I - PLANCHE 3 Analyse de programme

1 **1.** Créez une classe `Voiture` avec deux attributs d'instance:

- `couleur`, qui stocke la couleur de la voiture sous forme de chaîne de caractères
- `kilometrage`, qui stocke le nombre de kilomètres de la voiture sous forme d'entier.

2. Instanciez deux objets de type `Voiture` : une voiture bleue de 20 000 kilomètres et une voiture rouge de 30 000 kilomètres.

3. a. Écrire une méthode `affiche` de la classe `Voiture` : cette méthode affiche les attributs de la voiture `self` correspondant.

b. Comment utiliser cette méthode pour faire afficher :

Résultat

```
La voiture rouge a 20000 kilomètres
La voiture bleue a 30000 kilomètres
```

c. Comment faire pour que les instructions `print(v1)`, `print(v2)` affichent le résultat de la question précédente ?

2 On considère le programme suivant :

Code python

```python
class Chat:
    def bonjour(self):
        print("miaou")

class Chien:
    def bonjour(self):
        print("ouaf")

c = Chat()
c.bonjour()
c = Chien()
c.bonjour()
```

1. Qu'affiche le programme ?

2. Quelle est la particularité des méthodes de ce code ?

3 Vous répondrez aux questions suivantes en écrivant un code python dont l'exécution fournisse le résultat attendu. En particulier, toutes les méthodes dont on demande l'écriture devront être écrites *dans* le corps de la classe. Le code obtenu ne sera pas nécessairement écrit de bas en haut au fur et à mesure des questions, penser à laisser éventuellement de la place.

1. a. Écrire le code d'une classe `Heros` qui comporte trois attributs : `nom`, `cri_victoire`, `cri_defaite` (des chaines de caractères), `force` (un entier compris entre 0 et 100), et `pdv` (un entier valant 10 par défaut, représentant les points de vie du héro).

b. Instancier deux variables de type `Heros` :

- `hercule`, représentant le héros Hercule de force 99, criant "HAN" dans la victoire, mais "Ouch" il est blessé.
- `limace`, représentant le héros dit "la Limace Folle" de force 100, criant "GNIII" lorsqu'elle terrasse ses ennemis et "Blup" lorsqu'elle est blessée.

2. Écrire le code des méthodes `crier_victoire` et `crier_defaite` (de la classe `Heros`) qui affichent respectivement le `cri_victoire` et le `cri_defaite` de l'objet `self`.

Code python
```
def crier_victoire(self):
    """ Heros -> None"""
    pass
```

Code python
```
def crier_defaite(self):
    """ Heros -> None"""
    pass
```

3. Écrire une méthode `recoit_blessure` de la classe `Heros` qui prend en argument un entier `val` et qui diminue les points de vie du héros `self` de ce montant, puis affiche le cri de défaite du héros `self`.

Vous **n'utiliserez pas** l'attribut `cri_defaite` dans le corps de la méthode `recoit_blessure`.

Code python
```
def recoit_blessure(self, val):
    """ Hero, int -> None """
```

4. Écrire une méthode `attaquer` de la classe `Heros` qui prend en argument un objet `other` de type `Heros` et qui simule l'attaque de `other` par le héros `self` :

- si la force de `other` est strictement supérieure à celle de `self`, alors elle affiche `Rien ne se passe`.
- sinon, elle affiche le cri de victoire de l'objet `self` et inflige 1 point de blessure à l'objet `other`. La force du héros `self` est diminuée de moitiée (arrondi à l'entier supérieur) : il en faut de l'énergie pour porter un tel coup !

  Vous **n'utiliserez pas** l'attribut `pdv` dans le corps de la méthode `attaquer`.

  Code python
  ```
      def attaquer(self, other):
          """ Hero, Hero -> None """
  ```

5. Qu'affiche le code suivant ?

Code python
```
hercule.attaquer(limace)
limace.attaquer(hercule)
print(hercule.pdv, limace.pdv)
```

6. a. Écrire une fonction `jusqua_la_mort` qui étant donné deux héros `hero1` et `hero2` les font s'affronter en un combat singulier, jusqu'à ce que l'un d'entre eux soit terrassé (ses points de vie sont réduits à 0). À chaque round, chaque héros attaque l'autre, en commençant par `hero1`. La fonction `jusuqua_la_mort` renverra le héros vainqueur.

Code python
```
def jusqua_la_mort(hero1, hero2):
    """ Heros, Heros -> Heros """
    while hero1.pdv > 0 or hero2.pdv > 0:
        hero1.attaquer(hero2)
        hero2.attaquer(hero1)
    return hero1.nom if hero1.pdv > 0 else hero2.nom
```

b. Quelle instruction python permet d'afficher le nom du héros vainqueur dans le combat qui oppose Hercule à la Limace Folle ?

