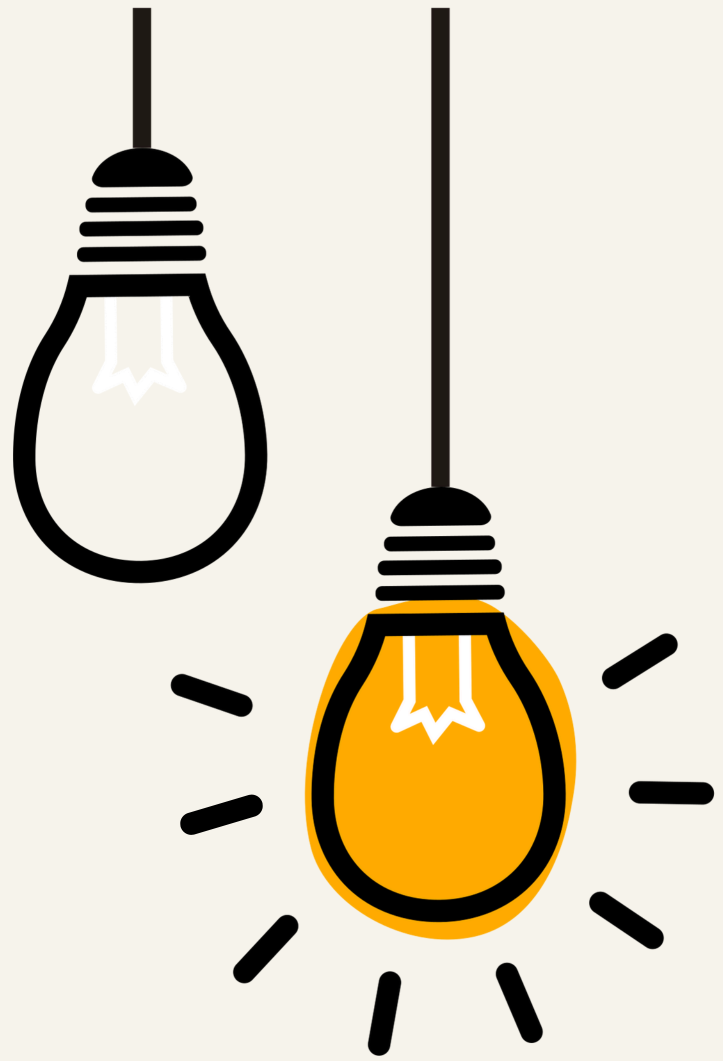


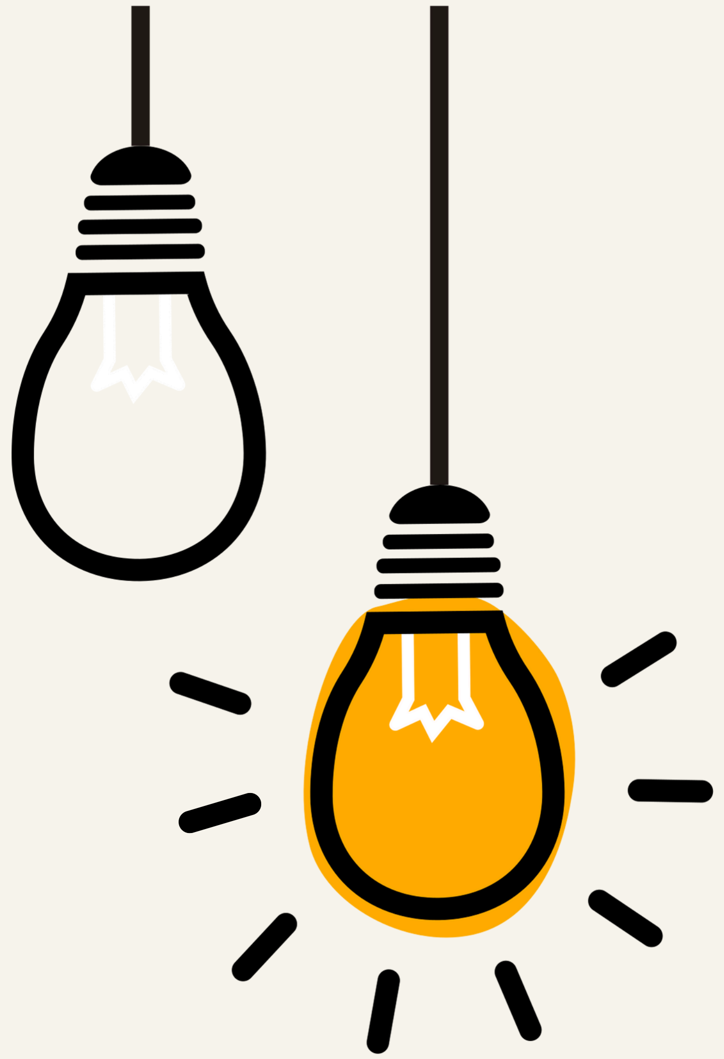
Filoger Comprehensive Python For AI Course 2024

