# ASSIGNMENT-6.4

NAME **: NITHIN VEMULA**

ROLL NO **: 2403a52355**

### Prompt:

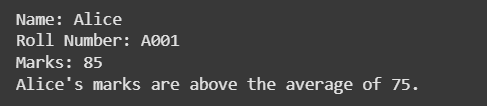
Start a Python class named Student with attributes name, roll\_number, and marks. Prompt GitHub Copilot to complete methods for displaying details and checking if marks are above average

### Code:

A screen shot of a computer program

AI-generated content may be incorrect.

### Output:



### Explaination:

This code defines a Python class called Student. Here's a breakdown:

class Student:: This line declares the beginning of a class definition named Student. Classes are blueprints for creating objects (instances).

\_\_init\_\_(self, name, roll\_number, marks):: This is the constructor method. It's automatically called when you create a new Student object.

self refers to the instance of the class being created.

name, roll\_number, and marks are parameters that you pass when creating a student object.

Inside the constructor, self.name = name, self.roll\_number = roll\_number, and self.marks = marks assign the values passed as arguments to the object's attributes (properties).

display\_details(self):: This is a method that belongs to the Student class.

self refers to the instance of the class calling the method.

It prints the name, roll\_number, and marks attributes of the specific student object.

is\_above\_average(self, average\_marks):: This method checks if a student's marks are above a given average.

self refers to the instance of the class.

average\_marks is a parameter representing the average mark to compare against.

It returns True if the student's marks are greater than average\_marks, and False otherwise.

student1 = Student("Alice", "A001", 85): This line creates an instance (object) of the Student class. It's named student1, and it's initialized with the name "Alice", roll number "A001", and marks 85.

student1.display\_details(): This line calls the display\_details method on the student1 object, which prints its details.

average\_marks = 75: This line sets a variable average\_marks to 75.

if student1.is\_above\_average(average\_marks):: This line calls the is\_above\_average method on the student1 object, passing average\_marks (75) as an argument. The if statement checks if the method returns True.

print(...): Based on the result of the is\_above\_average method, one of the two print statements is executed, indicating whether Alice's marks are above the average.

### Prompt-2:

Write the first two lines of a for loop to iterate through a list of numbers. Use a comment prompt to let Copilot suggest how to calculate and print the square of even numbers only.

### Code:

A black background with green text and numbers

AI-generated content may be incorrect.

### Output:

A black rectangular object with white text

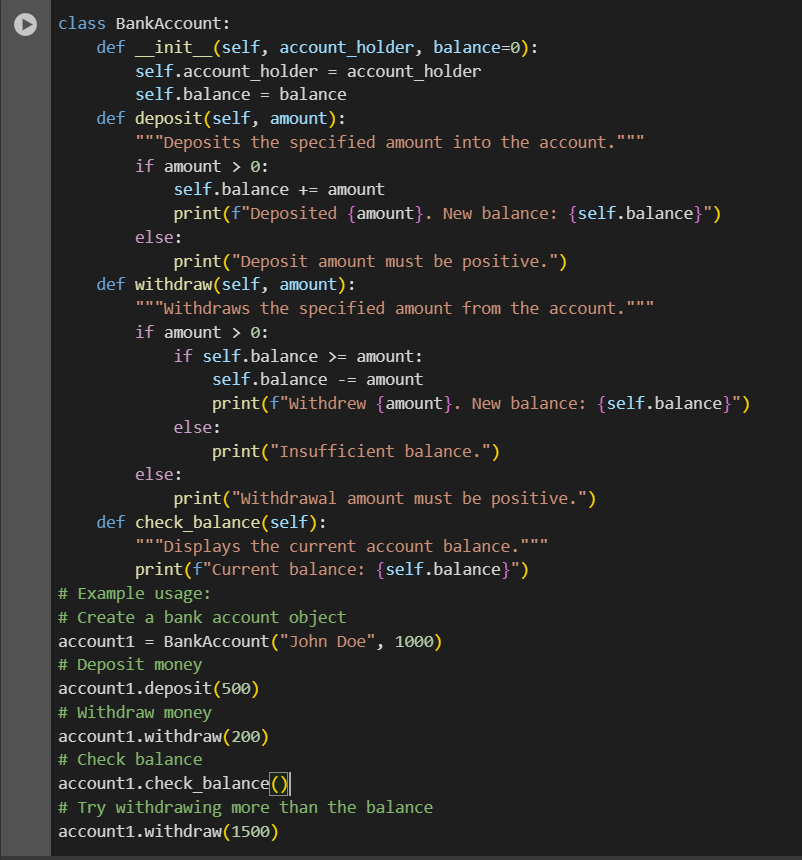
AI-generated content may be incorrect.

