**LAB ASSIGNMENT:6.4**

**NAME:CHADA SATHWIKA REDDY**

**HTNO:2403A51334**

**BTNO:14**

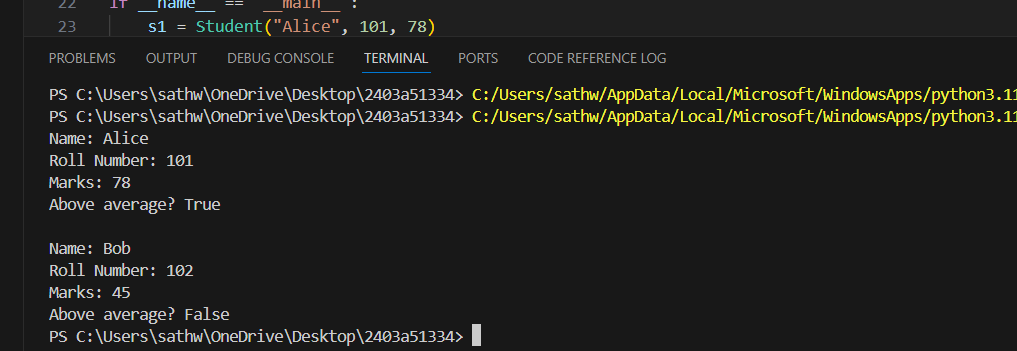
**TASK1:**

**PROMPT**:Create a student class with attributes such as name roll no marks and define methods that display details and checks the marks that are upto avg marks

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION**

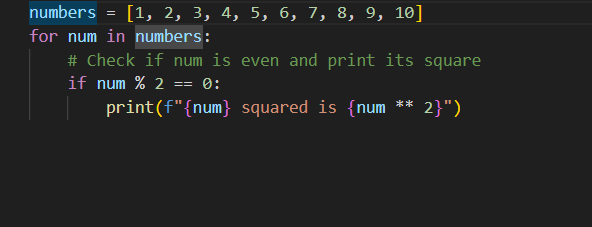
**Observations:**

1. **Class Structure & Initialization:**
   * The Student class is well-structured with three attributes: name, roll\_number, and marks.
   * These attributes are initialized through the constructor (\_\_init\_\_ method), which ensures every student object has these essential pieces of information.
2. **Display Method:**
   * The display\_details method provides a clear and readable way to print the student’s information.
   * This method makes it easy to output the student’s details without accessing each attribute separately.
   * It enhances code readability and reusability by encapsulating the display logic inside the class.
3. **Marks Comparison Method:**
   * The is\_above\_average method accepts an average value and checks if the student’s marks exceed this value.
   * This is a useful utility method that allows you to quickly check the student’s performance relative to a given benchmark.
   * The method returns a boolean (True or False), which makes it versatile for conditional checks or filtering.
4. **Code Simplicity & Readability:**
   * The code is simple and easy to understand, making it good for educational purposes or basic student record management.
   * Using formatted strings (f-strings) in display\_details improves output formatting and clarity.

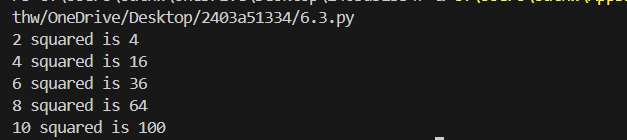
TASK2:

PROMPT:

CODE:



OUTPUT:



OBSERVATION:

