STRUCTURES DE DONNEES – DEVOIR

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
```

a) Quels sont les attributs et méthodes de cette classe ? **Attributs** : `nom, prenom, an`
Méthodes : `age` et la méthode spéciale `__init__`

Qu´affichera la console à la suite de l´exécution des instructions suivantes ?

b)
```
maria=Identite('Pires','Maria',1999)
print(maria.nom + ' ' + maria.prenom)
```
```
>> Maria Pires
```

c)
```
print(maria.nom,maria.an)
```
```
>> Maria 1999
```

d)
```
hugo=Identite('Dupont','Hugo',2008)
print(hugo.age(2024))
```
```
>> 16
```

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

```
>>> F430=Voiture('Ferrari','rouge')
>>> print(F430)
Ferrari rouge
```

Réponse :

```
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 `moyenne` de la classe `Piece` calcule la moyenne des valeurs contenues dans la liste `tirage` qui ne contient que des 0 ou des 1 tirés aléatoirement grâce à la méthode `alea`.

4) On définit la classe Toto de la manière suivante :

```
class Toto:
    def __init__(self,a, b, c):
        self.a=a
        self.b=b
        self.c=c
    def myst(self):
        if self.a**2+self.b**2==self.c**2:
            return True
        else:
            return False
```

a) `t1.myst()` renvoie **True**

b) `t2.myst()` renvoie **False**

c) La méthode **`myst()`** vérifie si le triangle de côtés a, b et c est rectangle.

On saisit ensuite :

```
t1=Toto(3,4,5)
t2=Toto(4,5,6)
```

EXERCICE 2: LA CLASSE Date

On construit le code d'une variable globale `liste` 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.

```
d1 = Date(17, 12, 2021)
d2 = Date(11, 12, 2021)
print(d1<d2)
```

La console affichera `False` car la méthode spéciale `__lt__` permet de vérifier si une date est antérieure à une autre. Elle compare d´abord les années, puis les mois et enfin les jours si les deux autres sont égaux.

2) Ecrire le code d'une méthode de la classe Date qui permet d´afficher *'10octobre2024'* après avoir saisi les instructions suivantes :

```
>>> d3 = Date(10, 10, 2024)
>>> print(d3)
10octobre2024
```

On doit écrire la méthode spéciale `__repr__` :

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