TD1 - Introduction à la POO

Imène Kerboua

M2 DS
2022/2023

1 Rappels

1.1 Classe

Une classe est le squelette d'un objet. Elle définit l'objet grâce à un constructeur et contient un ensemble d'attributs et de méthodes qui lui sont propres.
Déclaration:

```
class Car(): # class definition
    def __init__(self): # constructor
        pass

if __name__ == "__main__":
    car_1 = Car() # car_1 is a Car object instance
    car_2 = Car()
```

1.2 Attribut

Un attribut représente une caractéristique d'un objet. Il prend la forme d'une variable.
Déclaration:

```
class Car(): # class definition
    def __init__(self, color_value): # constructor
        self.color = color_value
        # self.color is an attributte of Car
        # It is initialized using the passed parameter color_value

if __name__ == "__main__":
    car_1 = Car(color_value="red") # car_1 is a Car object instance
    car_2 = Car(color_value="green")

    print(car_1.color) # print color attribute of car_1
    # out> "red"
    print(car_2.color)
    # out> "green"
```

1.3 Méthode

Une méthode représente le comportement d'un objet, ou bien l'action qu'un objet peut exécuter. Elle prend la forme d'une fonction.
Déclaration:

```
class Car():
```

```
    def __init__(self, color):
        self.color = color
        self.changed_color = 0

    def change_color(new_color):
        self.color = new_color
        self.changed_color += 1

    def check_up(self):
        if self.changed_color == 0:
            print("The car is like new !")
        else:
            print(f"This car changed color {self.changed_color} time(s) !")

if __name__ == "__main__":
    car_1 = Car(color="red")
    print(car_1.color) # print color attribute of car_1
    # out> "red"
    car_1.checkup()
    # out> "The car is like new !"
    car_1 = car_1.change_color(new_color="black")
    print(car_1.color)
    # out> "black"
    car_1.checkup()
    # out> "This car has changed color 1 time(s) !"
```

2 Pratique

Questions

Q1 - Exécuter le code suivant et observer les résultats, que fait chaque appel `print` ?

```
class MaClasse:
    def __init__(self, attribut_nom: str) -> None:
        self.attribut_nom = attribut_nom

if __name__ == "__main__":
    obj = MaClasse(attribut_nom="Imene")
    print(obj)
    print(obj.attribut_nom)
    print(obj.__class__)
    print(obj.__class__.__name__)
```

Q2 - Idem pour le code suivant :

1. Quelle est la différence avec le code précédent ?
2. Que représente la valeur "Aucun" dans le constructeur ?

```
class MaClasse:
    def __init__(self, attribut_nom: str = "Aucun") -> None:
        self.attribut_nom = attribut_nom

    def __str__(self) -> str:
        return f"{self.attribut_nom}"

if __name__ == "__main__":
```

```
    obj = MaClasse()
    print(obj)
    print(obj.attribut_nom)
    print(obj.__class__)
    print(obj.__class__.__name__)
```

Q3 - Exécutez le code suivant, que remarquez-vous ?

```
class Couleur:
    def __init__(self, nom) -> None:
        self.nom = nom

class Voiture:
    def __init__(self, couleur: Couleur) -> None:
        self.couleur = couleur

    def __del__(self):
        print('Object deleted')

if __name__ == "__main__":
    couleur = Couleur(nom='rouge')
    print(couleur)

    voiture= Voiture(couleur=couleur)
    print(voiture)
    print(voiture.couleur)

    del voiture
    print(couleur)
    print(voiture)
```

Exercice 1 : Réfléchir en POO

Ecrire le code suivant en utilisant la POO.

```
"""Code for calcultating area of different shapes without OOP"""

def get_square_area(side):
    return side ** 2

def get_rectangle_area(length, width):
    return length * width

def get_circle_area(radius):
    return 3.14 * radius ** 2

if __name__ == "__main__":
    square1_side = 5
    square1_area = get_square_area(side=square1_side)
    print(f"Square 1: side={square1_side}, area={square1_area}")

    square2_side = 4
    square2_area = get_square_area(side=square2_side)
    print(f"Square 2: side={square2_side}, area={square2_area}")

    rectangle_length = 5
    rectangle_width = 3
    rectangle_area = get_rectangle_area(length=rectangle_length, width=rectangle_width)
```

```python
print(f"Rectangle: length={rectangle_length}, width={rectangle_width}, are={rectangle_area}")

circle1_radius = 2
circle1_area = get_circle_area(radius=circle1_radius)
print(f"Circle 1: radius={circle1_radius}, area={circle1_area}")

circle2_radius = 3
circle2_area = get_circle_area(radius=circle2_radius)
print(f"Circle 2: radius={circle2_radius}, area={circle2_area}")
```

Exercice 2

- Écrire une classe `Livre` avec les attributs suivants :
  - titre : Le titre du livre,
  - auteur : L'auteur du livre,
  - prix : Le prix du livre,
  - annee : L'année du livre.

