Python Projet

Nassib Abdallah

2020-2021

Votre travail consiste à développer un fichier "main.py" contenant le programme principal et un fichier **".py"** pour chacune des parties composant le projet.

1 Première Partie : Commencer avec la POO

Votre travail dans cette partie consiste à refaire tous les exemples présentés dans le cours dans un seul fichier "examples_POO.py".

CHAQUE LIGNE DE CODE DOIT ÊTRE COMMENTÉE.

Exemple 1 (diapositive 10) Écrire les deux fonctions **greetReturn** et **greetWithoutReturn** qui prennent un argument de chaîne de caractères en entrée. La première renvoie "*Bonjour*" avec le nom de la personne passée en argument d'entrée à la fonction. La seconde imprime directement dans la console "*Bonjour*" avec l'argument d'entrée de la fonction. Expliquer le résultat avec des commentaires.

Exemple 2 (diapositive 15) Créer une classe **Voiture**. Définir le constructeur de la classe avec trois paramètres d'entrée, dont l'un prend une valeur par défaut que vous définirez vous-même. Cette méthode va initialiser les trois attributs modele, prixHT et quantité. Définir les deux méthodes ajouter et retirer qui prennent un seul paramètre d'entrée pour ajouter et retirer une valeur saisie par l'utilisateur à l'attribut quantité. Créer deux objets Voiture (*peugeot* et *seat*), tester toutes vos méthodes et discuter du résultat avec des commentaires.

Exemple 3 (diapositive 28) Créer une classe **NoConstructor** qui ne prend pas de constructeur. Instancier deux objets *c1* et *c2* de cette classe. Afficher les objets *c1* et *c2* dans la console et expliquer les résultats avec des commentaires.

Exemple 4 (diapositive 29) Créer la classe **ClasseCount**. Définir un attribut de classe initialiser à zéro. Définir un constructeur qui prend un paramètre d'autoréférence. Dans le constructeur, il faudra incrémenter la valeur de l'attribut de classe et afficher le message "Call constructor" dans la console.

créer deux objets *o1* et *o2* de type **ClasseCount**. Imprimer dans la console la valeur de l'attribut de classe. Expliquer les résultats avec des commentaires.

Exemple 5 (diapositive 30) Créer la classe **GameCharacter**. Définir le constructeur qui initialise l'attribut nom. Créer deux objets (*thor* et *loki*) de la classe GameCharacter. Afficher les noms des deux objets dans la console. Expliquer les résultats avec des commentaires.

Exemple 6 (diapositive 38) Créer une classe **Personne**. Définir le constructeur qui initialise les deux attributs "nom" et "age". l'attribut "age" sera précédé d'un souligné. Définir une méthode d'affichage qui permet d'afficher les deux attributs de l'objet dans la console. Tester l'accès direct aux attributs en les affichant directement dans la console. Expliquer les résultats avec des commentaires.

Exemple 7 (diapositive 39) Ajouter un souligné au début de l'attribut "age" (Remplacer _age dans l'exemple précédent par deux soulignés __age) Tester l'accès direct aux attributs en les affichant dans la console. Discuter des résultats avec des commentaires.

Exemple 8 (diapositive 40) Dans la classe **Personne**, définir les méthodes d'accès et de modification des attributs de l'objet. Tester les méthodes. Expliquer les résultats avec des commentaires.

Exemple 9 (diapositive 44)

- Créer une classe **Vehicule**. Définir un constructeur qui initialise le "poids" et le "type de la boîte de vitesse du véhicule". Définissez une méthode spécifique à la classe qui affiche le message "la voiture se déplace..." lorsqu'elle est appelée.
- Créer une classe **Voiture** qui hérite de la classe **Vehicule**. Définir le constructeur qui appelle le constructeur de la classe parente et initialise l'attribut spécifique "nombre de passagers". Définir la méthode " allumer_radio " spécifique à la classe.
- Créer une classe **Camion** qui hérite de la classe **Véhicule**. Définir le constructeur qui appelle le constructeur de la classe mère et initialise l'attribut "cargo" spécifique à la classe. Définir la méthode " charge_cargo " spécifique à la classe.