**Observations:**

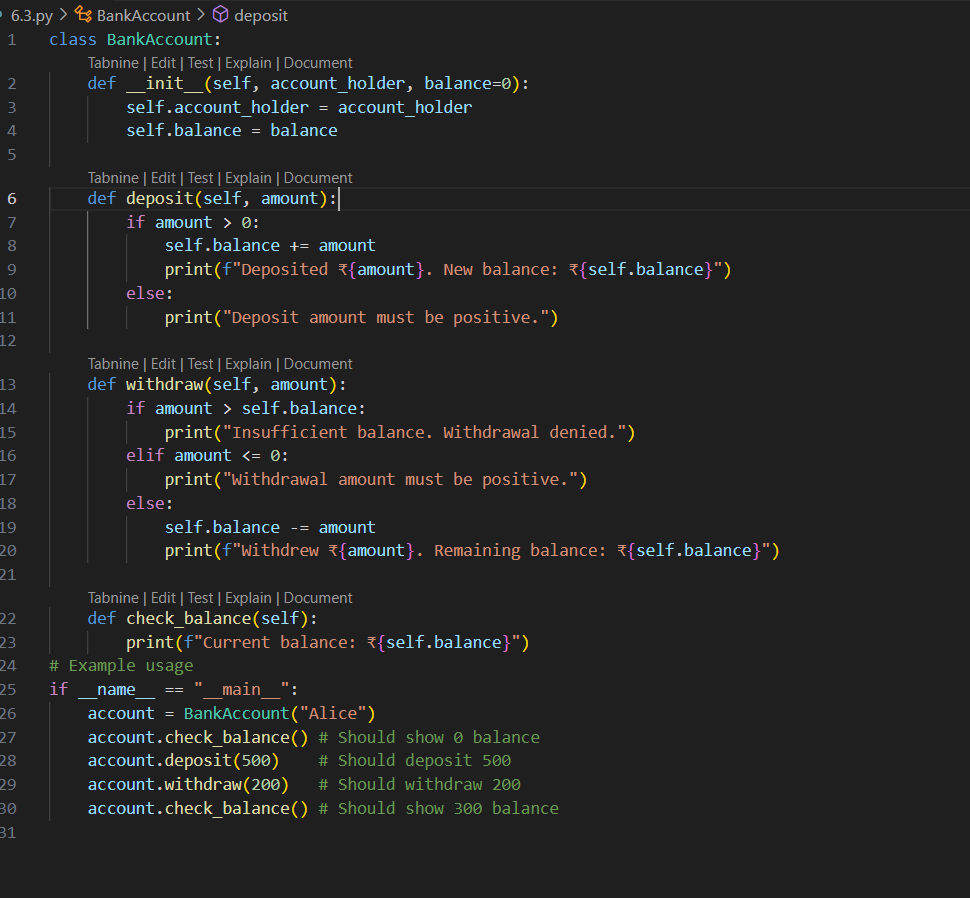
1. **Iteration over List:**
   * The loop iterates through each element in the numbers list one by one using a straightforward for loop.
   * This approach is simple and readable, making it clear what elements are being processed.
2. **Condition to Filter Even Numbers:**
   * The condition if num % 2 == 0: checks if the number is even by testing if the remainder when divided by 2 is zero.
   * This is an efficient and commonly used method to identify even numbers.
3. **Calculation and Output:**
   * For even numbers, the code calculates the square by using the exponent operator \*\* 2.
   * It immediately prints the squared value, providing instant output as the loop runs.
   * This inline approach keeps the code compact and easy to follow.

**TASK3:**

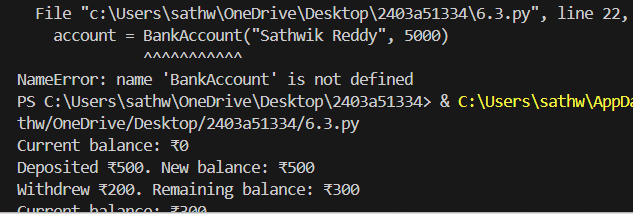
Write a python program Create a class called BankAccount with attributes account\_holder and balance. Use Copilot to complete methods for deposit(), withdraw(), and check for insufficient balance

**PROMPT:**

**CODE:**



**OUTPUT:**



**OBSERVATION**

The class correctly models a simple bank account with:

* account\_holder and balance attributes.
* Methods for deposit(), withdraw(), and check\_balance().

**Deposit Logic**

* Accepts only positive values.
* Updates the balance and confirms the transaction.
* Rejects zero or negative deposits with a clear message.

**Withdraw Logic**

* Checks for sufficient balance before allowing withdrawal.
* Prevents overdraft and informs the user.
* Rejects zero or negative withdrawals.

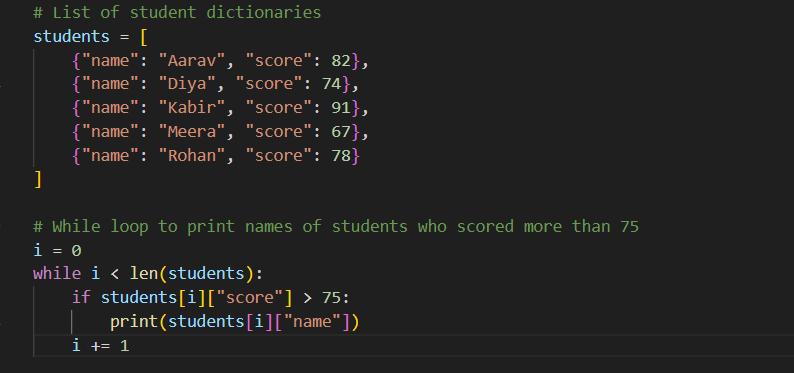
**Balance Check**

* Displays the current balance in a readable format.

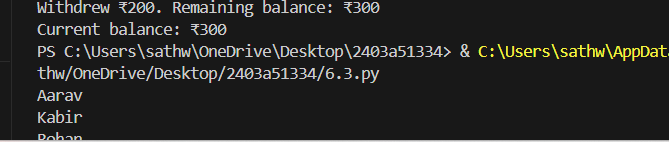
TASK4:

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

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION**

A list of dictionaries is defined, each representing a student with name and score**.**

* A while loop iterates through the list using an index variable i.

