Programmation évènementielle en Python

L'orientée objet en Python

Aurel Megnigbeto

Université d'Abomey-Calavi

École Nationale d'Économie Appliquée et de Management

11 mars 2025

Plan

Plan

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- . . .

Rappel séance précédente

- Programmation
- Langage interprété vs compilé vs semi-compilé
- Structures de données
- Programmation séquentielle
  1. Définition
  2. Structure de base
- Programmation évènementielle
  1. Définition
  2. Structure de base
- Programmation orientée objet
- ...

Plan

Méthodes d'évaluation du cours

- +/- partielles
- Un projet de groupe de 2 personnes
- Un projet individuel
- L'examen final

Méthodes d'évaluation du cours

- +/- partielles
- Un projet de groupe de 2 personnes
- Un projet individuel
- L'examen final

Méthodes d'évaluation du cours

- +/- partielles
- Un projet de groupe de 2 personnes
- Un projet individuel
- L'examen final

Méthodes d'évaluation du cours

- +/- partielles
- Un projet de groupe de 2 personnes
- Un projet individuel
- L'examen final

Plan

Définition

Programmation Orientée Objet

La programmation orientée objet est un paradigme de programmation qui utilise des objets et leurs comportements pour concevoir des applications et des programmes.

Par les objets, on peut représenter :

- des entités du monde réel (voiture, personne, etc.)
- des entités abstraites (fichier, connexion, etc.)
- et les opérations qu'ils peuvent effectuer

Définition

Programmation Orientée Objet

La programmation orientée objet est un paradigme de programmation qui utilise des objets et leurs comportements pour concevoir des applications et des programmes.

Par les objets, on peut représenter :

- des entités du monde réel (voiture, personne, etc.)
- des entités abstraites (fichier, connexion, etc.)
- et les opérations qu'ils peuvent effectuer

Définition

Programmation Orientée Objet

La programmation orientée objet est un paradigme de programmation qui utilise des objets et leurs comportements pour concevoir des applications et des programmes.

Par les objets, on peut représenter :

- des entités du monde réel (voiture, personne, etc.)
- des entités abstraites (fichier, connexion, etc.)
- et les opérations qu'ils peuvent effectuer

Définition

Programmation Orientée Objet

La programmation orientée objet est un paradigme de programmation qui utilise des objets et leurs comportements pour concevoir des applications et des programmes.

Par les objets, on peut représenter :

- des entités du monde réel (voiture, personne, etc.)
- des entités abstraites (fichier, connexion, etc.)
- et les opérations qu'ils peuvent effectuer

Définition

À retenir :

- Façon de représenter les données et les opérations sur ces données
- Les objets ont des propriétés et des comportements (attributs et méthodes)
- Les objets suivent un modèle prédéfini (classe)
- Le processus de création d'objets à partir de classes est appelé instanciation

Définition

À retenir :

- Façon de représenter les données et les opérations sur ces données
- Les objets ont des propriétés et des comportements (attributs et méthodes)
- Les objets suivent un modèle prédéfini (classe)
- Le processus de création d'objets à partir de classes est appelé instanciation

Définition

À retenir :

- Façon de représenter les données et les opérations sur ces données
- Les objets ont des propriétés et des comportements (attributs et méthodes)
- Les objets suivent un modèle prédéfini (classe)
- Le processus de création d'objets à partir de classes est appelé instanciation

Définition

À retenir :

- Façon de représenter les données et les opérations sur ces données
- Les objets ont des propriétés et des comportements (attributs et méthodes)
- Les objets suivent un modèle prédéfini (classe)
- Le processus de création d'objets à partir de classes est appelé instanciation

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- . . .

Classes et objets

Beaucoup de langages de programmation supportent la POO (Python, Java, C++, etc.). Mais chaque langage a ses particularités pour implémenter les concepts de la POO.
Néanmoins, les concepts de base sont les mêmes :

- Classe
- Objet
- Attributs
- Méthodes
- Héritage
- Polymorphisme
- Encapsulation
- Abstraction
- ...

Plan

Programmation Orienté Objet

Classe

Définition

Une classe est un modèle pour créer des objets. Elle définit les attributs et les méthodes que les objets auront.

En d'autres termes, une classe est un patron que le développeur définit pour créer des objets.
Exemple :

- Classe : Voiture
- Attributs : marque, modèle, couleur, etc.
- Méthodes : démarrer, accélérer, freiner, etc.