Créer deux objets (*v1* de type **Voiture** et *c1* de type **Camion**). Tester les méthodes. Expliquer les résultats avec des commentaires.

Exemple 10 (diapositive 48)

- Créer la classe **Animal**. Définir le constructeur qui initialise l'age de l'animal. Définir deux méthodes : la méthode "type" qui affiche le message "type d'animal différent" dans la console. la méthode "age" qui affiche le message "âge de l'animal".
- créer la classe **Panda**. Définir les deux méthodes avec le même nom que dans la classe "Animal" : la méthode "type" qui affiche le message "Je suis un panda". la méthode "âge" qui affiche l'âge du panda.
- Créer la classe **Lion**. Définir les deux méthodes avec le même nom que dans la classe Animal et Panda : la méthode "type" qui affiche le message "Je suis un lion". la méthode "age" qui affiche l'âge du lion.
- Créer trois objets :
  - un objet de type Animal.
  - un objet de type Panda.
  - un objet de type Lion.
- Appeler les deux méthodes (type et age) pour chaque objet.
- Expliquer les résultats avec des commentaires.

2 Deuxième partie : Jeu avec la POO

Dans cette partie, il faudra coder un jeu qui consiste à deviner un nombre aléatoire généré par l'ordinateur.
(utiliser la bibliotèque random - https://docs.python.org/3/library/random.html).

Le jeu sera codé en deux étapes :

- La première étape est de créer une classe **DevinerNum**. Dans cette classe, vous devrez définir les deux méthodes suivantes :
  - un constructeur qui vous permet d'initialiser les paramètres du jeu (générer un nombre aléatoire entre 0 et un "nombre" comme argument d'entrée).
  - une méthode "deviner", qui prends les deux arguments self et numero et retourne "Correct" si l'utilisateur trouve le nombre, "le numéro à trouver est plus grand" si l'utilisateur entre une valeur plus petite que le nombre généré, et "le numéro à trouver est plus petit" si l'utilisateur entre une valeur plus grande.
- La deuxième étape sera de coder l'algorithme principal pour jouer le jeu. Le joueur gagne s'il peut deviner sans dépasser le nombre d'essais autorisés.

Il faudra obliger l'utilisateur à entrer un nombre de type **numérique** pour définir la plage de nombres souhaitée pour générer le nombre aléatoire. Créer une instance de la classe **DevinerNum**. Initialiser les variables du jeu (nombre d'essais). Écrire le code qui permettra à un utilisateur de jouer.

CHAQUE LIGNE DE CODE DOIT ÊTRE COMMENTÉE.

3 Troisième Partie : Maitriser la POO

3.1 Classe Point

Créer une classe Point décrivant un point du plan par deux coordonnées entières ("x" et "y").
- Donner un constructeur.
- Définir une méthode "additionner" prenant un argument **Point** et renvoie un **Point** résultant de l'addition (coordonnée à coordonnée) de l'argument avec l'objet de référence.
- Définir la méthode "displayPoint" qui renvoie une chaine de caractères de façon à afficher les coordonnées d'un **Point** sous forme de couple. exemple : pour x=1 et y=2, la méthode doit renvoyer "(1,2)".
Créer deux objets *P1* et *P2*. Tester les méthodes additionner et displayPoint.

3.2 Pile

Créer une classe **Pile** permettant de gérer une pile d'objet de type **Point**. Une pile (en anglais a stack) est une structure de données reposant sur le principe de "dernier arrivé, premier sorti". On parle également de mode LIFO : Last In, First Out.