### Explaination:

### Prompt-3:

Create a class called BankAccount with attributes account\_holder and balance and complete methods for deposit(), withdraw(), and check for insufficient balance.

Code:



### Output:

A screenshot of a computer

AI-generated content may be incorrect.

### Explaination:

This code defines a Python class called BankAccount, which is designed to simulate a simple bank account. Here's a breakdown of the code:

* **class BankAccount:**: This line declares the beginning of the BankAccount class definition.
* **\_\_init\_\_(self, account\_holder, balance=0):**: This is the constructor method. It's called when you create a new BankAccount object.
  + self refers to the instance of the class being created.
  + account\_holder is a required parameter for the name of the account holder.
  + balance=0 is an optional parameter for the initial balance, defaulting to 0 if not provided.
  + Inside the constructor, self.account\_holder = account\_holder and self.balance = balance assign the provided values to the object's attributes.
* **deposit(self, amount):**: This method handles depositing money into the account.
  + self refers to the instance of the class.
  + amount is the amount to deposit.
  + It checks if the amount is greater than 0. If it is, the amount is added to the self.balance, and a confirmation message is printed. Otherwise, it prints an error message.
* **withdraw(self, amount):**: This method handles withdrawing money from the account.
  + self refers to the instance of the class.
  + amount is the amount to withdraw.
  + It first checks if the amount is greater than 0. If it is, it then checks if the self.balance is greater than or equal to the amount.
    - If there is enough balance, the amount is subtracted from self.balance, and a confirmation message is printed.
    - If there is not enough balance, it prints an "Insufficient balance" message.
  + If the amount is not positive, it prints an error message.
* **check\_balance(self):**: This method simply displays the current balance of the account.
  + self refers to the instance of the class.
  + It prints the current value of self.balance.
* **Example usage:** The code then demonstrates how to use the BankAccount class by:
  + Creating an account (account1 = BankAccount("John Doe", 1000)).
  + Depositing money (account1.deposit(500)).
  + Withdrawing money (account1.withdraw(200)).
  + Checking the balance (account1.check\_balance()).
  + Attempting to withdraw more than the balance to show the "Insufficient balance" message (account1.withdraw(1500)).

### Prompt-4:

Define a list of student dictionaries with keys name and score. Ask Copilot to write a while loop to print the names of students who scored more than 75.

### Code:

A screen shot of a computer code

AI-generated content may be incorrect.

### Output:

A black rectangular object with white text

AI-generated content may be incorrect.

### Explaination:

This code defines a list of dictionaries, where each dictionary represents a student with a name and a score. It then uses a while loop to iterate through this list and print the names of the students who have a score greater than 75.

Here's a breakdown:

students = [...]: This line creates a list named students. Each item in the list is a dictionary containing two key-value pairs: "name" and "score". This list holds the data for several students.

i = 0: This line initializes a variable i to 0. This variable will be used as an index to access the elements of the students list.

while i < len(students):: This is the while loop. It will continue to execute the code block inside it as long as the condition i < len(students) is true.

len(students) returns the number of items in the students list.

So, the loop will run as long as the index i is less than the total number of students, ensuring that we iterate through all elements in the list.

