Polynésie – 2025 – sujet2 - Correction

Exercice 1 (6 points)

Partie A – Classe `Carte`

1. Écrire le code du constructeur de la classe Carte

```
class Carte:
 def __init__(self, valeur):
     self.valeur = valeur
     self.TdB = self.calcul_TdB()
```

👉*Commentaire :*

- *Le constructeur initialise l'attribut `valeur`.*
- *On calcule immédiatement le nombre de têtes de bœuf via `calcul_TdB()`.*
- *On utilise `self` pour accéder aux attributs de l'objet courant.*

---

2. Méthode `calcul_TdB`

```
def calcul_TdB(self):
    nb = 0

    if self.valeur % 11 == 0:
        nb += 5

    if self.valeur % 10 == 0:
        nb += 3

    if self.valeur % 10 == 5:
        nb += 2

    if nb == 0:
        nb = 1

    return nb
```

👉 *Commentaire :*

*- Les conditions se **cumulent**, donc on utilise plusieurs if (et pas elif).*

- `% 11 == 0` teste la divisibilité par 11.

- `% 10` permet de tester le chiffre des unités.

- Si aucune règle ne s'applique, on met 1 TdB par défaut.

Exemple : 55 → divisible par 11 (+5) et finit par 5 (+2) → 7 TdB.

3. Méthode `est_superieure_a`

```
def est_superieure_a(self, autre):
    return self.valeur > autre.valeur
```

👉 Commentaire :

- *Compare simplement les valeurs des deux objets `Carte`.*
- *Retourne un booléen.*

Partie B – Classe `Paquet`

4. Compléter le code (lignes 6, 7 et 10)

```
class Paquet:

    def __init__(self, L):
        self.contenu = L

    def afficher(self):
        for carte in self.contenu:
            print(carte.valeur)

    def ajouter_carte(self, carte):
        self.contenu.append(carte)
```

👉 *Commentaire :*

- *`contenu` est une liste d'objets `Carte`.*
- *`afficher()` parcourt la liste et affiche chaque valeur.*
- *`append()` ajoute une carte en fin de liste.*

---

5. Méthode `nombre_TdB`

```
def nombre_TdB(self):
    total = 0
    for carte in self.contenu:
        total += carte.TdB
    return total
```

👉 *Commentaire :*

- *On additionne les TdB de chaque carte.*
- *On renvoie le total.*

6. Méthode `distribuer`

```
def distribuer(self, nbr):
    paquets = [Paquet([]) for _ in range(nbr)]

    for i in range(10):
        for j in range(nbr):
            carte = self.contenu.pop(0)
            paquets[j].ajouter_carte(carte)

    return paquets
```

👉*Commentaire :*

- Création de nbr paquets vides.
- Chaque joueur reçoit 10 cartes.
- `pop(0)` enlève la première carte du paquet initial.
- Distribution circulaire.

Partie C – Classe `Joueur`

7. Instancier le joueur J1

```
J1 = Joueur("Joueur 1", L[0])
```

👉*Commentaire :*

- *`"Joueur 1"` est le nom.*
- *`L[0]` correspond à son paquet.*

---

8. Méthode `ramasser_paquet`

```
    def ramasser_paquet(self, paquet):
        for carte in paquet.contenu:
            self.cartes_ramassees.ajouter_carte(carte)

        self.penalite += paquet.nombre_TdB()
```

👉*Commentaire :*

- *On ajoute les cartes au paquet `cartes_ramassees`.*
- *On met à jour la pénalité avec le total des TdB.*

9. Compléter le script principal (lignes 3, 7 et 9)

```
from random import *

# créer les 104 cartes
jeu = [Carte(i) for i in range(1, 105)]

# mélanger
shuffle(jeu)

# instancier le paquet
jeu_initial = Paquet(jeu)

# distribuer
distri = jeu_initial.distribuer(2)

Ordi = Joueur("Ordi", distri[0])
J1 = Joueur("J1", distri[1])
```

👉*Commentaire :*

- `range(1,105)` *permet d'obtenir 1 à 104 inclus.*
- `shuffle()` *mélange la liste.*
- `Paquet(jeu)` *crée le paquet initial.*
- `distribuer(2)` *distribue 10 cartes à chacun des deux joueurs.*