  La classe `Livre` doit pouvoir être construite avec les possibilités suivantes :
  - `Livre()`
  - `Livre(titre)`
  - `Livre(titre, auteur)`
  - `Livre(titre, auteur, prix)`
  - `Livre(titre, auteur, prix, annee)`
- Instancier deux livres de votre choix.
- Créer une fonction qui renvoie vrai si le titre de l'instance passée en paramètre correspond au titre demandé.

Exercice 3 : Le compte bancaire

Donner le code complet de la classe `CompteBancaire` en suivant les étapes suivantes:

- Créer une classe Python nommée `CompteBancaire` qui représente un compte bancaire, ayant pour attributs : `numeroCompte` (type numérique) , `nom` (nom du propriétaire du compte de type chaine), `solde`.
- Créer un constructeur ayant comme paramètres : `numeroCompte`, `nom`, `solde`.
- Créer une méthode `versement()` qui gère les versements.
- Créer une méthode `retrait()` qui gère les retraits.
- Créer une méthode `commission()` permettant de prélever une commission à un pourcentage de 5 % du solde.
- Créer une méthode afficher() permettant d'afficher les détails sur le compte.

Exercice 4 : Jeu de cartes

jeu_de_cartes.py : Créer un classe `JeuDeCartes` qui crée un jeu de 52 cartes.

- Une carte est composée d'une valeur et d'une forme (coeur, carreau, pique, trèfle).
- Les méthodes suivantes doivent être implémentées :
  - `nom_carte` : affiche le nom d'une carte de manière littérale, ex. Valet de Coeur.
  - `battre` : mélange le jeu de cartes (on utilisera la fonction `shuffle` du module `random`).
  - `afficher` : permet d'afficher le jeu de cartes.
  - `tirer` : permet d'extraire une carte du jeu.
- Vous ferez un jeu de test, qui exécutera les points suivants :
  - Créer un jeu de cartes.
  - Mélanger le jeu de cartes (battre).
  - Afficher le jeu de cartes mélangé.
  - Tirer une à une toutes les cartes et les afficher.

Exercice 5 :

Soit à développer une application pour la gestion d'un stock. Un article est caractérisé par son numéro de référence, son nom, son prix de vente et une quantité en stock. Le stock est représenté par une collection d'articles.
Travail à faire:

1. Créer la classe `Article` contenant les éléments suivants :
   - Les attributs/propriétés.
   - Un constructeur d'initialisation.
   - La méthode `__str__()`.
2. Créer la classe `Stock` qui contient une collection d'articles de votre choix.
3. Dans la fonction `main()`, permettre de choisir une option dans le menu suivant et de l'appliquer :
   - Rechercher un article par référence.
   - Ajouter un article au stock en vérifiant l'unicité de la référence.
   - Supprimer un article par référence.
   - Modifier un article par référence.
   - Rechercher un article par nom.
   - Rechercher un article par intervalle de prix de vente.
   - Afficher tous les articles.
   - Quitter.

Exercice 6 : Construire des objets - Devoir

- **Pile (stack)**[1]: Ecrire un programme qui implémente la structure d'une pile en utilisant les classes et les objets.implémenter les méthodes *push* (insérer un élément en haut de la pile), *pop* (extraire le premier élément de la pile) et d'autres méthodes qui permettent de voir les détails de la pile (affichage, taille, etc.). Créer également une méthode qui permet d'inverser la pile (*inverse()*).
- **File (queue)**[2]: Ecrire un programme qui implémente la structure d'une file en utilisant les classes et les objets.implémenter les méthodes *enfiler* (insérer un élément dans la file), *défiler* (extraire le premier élément de la file) et d'autres méthodes qui permettent de voir les détails de la file (affichage, taille, etc.).

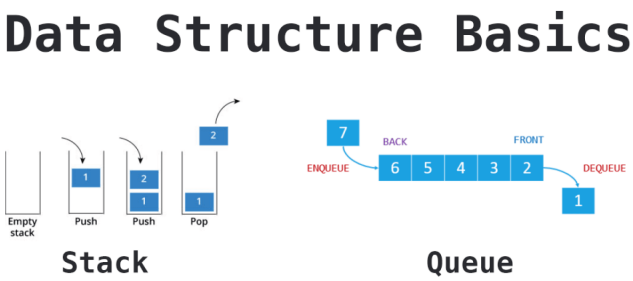

Figure 1: Pile (stack) et file (queue).

[1]Pile: Liste Last-In First-Out (LIFO), dernier arrivé premier sorti.
[2]File: Liste First-In First-Out (FIFO). Premier arrivé premier sorti.