TP – Polymorphisme en Python

Nom et Prénom : .....Nasser Ait Gaouz.....

Filière / Groupe : .....NSC.....

Exercice 1 : Polymorphisme avec les animaux

Objectif pédagogique :
Explorer le polymorphisme en Python à travers l'utilisation d'interfaces communes et de substitutions dynamiques.

Questions

1. Expliquer le rôle de la classe Animal.

2. Pourquoi la méthode parler() est-elle redéfinie dans chaque sous-classe ?

3. Expliquer le concept de polymorphisme illustré par la liste d'animaux.

4. Qu'est-ce que le duck typing ? Donner un exemple à partir du code.

5. Quel est le résultat obtenu lors de l'exécution du programme ?

Captures d'écran

➤ Code source de l'exercice 1

animaux.py > ...

```python
class Animal:
    def parler(self):
        raise NotImplementedError("Cette méthode doit être redéfinie")

class Chien(Animal):
    def parler(self):
        return "Ouaf !"

class Chat(Animal):
    def parler(self):
        return "Miaou !"

class Vache(Animal):
    def parler(self):
        return "Meuh !"

# Duck typing
class Robot:
    def parler(self):
        return "Bip bop !"

def faire_parler(animal):
    print(animal.parler())

if __name__ == "__main__":
    animaux = [Chien(), Chat(), Vache(), Robot()]
    for a in animaux:
        faire_parler(a)
```

➤ Résultat de l'exécution du programme

```
C:\Users\HP\Downloads\TP_Polymorphisme_Python\exercice1_animaux>C:\User
ython.exe c:/Users/HP/Downloads/TP_Polymorphisme_Python/exercice1_anima
Ouaf !
Miaou !
Meuh !
Bip bop !

C:\Users\HP\Downloads\TP_Polymorphisme_Python\exercice1_animaux>
```

Exercice 2 : Polymorphisme avec les formes géométriques

Objectif pédagogique :
Approfondir le polymorphisme en Python en créant une hiérarchie de formes capables de calculer leur aire via une interface commune.

Questions

1. Quel est le rôle de la classe abstraite Forme ?

2. À quoi sert le décorateur @abstractmethod ?

3. Expliquer le polymorphisme observé lors de l'affichage des formes.

4. Pourquoi peut-on stocker des objets de types différents dans la même liste ?

5. Donner l'avantage de la méthode __str__ dans ce programme.

Captures d'écran

➤ Code source de l'exercice 2

```python
from abc import ABC, abstractmethod
import math

class Forme(ABC):
    @abstractmethod
    def aire(self):
        pass

    def __str__(self):
        return f"{self.__class__.__name__} - aire : {self.aire():.2f}"

class Cercle(Forme):
    def __init__(self, rayon):
        self.rayon = rayon

    def aire(self):
        return math.pi * self.rayon ** 2

class Rectangle(Forme):
    def __init__(self, largeur, hauteur):
        self.largeur = largeur
        self.hauteur = hauteur

    def aire(self):
        return self.largeur * self.hauteur

class Triangle(Forme):
    def __init__(self, base, hauteur):
        self.base = base
        self.hauteur = hauteur

```

```python
    def aire(self):
        return 0.5 * self.base * self.hauteur

class Carre(Rectangle):
    def __init__(self, cote):
        super().__init__(cote, cote)

if __name__ == "__main__":
    formes = [
        Cercle(3),
        Rectangle(4, 5),
        Triangle(6, 2),
        Carre(4)
    ]

    for f in formes:
        print(f)
```

➤ Résultat de l'exécution du programme

```
C:\Users\HP\Downloads\TP_Polymorphisme_Python\exercice2_formes>C:\Users\HP\AppData\Local\Prog
exe c:/Users/HP/Downloads/TP_Polymorphisme_Python/exercice2_formes/formes.py
Cercle - aire : 28.27
Rectangle - aire : 20.00
Triangle - aire : 6.00
Carre - aire : 16.00

C:\Users\HP\Downloads\TP_Polymorphisme_Python\exercice2_formes>
```