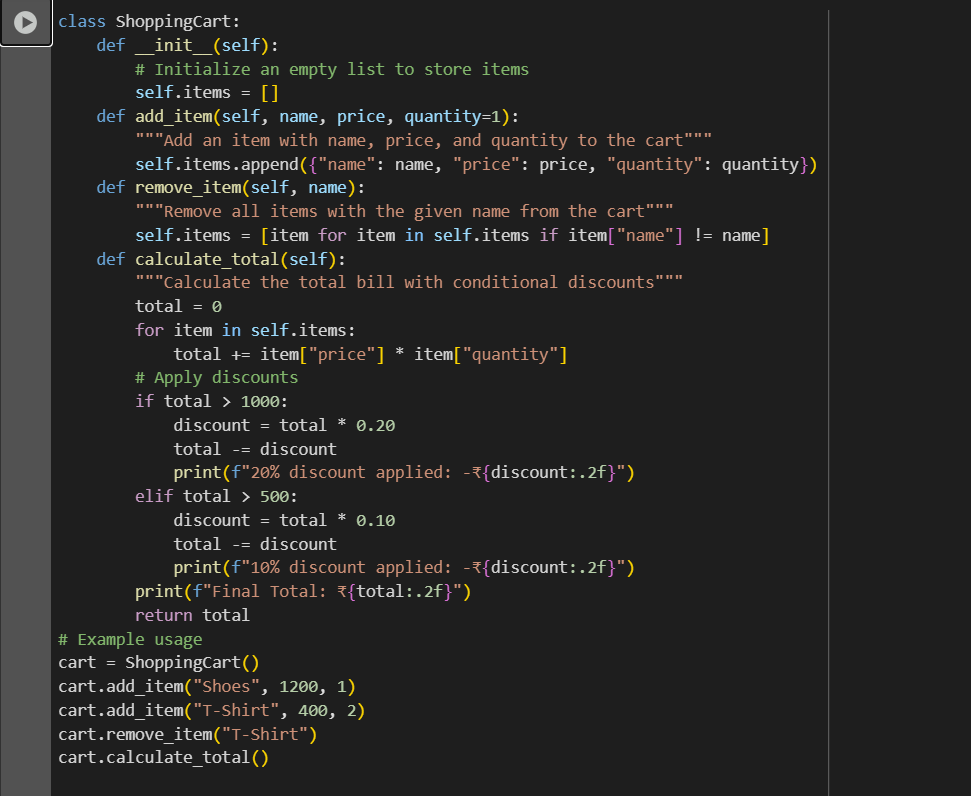
if students[i]["score"] > 75:: Inside the while loop, this is an if statement that checks a condition for the current student.students[i] accesses the dictionary at the current index i in the students list.["score"] accesses the value associated with the key "score" in that dictionary.The condition > 75 checks if the student's score is greater than 75.print(students[i]["name"]): If the condition in the if statement is true (i.e., the student's score is greater than 75), this line is executed.students[i]["name"] accesses the value associated with the key "name" in the current student's dictionary.print(...) displays the student's name to the console.

i += 1: This line increments the value of i by 1 after each iteration of the while loop. This is crucial for moving to the next student in the list and eventually terminating the loop when i is no longer less than len(students).

### Prompt-5:

Begin writing a class ShoppingCart with an empty items list. Prompt Copilot to generate methods to add\_item, remove\_item, and use a loop to calculate the total bill using conditional discounts

### Code:



### Output:

A black background with white text

AI-generated content may be incorrect.

### Explaination:

The ShoppingCart class is used to keep track of items a customer wants to buy.

1. When we create a shopping cart, it starts empty.
2. We can add items to the cart by giving their name, price, and quantity. Each item is stored in a list.
3. If we don’t want something anymore, we can remove it from the cart using its name.
4. To get the total bill, the program goes through all the items, multiplies price × quantity, and adds everything together.
5. After calculating the total, discounts are applied:
   * If the total is more than ₹1000, a 20% discount is given.
   * If the total is more than ₹500, but less than or equal to ₹1000, a 10% discount is given.
   * If the total is ₹500 or less, there is no discount.
6. Finally, the program prints the discount (if any) and the final total bill.

**Example:**  
If you add Shoes for ₹1200, the total is ₹1200. Since it’s above ₹1000, a 20% discount (₹240) is applied. So the final bill becomes ₹960.