiez toutes vos réponses.

a)

```
Maria Pires
```

En première ligne, on crée une instance (maria**) de la classe** Identite **en spécifiant les trois attributs de la classe. On affiche ensuite** 'Maria Pires' **par concaténation des deux premiers attributs qui sont des chaines de caractères.**

b) La console retourne une erreur (TypeError**) car on cherche à concaténer les attributs** nom **et** an **qui sont respectivement de types** chaine **et** entier**.**

c)

```
17
```

La méthode age **retourne la différence entre la valeur** a **passée en paramètre et l´attribut de l´instance de classe** an**. Elle permet donc de calculer l´âge à son anniversaire d'une année donnée.**

2) Ecrire le code de la classe Voiture **qui permet d´afficher** *'Ferrari Rouge'* **après avoir saisi les instructions suivantes:**

Nous avons besoin d´un constructeur qui définit la marque **et la** couleur **de la voiture et d´une méthode spéciale** __repr__ **de représentation de l´instance de classe:**

```
class Voiture:
    def __init__(self, nom, couleur):
        self.nom=nom
        self.couleur=couleur
    def __repr__(self):
        return self.nom  +' '+ self.couleur
```

3) Expliquez en détail ce que permet d´afficher ce programme:

`import random`
 	
`class Piece:`
`  def alea(self):`
`  return random.randint(0,1)`
`  def moyenne(self,n):`
`  tirage=[]`
`  for i in range (n):`
`  tirage.append(self.alea())`
`  return sum(tirage)/n`
`p=Piece()`
`print(p.moyenne(100))`
	La méthode alea de la classe Piece **retourne un entier entre 0 et 1 inclus. Elle effectue donc un tirage que l´on peut penser** équiprobable **entre ces deux valeurs.**
	
	La méthode moyenne retourne la moyenne des éléments de la liste tirage après n tirages. Cela revient à connaître la fréquence d´apparition **de la valeur 1.**
	
	On crée une instance de la classe Piece **(sans attribut d´instance d´où l´absence de constructeur) puis on affiche la fréquence d´apparition de la valeur 1 sur** 100 tirages**. Si on teste plusieurs fois le programme, on obtient une valeur toujours proche de 0.5.**
	
	Le programme suivant joue à pile ou face**.**

EXERCICE 2: LA CLASSE Date

On construit le code d'une variable globale et d'une classe Date **représenté ci-dessous:**

```
liste = ["janvier", "février", "mars", "avril", "mai", "juin", "juillet", "aout",
"septembre", "octobre", "novembre", "décembre"]
class Date:
    def __init__(self, jour, mois, annee):
            self.jour = jour
            self.mois = mois
            self.annee = annee
    def __lt__(self, d):
        if self.annee < d.annee:
            return True
        elif self.annee > d.annee:
            return False
        else:
            if self.mois < d.mois:
                return True
            elif self.mois > d.mois:
                return False
            else:
                return self.jour < d.jour
```

1) **Qu´affichera la console à l´exécution des instructions suivantes? Justifiez votre réponse.**

La console affichera False**. En effet, la méthode spéciale** __lt__ **retoune un booléen après évaluation de** d1<d2**. Elle détermine si l´instance** d1 **représente une date antérieure à celle représentée par l´instance** d2**.**

2) **Ecrire le code d'une méthode de la classe** Date **qui permet d´afficher** *'17décembre2021'* **après avoir saisi les instructions suivantes:**

Il faut définir une méthode spéciale **de représentation de l´instance de classe par concaténation. Il ne faut bien sûr pas oublier de convertir les entiers en chaine de caractères.**

```
def __repr__(self):
        return str(self.jour)+str(liste[self.mois-1])+str(self.annee)
```

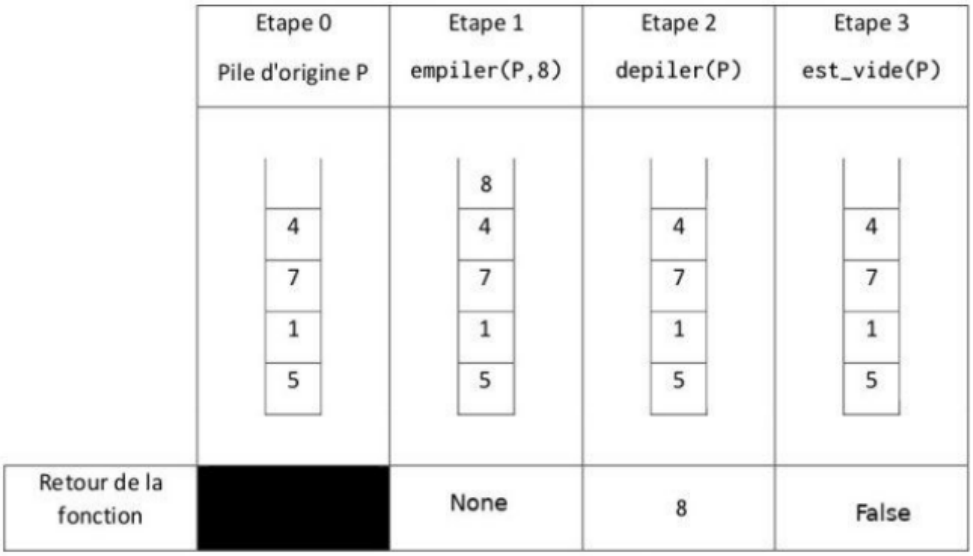
EXERCICE 3 : UNE HISTOIRE DE PILE

Dans cet exercice, on considère une pile d'entiers positifs. On suppose que les quatre fonctions suivantes ont été programmées préalablement en langage Python :