Exercises 19 && 20

Deadline: 2024 24 October

Score: 400 + 100

Thursday - 2024 18 October



Exercises 19

Deadline: 2024 24 October

Github: 50

Score: 150

Thursday - 2024 18 October

QuestionI(Encapsulation)

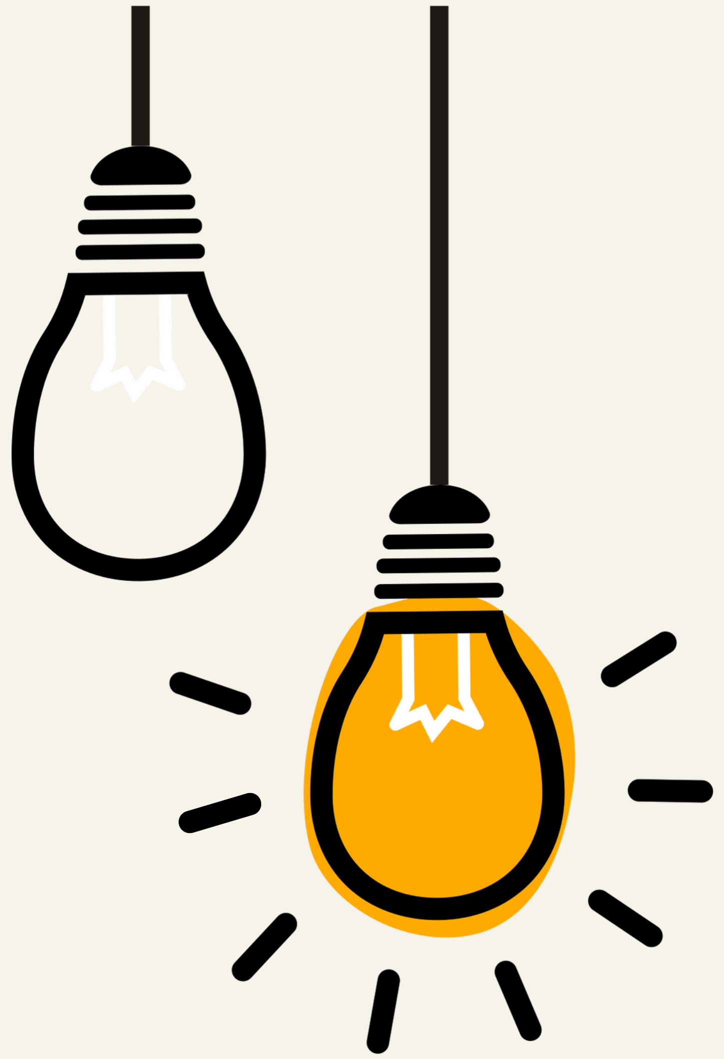
Design a class called BankAccount with three attributes:

- accountNumber (public), balance (protected), password (private).

1. In the BankAccount class, write methods to get and set the balance and password attributes, ensuring proper access control by using getters and setters where necessary.
2. Create a subclass called SavingsAccount, which should try to access the balance and password attributes directly. After encountering access issues, modify the subclass to use the getter and setter methods to access these attributes.
3. Finally, create an object of the BankAccount class and try to access and print the accountNumber, balance, and password attributes directly. Use the getter and setter methods when necessary for accessing the balance and password or changing them.

Write comment for:

- What happens when accessing each attribute directly.
- Why it is necessary to use getters and setters for certain attributes.



Exercises 20

Deadline: 2024 24 October

Github: 50

Score: 250

Thursday - 2024 18 October

QuestionI(OOP)

Create a Python project that involves a class named BankAccount with the following attributes and methods:

Attributes:

- account_holder: The name of the account holder.
- balance: The current balance of the account.
- account_number: A unique numeric identifier for the account.

QuestionI(OOP)

Methods:

- deposit(amount): Adds the specified amount to the account balance.
- withdraw(amount): Deducts the specified amount from the balance. If the amount exceeds the balance, it should raise a custom exception InsufficientFundsError.
- check_balance(): Prints the current balance.
- transfer_to(another_account, amount): Transfers a specified amount to another account, reducing the balance from the current account and adding it to another_account. If the balance is insufficient, raise the InsufficientFundsError.

Additionally, implement a custom exception class named InsufficientFundsError that is raised whenever a withdrawal or transfer operation exceeds the available balance.

Did you know ?

You automatically
lose the chances you don't take.
Trust yourself. You can do this.

Don't give up on your dreams :)