Programmation Orienté Objet

Classe

Définition

Une classe est un modèle pour créer des objets. Elle définit les attributs et les méthodes que les objets auront.

En d'autres termes, une classe est un patron que le développeur définit pour créer des objets.
Exemple :

- Classe : Voiture
- Attributs : marque, modèle, couleur, etc.
- Méthodes : démarrer, accélérer, freiner, etc.

Programmation Orienté Objet

Classe

Définition

Une classe est un modèle pour créer des objets. Elle définit les attributs et les méthodes que les objets auront.

En d'autres termes, une classe est un patron que le développeur définit pour créer des objets.
Exemple :

- Classe : Voiture
- Attributs : marque, modèle, couleur, etc.
- Méthodes : démarrer, accélérer, freiner, etc.

Programmation Orienté Objet

Classe

Définition

Une classe est un modèle pour créer des objets. Elle définit les attributs et les méthodes que les objets auront.

En d'autres termes, une classe est un patron que le développeur définit pour créer des objets.
Exemple :

- Classe : Voiture
- Attributs : marque, modèle, couleur, etc.
- Méthodes : démarrer, accélérer, freiner, etc.

Programmation Orienté Objet

Classe

En Python :

- Une classe est définie par le mot clé `class`
- Une classe a une méthode spéciale `__init__` qui est appelée lors de l'instanciation
- et d'autres méthodes spéciales (`__str__`, `__repr__`, etc.)

Programmation Orienté Objet

Classe

En Python :

- Une classe est définie par le mot clé `class`
- Une classe a une méthode spéciale `__init__` qui est appelée lors de l'instanciation
- et d'autres méthodes spéciales (`__str__`, `__repr__`, etc.)

Programmation Orienté Objet

Classe

En Python :

- Une classe est définie par le mot clé `class`
- Une classe a une méthode spéciale `__init__` qui est appelée lors de l'instanciation
- et d'autres méthodes spéciales (`__str__`, `__repr__`, etc.)

Plan

Programmation Orienté Objet

Attributs et Méthodes

Attributs

Les attributs sont des variables qui définissent les propriétés de l'objet. Mais pas que,les attributs peuvent représenter les propriétés d'une classe. Dans ce cas, ils sont appelés attributs de classe.

Méthodes

Les méthodes sont des fonctions qui définissent les opérations que l'objet peut effectuer. (Le comportement de l'objet)
Les méthodes peuvent aussi représenter les opérations que la classe peut effectuer. Dans ce cas, elles sont appelées méthodes de classe et ne peuvent manipuler que les attributs de classe.

Programmation Orienté Objet

Attributs et Méthodes

Attributs

Les attributs sont des variables qui définissent les propriétés de l'objet. Mais pas que,les attributs peuvent représenter les propriétés d'une classe. Dans ce cas, ils sont appelés attributs de classe.

Méthodes

Les méthodes sont des fonctions qui définissent les opérations que l'objet peut effectuer. (Le comportement de l'objet)
Les méthodes peuvent aussi représenter les opérations que la classe peut effectuer. Dans ce cas, elles sont appelées méthodes de classe et ne peuvent manipuler que les attributs de classe.

Programmation Orienté Objet

Attributs et Méthodes

Exemple :

```python
class Car:
    total_cars = 0

    def __init__(self, provider, model):
        self.provider = provider
        self.model = model
        Car.total_cars += 1

    def start(self):
        print(f"{self.provider} {self.model} started")

    @staticmethod
    def class_name():
        return "Car"

    @classmethod
    def display_total_cars(cls):
        print(f"Total cars created: {cls.total_cars}")
```

Listing – Attributs et Méthodes d'une classe

Plan

Programmation Orienté Objet

Instanciation

Définition

L'instanciation est le processus de création d'un objet à partir d'une classe. Ce processus permet de rendre concret le modèle défini par la classe.

Conséquences :

- C'est sur l'objet que les opérations sont effectuées.
- Les propriétés de l'objet modifiées ne sont pas partagées avec les autres objets de la même classe.

Programmation Orienté Objet

Instanciation

Définition

L'instanciation est le processus de création d'un objet à partir d'une classe. Ce processus permet de rendre concret le modèle défini par la classe.

Conséquences :

- C'est sur l'objet que les opérations sont effectuées.
- Les propriétés de l'objet modifiées ne sont pas partagées avec les autres objets de la même classe.

Programmation Orienté Objet

Instanciation

Définition

L'instanciation est le processus de création d'un objet à partir d'une classe. Ce processus permet de rendre concret le modèle défini par la classe.