Pour cela :

- on définira un constructeur à un argument qui précisera le nombre maximum d'éléments de la pile.
- les points seront conservés dans une liste.
- le nombre maximum et le nombre courant d'éléments seront stockés dans deux autres attributs.
- on définira la méthode booléenne pleine (resp. vide) qui vérifie si la pile est pleine (resp. vide).
- on définira une méthode "empiler" à un argument Point qui ajoute l'objet de type Point sur la pile.
- on définira une méthode "depiler" sans argument qui renvoie le sommet de pile tout en dépilant.
- on définira une méthode "getCourant" sans argument qui renvoie le point au sommet de pile et appelle la méthode "displayPoint" pour afficher le point dans la console.

Tester : Créer un objet p de la classe **Pile**, empiler et depiler des objets de type **Point** et expliquer le code avec des commentaires.

3.3 Classe Segment / Classe Cercle

- Créer une classe **Segment** décrivant un segment du plan par un point origine et un point final. Donner un constructeur et deux méthodes d'accès à ses attributs. - Créer une classe **Cercle** décrivant un cercle du plan par un point origine et un rayon. Donner un constructeur et deux méthodes d'accès à ses attributs.
- Définir dans chacune des classes **Segment** et **Cercle** les deux méthodes suivantes :

- "dessiner" avec un argument **Pile** dont le rôle est d'afficher le type de forme concerné ainsi que ses attributs **Point** translatés par la transformation courante donnée par le sommet de pile.
- "deplacer" avec un argument **Point** dont le rôle est d'ajouter l'argument **Point** aux attributs **Point** de la classe concernée.

Tester ce programme pour voir s'il correspond bien à vos attentes (créer une pile, empiler des **Point**, créer, dessiner et déplacer un **Segment**, dépiler un **Point** de la pile, créer, dessiner et déplacer un **Cercle**).

3.4 Classe Forme

Modifier le programme précédent de façon à introduire une classe Forme mère de **Segment** et de **Cercle**. En particulier, vous déplacerez les caractéristiques (structurelles et comportementales) communes à **Segment** et à **Cercle** au niveau de la classe **Forme**. Tester.

3.5 Classe Image

Créer une classe **Image**, nouvelle fille de **Forme** et collection de plusieurs segments, cercles et autres formes que l'on pourrait introduire plus tard. Décrire une **Image** par un point origine, une liste des formes qui la composent et un nombre de formes qui la composent. Donner un constructeur avec un argument **Point**. Définir la méthode ajouter qui ajoute une forme à l'image, dessiner et deplacer dont on extrapolera le rôle. Tester.

4 Code principal

Rassemblez tous vos tests en un seul code principal qui respecte l'algorithme ci-dessous :

- Le programme doit afficher un menu contenant un numéro pour chaque partie de cet énoncé suivi du titre de la partie. Exemple :
  1- Commencer avec la POO
  2- Jeu avec la POO
  3- Maitriser la POO
  4-Quitter

- L'utilisateur choisit la partie qu'il veut tester en entrant son numéro. Si le numéro saisi n'est pas dans le menu, un message d'erreur s'affiche *"le numéro saisi n'existe pas, veuillez saisir un numéro valide"*.

- Dans chaque partie, le programme affiche une description et montre toutes les fonctions qu'elle contient.

5 Rendu du projet

Envoyez votre travail sous la forme d'une archive(.zip) à l'adresse nassib.abdallah@univ-angers.fr.
Le nom de cette archive sera NOM_Prenom_Projet.zip (exemple ABDALLAH_Nassib_Projet.zip).
Cette archive contiendra un répertoire dont le nom sera NOM_Prenom_Projet.
Le contenu de ce répertoire sera organisé ainsi :

- Les noms des fichiers/dossiers seront en minuscules et sans caractère spéciaux (-,+,é,è,etc.).

- un fichier lisezmoi.txt **(format txt simple, pas de .docx ou .odt)** avec votre nom et prénom et indiquant tout point particulier mettant en valeur votre travail et sur lequel vous voulez attirer l'attention du correcteur.