4 On se propose d'implémenter quelques fonctionnalités Yahtzee, dont le but est d'enchaîner les combinaisons à l'aide de cinq dés pour remporter un maximum de points.
Dans cette version extrêmement simplifiée, on propose de jouer selon le principe suivant :

- À chaque tour de jeu, on prend dans sa main cinq dés ;
- On répète au plus 3 fois les opérations suivantes :
  - On lance tous les dés présents dans notre main ;
  - On peut mettre de côté tout ou partie des dés.
    Ceux-ci ne seront pas relancés par la suite.
- À l'issue des 3 lancers, tous les dés restants sont écartés.
- On compte le score : celui-ci est égal à la somme des dés mis de côté.

Code python

```python
from random import randint
class De:
    def __init__(self):
        self.valeur = None

    def lancer(self):
        self.valeur = randint(1, 6)
        return self.valeur
```

1. Instancier une variable `de1`. Quelle instruction doit-on utiliser pour afficher le résultat d'un lancer dé dé ?

2. La classe `Jeu` simule le fonctionnement simplifié du jeu de Yahtzee décrit plus haut. Toutes les méthodes dont on demande l'écriture ci-dessous sont des méthodes de la classe `Jeu`.

Code python

```python
class Jeu:
    def __init__(self):
        self.main = []
        self.des_de_cote = []
```

a. Écrire une méthode `prendre_des` qui étant donné un entier `n` ajoute à la liste `self.main` un nombre `n` de dés.

b. Écrire une méthode `lancer` qui lance tous les dés présents dans la liste `self.main`.
Cette méthode affichera également pour chaque dé lancé son indice dans la liste `self.main` ainsi que le résultat du dé.

c. Écrire une méthode `ecarter` qui étant donné une liste `nums` d'indices compatibles avec la taille de la liste `self.main` supprime de cette liste les éléments dont les indices sont donnés dans `nums` pour les ajouter à la liste `self.des_de_cote`.

d. Écrire une méthode `compter_score` qui calcule la somme des dés présents dans la liste `self.des_de_cote`.

e. Écrire une méthode `jouer` qui simule le fonctionnement du jeu.

Remarque. L'instruction `nums = map(int, list(input("Sélectionner les dés : ")))` convertit l'entrée utilisateur en une liste de chiffres.

3. Souligner en rouge dans la description du jeu les mots correspondants aux noms des méthodes, en bleu les mots correspondant aux noms des attributs. Que constate-t-on ?

5 On considère la classe `Point` suivante :

Code python

```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def deplace(self, dx, dy):
        self.x = self.x + dx
        self.y = self.y + dy

```

```python
    def symetrique(self):
        return Point(-self.x, -self.y)

    def __str__(self):
        return f"Point({self.x}, {self.y})"
```

1. a. Quelle instruction python permet d'instancier une variable `a` de type `Point` représentant le point $A(2;4)$?

b. Quels sont les attributs des objets de type `Point` ?

c. Quelles sont les méthodes des objets de type `Point` ?

d. Donner la signature complète des méthodes `deplace` et `symetrique` de la classe `Point`.

e. Qu'affiche le code ci-dessous ? Justifier votre réponse.

Code python

```python
b = Point(1, 2) # représente B(1 ; 2)
print(b)
b.deplace(3, 5)
print(b)
b.symetrique()
print(b)
```

2. a. Écrire le code d'une classe `Segment`, dont les instances de classe possèdent un attribut `debut` et un attribut `fin`, tous deux de type `Point`.

b. Écrire l'instruction python qui permet d'instancier une variable `s1` représentant le segment $[AB]$.

3. a. Écrire une méthode `deplace` de la classe `Segment` qui prend en entrée deux nombres flottants `dx` et `dy` et qui translate le segment `self` selon le vecteur $\begin{pmatrix} dx \\ dy \end{pmatrix}$.

Code python

```python
def deplace(self, dx, dy):
    """ Segment, float, float -> None """
    pass
```

b. Écrire une méthode `symetrique` de la classe `Segment` qui renvoie le segment symétrique du segment `self` par rapport à l'origine du repère.

Code python

```python
def symetrique(self):
    """ Segment -> Segment """
    pass
```

4. La méthode `__str__` de la classe `Segment` est définie de la manière suivante :

Code python

```python
def __str__(self):
    return f"Segment {self.debut.x, self.debut.y} -- {self.fin.x,
     ↪  self.fin.y}"
```

Quel affichage réalise le code suivant ? Justifier votre réponse.

Code python

```python
print(s1)
s1.deplace(8, 2)
print(s1)
s1.symetrique()
print(s1)
print(a, b)
```