Conséquences :

- C'est sur l'objet que les opérations sont effectuées.
- Les propriétés de l'objet modifiées ne sont pas partagées avec les autres objets de la même classe.

Programmation Orienté Objet

Instanciation

Exemple :

```
class Car:
    def __init__(self, company, year):
        self.company = company
        self.year = year
        self.color = None
        self.price = None
        self.model = None

c1 = Car("Toyota", 2020)
c2 = Car("Honda", 2021)
```

Listing – Instanciation d'une classe

Plan

Programmation Orienté Objet

Héritage

Défintion

L'héritage est un mécanisme qui permet à une classe nommée classe fille de récupérer les attributs et les méthodes d'une autre classe nommée classe mère.

La classe mère peut quand même décider de ce qu'elle veut partager avec ses classes filles à travers les mécanismes d'encapsulation.

Programmation Orienté Objet

Héritage

Exemple :

```
class Moto:
    def __init__(self, nbr_roues: int):
        self.nbr_roues = nbr_roues
        self.immatriculation = None


class MotoElectrique(Moto):
    def __init__(self, nbr_roues: int, autonomie: int):
        super().__init__(nbr_roues)
        self.autonomie = autonomie

m1 = MotoElectrique(2, 100)
```

Listing – Héritage d'une classe

Plan

Programmation Orienté Objet

Polymorphisme

Définition

Le polymorphisme est un mécanisme qui permet à une classe de fournir une même interface pour des sous-classes qui pourraient avoir des implémentations différentes.

Par exemple, une méthode peut être définie dans une classe mère et redéfinie dans une classe fille : (surcharge).

Programmation Orienté Objet

Exemple de polymorphisme

```
class Forme:
    def aire(self):
        pass
class Carre(Forme):
    def __init__(self, cote):
        super().__init__()
        self.cote = cote

    def aire(self):
        return self.cote ** 2
class Cercle(Forme):
    def __init__(self, rayon):
        super().__init__()
        self.rayon = rayon

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

Listing – Surcharge d'une méthode de classe

Plan

Programmation Orienté Objet

Encapsulation

Définition

L'encapsulation est un mécanisme qui permet de cacher les détails d'implémentation d'une classe à l'extérieur de la classe.

- La classe étant considérée ici comme une boîte noire qui n'expose que ce qu'elle veut exposer.
- Python ne propose pas de mécanisme de protection des attributs et méthodes.
- Mais il existe des conventions pour indiquer que certains attributs ou méthodes ne doivent pas être modifiés ou utilisés à l'extérieur de la classe.

Programmation Orienté Objet

Encapsulation

Définition

L'encapsulation est un mécanisme qui permet de cacher les détails d'implémentation d'une classe à l'extérieur de la classe.

- La classe étant considérée ici comme une boîte noire qui n'expose que ce qu'elle veut exposer.
- Python ne propose pas de mécanisme de protection des attributs et méthodes.
- Mais il existe des conventions pour indiquer que certains attributs ou méthodes ne doivent pas être modifiés ou utilisés à l'extérieur de la classe.

Programmation Orienté Objet

Encapsulation

Définition

L'encapsulation est un mécanisme qui permet de cacher les détails d'implémentation d'une classe à l'extérieur de la classe.

- La classe étant considérée ici comme une boîte noire qui n'expose que ce qu'elle veut exposer.
- Python ne propose pas de mécanisme de protection des attributs et méthodes.
- Mais il existe des conventions pour indiquer que certains attributs ou méthodes ne doivent pas être modifiés ou utilisés à l'extérieur de la classe.

Programmation Orienté Objet

Encapsulation

Définition

L'encapsulation est un mécanisme qui permet de cacher les détails d'implémentation d'une classe à l'extérieur de la classe.

- La classe étant considérée ici comme une boîte noire qui n'expose que ce qu'elle veut exposer.
- Python ne propose pas de mécanisme de protection des attributs et méthodes.
- Mais il existe des conventions pour indiquer que certains attributs ou méthodes ne doivent pas être modifiés ou utilisés à l'extérieur de la classe.

Programmation Orienté Objet

Exemple d'encapsulation

