Nom : **Prénom :**

TP d'introduction à la programmation orientée objet (POO)

1 Préambule

On considère les variables suivantes :

```
quantite = 10
chiffre_pairs = [0, 2, 4, 6, 8]
humain = {'nom': 'Holzer', 'prenom': 'Marie', 'age': 16}
```

1. De tête, de quels types sont les variables `quantite`, `chiffres_pairs` et `humain` ?

2. (a) Ouvrir votre éditeur Python.
   (b) Créer un nouveau fichier et l'enregistrer tout de suite dans un espace de travail que vous pourrez consulter toute l'année.
   (c) Y copier le programme ci-dessus.
   (d) Exécuter immédiatement le programme pour vérifier qu'il n'y a pas d'erreur. Ceci est à faire régulièrement !

3. Taper dans la console de votre éditeur :

   ```
   type(quantite)
   type(chiffre_pairs)
   type(humain)
   ```

   Que se passe-t-il ? Commenter le résultat :

4. Ajouter à votre script (donc dans le fichier Python, **pas** dans la console) les instructions :

   ```
   type(quantite)
   type(chiffre_pairs)
   type(humain)
   ```

   Que se passe-t-il ? Que faut-il faire pour obtenir le même résultat qu'à la question précédente ?

–

En Python, les tableaux, les dictionnaires (et les autres types de variables) sont des classes. La notion de classe va en quelque sorte nous permettre de généraliser la notion de type. Les classes permettent de :

- définir des objets (comme les tableaux, les dictionnaires, les entiers, etc.)
- travailler dessus pour les modifier ou en extraire des informations.

On dit que `chiffre_pairs` est un **objet de la classe** `list`.

On dit que `humain` est un **objet de la classe** `dict`.

2 Découvertes à partir d'un exemple

Robert le concessionnaire possède des voitures.

Ces voitures ont des caractéristiques : une marque, un modèle, une couleur, une puissance et un kilométrage.

Sur ces voitures, le garagiste souhaite faire des traitements. Par exemple, il souhaite préciser s'il s'agit d'une voiture de sport (il considère qu'une voiture est une voiture de sport si la puissance du moteur dépasse les 180 ch).

Robert le concessionnaire vient de recevoir :

- Une Porsche Piment noire neuve dont la motorisation fait 340 ch ;
- Une Phiat Ourson rouge avec 10 000 km au compteur dont la motorisation fait 50 ch.

2.1 Approche naïve

Exercice 1

1. Proposer une structure de données permettant de stocker les informations liées à une voiture. Illustrer ceci en prenant comme exemple l'une des deux voitures reçues par Robert.

2. À partir de cette modélisation, écrire une fonction `est_sportive` qui renvoie un booléen indiquant si la voiture passée en paramètre est sportive (`True` si elle est sportive, `False` sinon).

3. Écrire un test permettant de vérifier le bon fonctionnement de la fonction écrite précédemment.

–

L'inconvénient d'une telle modélisation est la saisie un peu lourde de chaque modèle de voiture mais surtout, en cas de modification de la structure de donnée, les fonctions écrites ne seraient plus cohérentes. Il faudrait ainsi changer toutes les voitures de façon à ce qu'elles reflètent le nouveau modèle. On pourrait imaginer ajouter un paramètre utile, par exemple `automatique`. Il paraît inenvisageable de modifier manuellement des milliers d'enregistrements de voitures. On peut aussi facilement faire une coquille d'orthographe (comme écrire *puisance* au lieu de *puissance*) et nos fonctions renverraient des erreurs.

–

2.2 Approche par des objets

Nous allons désormais définir les voiture précédentes comme des objets. Il faut donc pour cela définir une classe qui permettra de créer ces objets. Pour créer une classe que l'on pourrait appeler Voiture on utilise la syntaxe suivante (sans oublier le *docstring* correspondant pour alimenter la documentation de notre programme) :

```
class Voiture:
    '''Définition d'une voiture'''
```

Remarque : selon la convention Python PEP8 (`https://peps.python.org/pep-0008/`), le nom d'une classe s'écrit en *CamelCase* (`https://fr.wikipedia.org/wiki/Camel_case`).

Exercice 2

1. Copier le script ci-dessous dans PythonTutor (`https://pythontutor.com`) afin de visualiser l'exécution de ce programme pas-à-pas.

```
class Voiture:
    '''Définition d'une voiture'''

# Instruction permettant de créer un objet
ma_voiture = Voiture()

# Instruction permettant de donner une marque à la voiture créée
ma_voiture.marque = 'Phiat'

```

→ Le script précédent a permis :

- de créer un objet `ma_voiture`. On dit que `ma_voiture` est une **instance** de la classe `Voiture`.
- d'associer à cet objet une caractéristique appelée `marque` et de lui donner la valeur `Phiat`. On dit qu'on vient d'ajouter à `ma_voiture` un **attribut** `marque` qui prend la valeur `Phiat`.

2. Ajouter à votre programme le code suivant :

```
ma_voiture.modele = 'Piment'
ma_voiture.couleur = 'Noire'
ma_voiture.marque = 'Porsche'
ta_voiture = Voiture()

print(ma_voiture.marque)
print(ma_voiture)
print(ta_voiture)
```

(a) Sans exécuter le programme, quelles sont les marques des deux voitures ?

(b) Faire des affichages pour vérifier.

3. Saisir une suite d'instructions permettant de créer la voiture Phiat Ourson rouge avec 10 000 km au compteur dont la motorisation fait 50 ch. Pour indiquer la puissance du moteur et le kilométrage on créera respectivement les attributs `puissance` et `km`.

4. Voici quelques instructions supplémentaires.

```
ma_caisse = Voiture()
ma_caisse.marque = 'TesteLa'
ma_caisse.modele = 'S'
ma_caisse.km = 2000
nv_voiture = ma_caisse
nv_voiture.km = 0
```

(a) Sans exécuter ces lignes avec Python, quels sont les valeurs des attributs des objets `ma_caisse` et `nv_voiture` ?

(b) Exécuter ce code avec Python pour vérifier votre réponse.

(c) En exécutant ce même script avec PythonTutor, proposer une explication au résultat observé.

$\rightarrow$ Si `nom_objet` est une instance d'une classe, `nom_objet` est appelé par abus de langage un objet alors qu'en réalité c'est en fait une **référence** à l'objet, c'est-à-dire l'adresse mémoire dans l'ordinateur.

5. Toutes les instances de la classe `Voiture` ont-elles les mêmes attributs ? Est-ce que ça peut poser problème ?

6. La fonction `est_sportive` vous semble-t-elle judicieusement créée ?