`empiler(P, e)` : ajoute l'élément e sur la pile P ;
`depiler(P)` : enlève le sommet de la pile P et retourne la valeur de ce sommet ;
`est_vide(P)` : retourne True si la pile est vide et False sinon ;
`creer_pile()` : retourne une pile vide.

Dans cet exercice, seule l'utilisation de ces quatre fonctions sur la structure de données pile est autorisée.

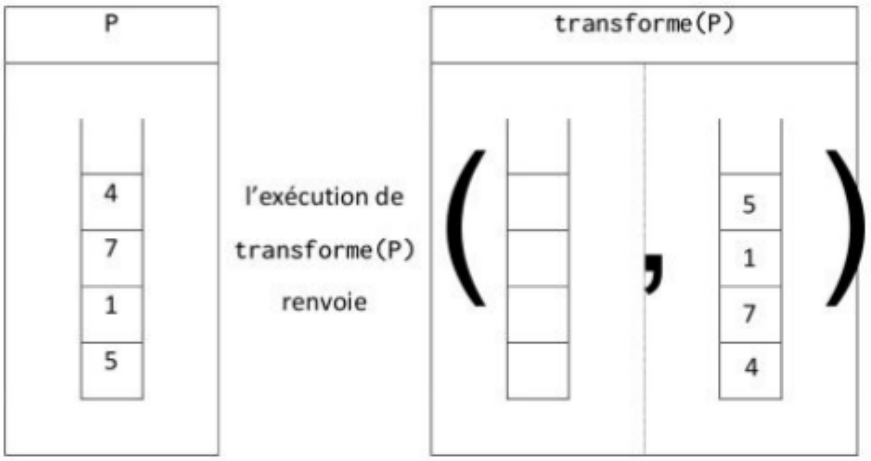
1) **Compléter le schéma ci-dessous.**



2) On propose la fonction ci-dessous, qui prend en argument une pile P et renvoie un couple de piles:

```
def transforme(P) :
    Q = creer_pile()
    while not est_vide(P) :
        v = depile(P)
        empile(Q,v)
    return (P,Q)
```

a. **Compléter sur le sujet le document ci-dessous** :



3) Écrire une fonction en langage Python maximum(P) recevant une pile P comme argument et qui renvoie la valeur maximale de cette pile. Après exécution de la fonction, la pile P devra être dans son état initial.

```
def maximum(P):
    assert not est_vide(P),'Pile vide!'
    Q=creer_pile()
    maxi=depiler(P)
    empiler(Q,maxi)
    while not est_vide(P):
        x=depiler(P)
        empiler(Q,x)
        if x>maxi:
            maxi=x
    while not est_vide(Q):
        x=depiler(Q)
        empiler(P,x)
    return maxi
```

4) On souhaite connaître le nombre d'éléments d'une pile à l'aide de la fonction taille(P), sans modifier la pile P.

a. Proposer une stratégie écrite en langage naturel et/ou expliquée à l'aide de schémas, qui permette de mettre en place une telle fonction.

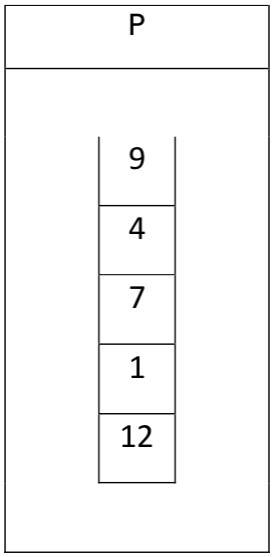
Stratégie :
La fonction taille(P) initialisera une variable compteur à 0 et une pile supplémentaire Q vide.

Puis, tant que la pile P n'est pas vide, on incrémente le compteur de 1, on dépile la pile P, et on empile l'élément dépilé sur la pile Q.

Une fois que cette boucle «while **» est finie, on en crée une deuxième pour dépiler Q et empiler P avec les éléments de Q dépilés. Cela permet que la pile P retrouve son état initial.**

b. Donner le code Python de cette fonction taille(P) (on pourra utiliser les cinq fonctions déjà programmées).

```
def taille(P):
    t=0
    Q=creer_pile()
    while not est_vide(P):
        x=depiler(P)
        empiler(Q,x)
        t+=1
    while not est_vide(Q):
        x=depiler(Q)
        empiler(P,x)
    return t
```



taille(P) retournera donc l'entier 5