```
public class Voiture {
  private int  Compteur;
  private String Immat ;

  public int getCompteur () { ...}
  public void setCompteur (int c) {...}
  public  Voiture (String im) {...}
  public boolean Rouler (int Km) {...}

  private boolean ChangerPneu () {... }
}
```

Plan

Programmation Orienté Objet

Abstraction

Définition

L'abstraction est un mécanisme qui permet de représenter les caractéristiques essentielles d'un objet sans tenir compte des détails d'implémentation.

Dans certains cas, on parle de :

- Classe abstraite (une classe qui ne peut pas être instanciée car ayant une méthode abstraite)
- Méthode abstraite (une méthode qui n'a pas d'implémentation)
- Interface (une classe abstraite qui ne contient que des méthodes abstraites)
- D'implémentation (une classe qui hérite et propose des implémentation d'une interface)

Programmation Orienté Objet

Abstraction

Définition

L'abstraction est un mécanisme qui permet de représenter les caractéristiques essentielles d'un objet sans tenir compte des détails d'implémentation.

Dans certains cas, on parle de :

- Classe abstraite (une classe qui ne peut pas être instanciée car ayant une méthode abstraite)
- Méthode abstraite (une méthode qui n'a pas d'implémentation)
- Interface (une classe abstraite qui ne contient que des méthodes abstraites)
- D'implémentation (une classe qui hérite et propose des implémentation d'une interface)

Programmation Orienté Objet

Abstraction

Définition

L'abstraction est un mécanisme qui permet de représenter les caractéristiques essentielles d'un objet sans tenir compte des détails d'implémentation.

Dans certains cas, on parle de :

- Classe abstraite (une classe qui ne peut pas être instanciée car ayant une méthode abstraite)
- Méthode abstraite (une méthode qui n'a pas d'implémentation)
- Interface (une classe abstraite qui ne contient que des méthodes abstraites)
- D'implémentation (une classe qui hérite et propose des implémentation d'une interface)

Programmation Orienté Objet

Abstraction

Définition

L'abstraction est un mécanisme qui permet de représenter les caractéristiques essentielles d'un objet sans tenir compte des détails d'implémentation.

Dans certains cas, on parle de :

- Classe abstraite (une classe qui ne peut pas être instanciée car ayant une méthode abstraite)
- Méthode abstraite (une méthode qui n'a pas d'implémentation)
- Interface (une classe abstraite qui ne contient que des méthodes abstraites)
- D'implémentation (une classe qui hérite et propose des implémentation d'une interface)

Programmation Orienté Objet

Abstraction

Définition

L'abstraction est un mécanisme qui permet de représenter les caractéristiques essentielles d'un objet sans tenir compte des détails d'implémentation.

Dans certains cas, on parle de :

- Classe abstraite (une classe qui ne peut pas être instanciée car ayant une méthode abstraite)
- Méthode abstraite (une méthode qui n'a pas d'implémentation)
- Interface (une classe abstraite qui ne contient que des méthodes abstraites)
- D'implémentation (une classe qui hérite et propose des implémentation d'une interface)

Programmation Orienté Objet

Exemple d'abstraction

```python
import abc, math
# Abstract Base Class - Shape
class Shape(metaclass=abc.ABCMeta):
    @abc.abstractmethod
    def area(self):
        pass
# Concrete Implementation - Rectangle
class Rectangle(Shape):
    def __init__(self, width, height):
        self._width = width
        self._height = height

    def area(self):
        return self._width * self._height
# Concrete Implementation - Circle
class Circle(Shape):
    def __init__(self, radius):
        self._radius = radius