**Logic**

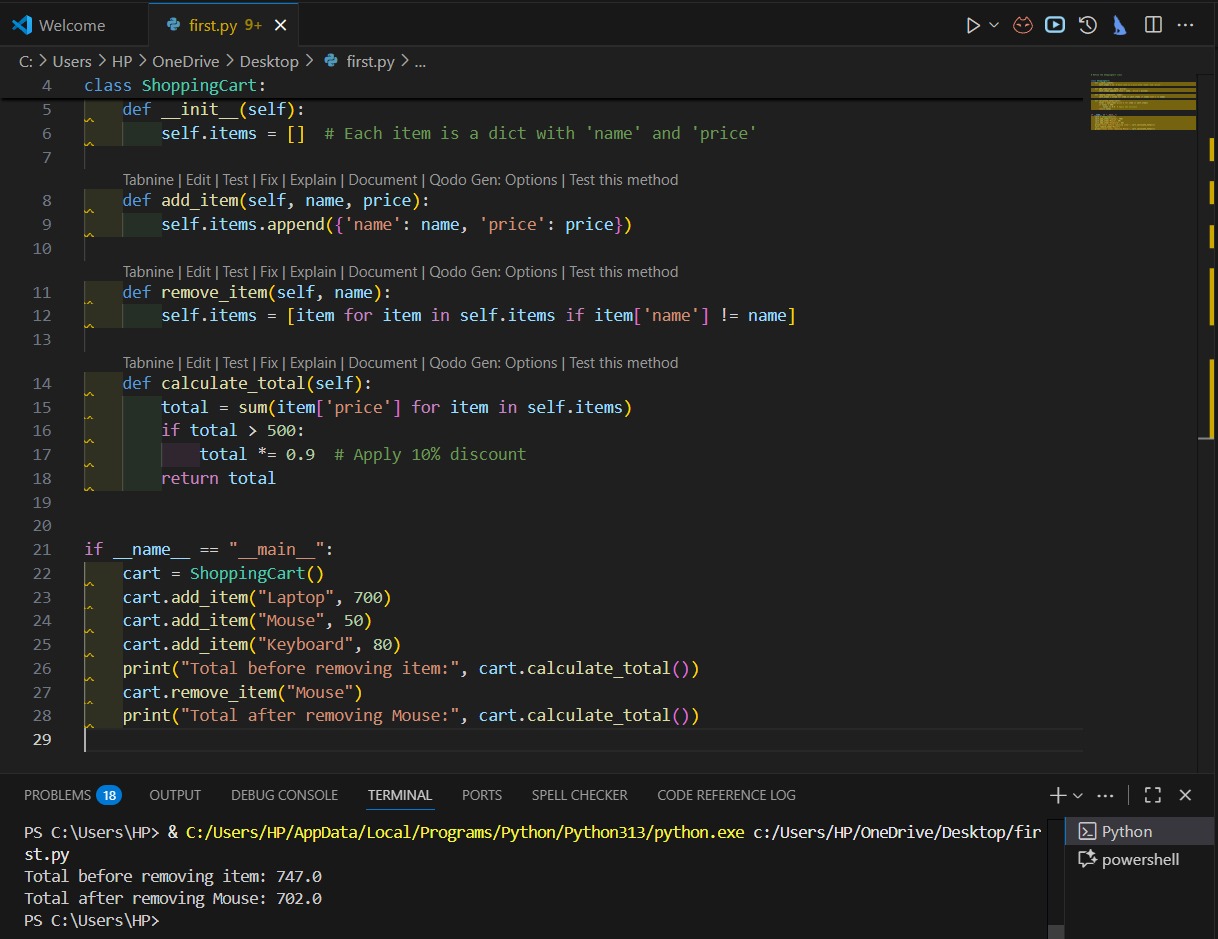
* For each student, the code checks if their score is greater than 75.
* If the condition is met, it prints the student's name.
* The loop increments i until all students are processed.

TASK5:

PROMPT:

Write a python 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 and OUTPUT:



Observation:

The ShoppingCart class initializes with an empty items list.

* Each item is stored as a dictionary with name and price keys.
* The design is modular and easy to extend (e.g., adding quantity, categories, etc.).

**add\_item(name, price)**

* Adds a new item to the cart.
* Provides immediate feedback confirming the addition.
* No duplicate check—multiple items with the same name are allowed.

**remove\_item(name)**

* Searches for the first item matching the name and removes it.
* If the item isn’t found, it prints a helpful message.
* Only removes one instance even if duplicates exist.

**calculate\_total()**

* Iterates through all items using a for loop.