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 une liste 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 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 Applications

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)
    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 : 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 du 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 3 : 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 4 : Construire des objets

- **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.).

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

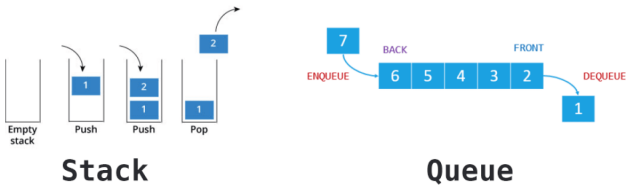

Figure 1: Pile (stack) et file (queue).

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. Dans la fonction `main()` créer :
   Le stock sous forme d'une collection d'articles de votre choix.
   Un menu présentant les fonctionnalités suivantes :
   - 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.

3 Exercices pour la partie Opérateurs, à ajouter dans un autre TD

Exercice 6 :

Soit un nombre complexe composé d'une partie imaginaire et d'une partie réelle.

- Représenter à l'aide d'un objet un nombre complexe.
- Permettre d'afficher le nombre complexe en utilisant la méthode `print()`.
  Affichage attendu : `print(a) -> "3i + 2"`

- Permettre de comparer deux objets complexes entre eux. Deux nombres complexes sont égaux si leurs parties imaginaires sont égales et 7leurs parties réelles sont égales.
- Écrire une fonction qui permet d'additionner deux nombres complexes. L'addition de deux nombres complexes se fait en additionnant les parties imaginaires entre elles et les parties réelles entre elles.

Exercice 7 :

Un étudiant étudie 2 matières: Informatique et Mathématiques. Il a comme caractéristiques : un nom, une date de naissance, une note pour chaque matière qu'il étudie et une moyenne générale. Il est demandé dans cet exercice de:

- Construire une liste de 5 objets "Etudiant" avec différentes valeurs.
- Afficher la liste des étudiants.
- Afficher la classement des étudiants selon leur moyenne générale (ordre décroissant).
- Créer deux étudiants avec la même moyenne générale. Afficher seulement les étudiants ayant la même moyenne générale.