    def area(self):
        return math.pi * (self._radius ** 2)
```

Listing – Exemple d'abstraction

Plan

Rappel des bases en Python

- structures de données en python
- instructions conditionnelles
- boucles
- fonctions
- exceptions
- modules
- environnement de développement

Rappel des bases en Python

- structures de données en python
- instructions conditionnelles
- boucles
- fonctions
- exceptions
- modules
- environnement de développement

Rappel des bases en Python

- structures de données en python
- instructions conditionnelles
- boucles
- fonctions
- exceptions
- modules
- environnement de développement

Rappel des bases en Python

- structures de données en python
- instructions conditionnelles
- boucles
- fonctions
- exceptions
- modules
- environnement de développement

Rappel des bases en Python

- structures de données en python
- instructions conditionnelles
- boucles
- fonctions
- exceptions
- modules
- environnement de développement

Rappel des bases en Python

- structures de données en python
- instructions conditionnelles
- boucles
- fonctions
- exceptions
- modules
- environnement de développement

Rappel des bases en Python

- structures de données en python
- instructions conditionnelles
- boucles
- fonctions
- exceptions
- modules
- environnement de développement

Plan

Travaux pratiques

Instruction 1

Soit une réglementation du gouvernement qui définit une banque comme une entreprise qui offre des services financiers.
Une banque formelle doit offrir tous les services suivants :

- Dépôt
- Retrait
- Consultation de solde

Créer une interface `Banque` qui définit les méthodes que doivent exposer les banques.

Travaux pratiques

Instruction 1

Soit une réglementation du gouvernement qui définit une banque comme une entreprise qui offre des services financiers.
Une banque formelle doit offrir tous les services suivants :

- Dépôt
- Retrait
- Consultation de solde

Créer une interface `Banque` qui définit les méthodes que doivent exposer les banques.

Travaux pratiques

Instruction 1

Soit une réglementation du gouvernement qui définit une banque comme une entreprise qui offre des services financiers.
Une banque formelle doit offrir tous les services suivants :

- Dépôt
- Retrait
- Consultation de solde

Créer une interface `Banque` qui définit les méthodes que doivent exposer les banques.

Travaux pratiques

Instruction 2

1. Créez une classe `BanqueBOA` qui implémente `Banque` avec les attributs privés *solde* et *titulaire_compte*.
2. Implémentez des méthodes pour le dépôt, le retrait et l'affichage des détails du compte.
3. Utilisez l'encapsulation pour protéger les attributs privés.

Travaux pratiques

Instruction 2

1. Créez une classe `BanqueBOA` qui implémente `Banque` avec les attributs privés *solde* et *titulaire_compte*.
2. Implémentez des méthodes pour le dépôt, le retrait et l'affichage des détails du compte.
3. Utilisez l'encapsulation pour protéger les attributs privés.

Travaux pratiques

Instruction 2

1. Créez une classe `BanqueBOA` qui implémente `Banque` avec les attributs privés *solde* et *titulaire_compte*.
2. Implémentez des méthodes pour le dépôt, le retrait et l'affichage des détails du compte.
3. Utilisez l'encapsulation pour protéger les attributs privés.

Travaux pratiques

Instruction 3

1. Créez une classe `BanqueDesRiches` qui hérite de la classe `BanqueBOA`.
2. Ajoutez un attribut *frais_gestion* = 1000 et redéfinissez la méthode de retrait pour prendre en compte les frais de gestion.
3. Lors d'un retrait, appliquez une soustraction des *frais_gestion* avant de soustraire le montant du retrait.

Travaux pratiques

Instruction 3

1. Créez une classe `BanqueDesRiches` qui hérite de la classe `BanqueBOA`.
2. Ajoutez un attribut *frais_gestion* $= 1000$ et redéfinissez la méthode de retrait pour prendre en compte les frais de gestion.
3. Lors d'un retrait, appliquez une soustraction des *frais_gestion* avant de soustraire le montant du retrait.

Travaux pratiques

Instruction 3

1. Créez une classe `BanqueDesRiches` qui hérite de la classe `BanqueBOA`.
2. Ajoutez un attribut $frais\_gestion = 1000$ et redéfinissez la méthode de retrait pour prendre en compte les frais de gestion.
3. Lors d'un retrait, appliquez une soustraction des *frais_gestion* avant de soustraire le montant du retrait.

Travaux pratiques

Instruction 4

1. Créez une classe `BanqueDEpargne` qui hérite de la classe `BanqueBOA`.
2. Ajoutez un attribut $taux\_interet = 0.1$ et redéfinissez la méthode de dépôt pour prendre en compte les intérêts.
3. Lors d'un dépôt, appliquez une addition des intérêts avant d'ajouter le montant du dépôt.

Travaux pratiques

Instruction 4

1. Créez une classe `BanqueDEpargne` qui hérite de la classe `BanqueBOA`.
2. Ajoutez un attribut $taux\_interet = 0.1$ et redéfinissez la méthode de dépôt pour prendre en compte les intérêts.
3. Lors d'un dépôt, appliquez une addition des intérêts avant d'ajouter le montant du dépôt.

Travaux pratiques

Instruction 4

1. Créez une classe `BanqueDEpargne` qui hérite de la classe `BanqueBOA`.
2. Ajoutez un attribut $taux\_interet = 0.1$ et redéfinissez la méthode de dépôt pour prendre en compte les intérêts.
3. Lors d'un dépôt, appliquez une addition des intérêts avant d'ajouter le montant du dépôt.

Fin

Fin, End