SR UNIVERSITY

AI ASSIST CODING

Lab-6.4: *AI-Based Code Completion – Classes, Loops, and Conditionals*

ROLL NO:2503A51L13

NAME:BEGALA HASINI

BATCH:19

Lab Objectives:

• To explore AI-powered auto-completion features for core Python constructs.

• To analyze how AI suggests logic for class definitions, loops, and conditionals.

• To evaluate the completeness and correctness of code generated by AI assistants. **Lab Outcomes (LOs):**

After completing this lab, students will be able to:

• Use AI tools to generate and complete class definitions and methods.

• Understand and assess AI-suggested loops for iterative tasks.

• Generate conditional statements through prompt-driven suggestions.

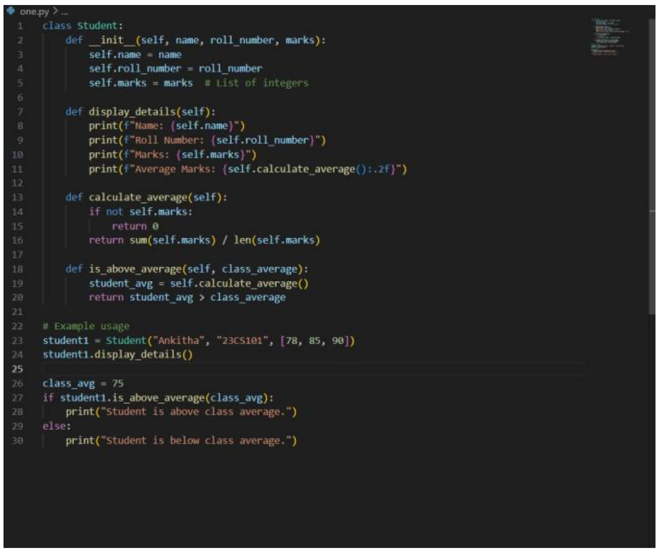
• Critically evaluate AI-assisted code for correctness and clarity

• TASK #1:

• 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.

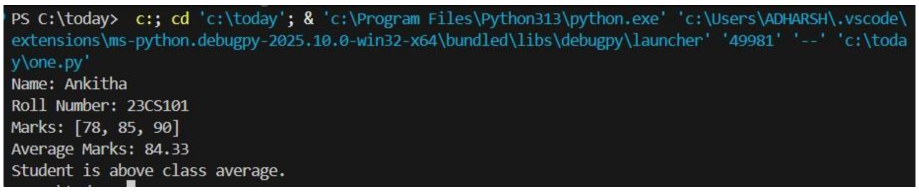
Code:
```
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks  # List of integers

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")
        print(f"Average Marks: {self.calculate_average():.2f}")

    def calculate_average(self):
        if not self.marks:
            return 0
        return sum(self.marks) / len(self.marks)

    def is_above_average(self, class_average):
        student_avg = self.calculate_average()
        return student_avg > class_average

# Example usage
student1 = Student("Ankitha", "23CS101", [78, 85, 90])
student1.display_details()

class_avg = 75
if student1.is_above_average(class_avg):
    print("Student is above class average.")
else:
    print("Student is below class average.")
```

Code Generated:



```
one.py > ...
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks  # List of integers

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")
        print(f"Average Marks: {self.calculate_average():.2f}")

    def calculate_average(self):
        if not self.marks:
            return 0
        return sum(self.marks) / len(self.marks)

    def is_above_average(self, class_average):
        student_avg = self.calculate_average()
        return student_avg > class_average

# Example usage
student1 = Student("Ankitha", "23CS101", [78, 85, 90])
student1.display_details()

class_avg = 75
if student1.is_above_average(class_avg):
    print("Student is above class average.")
else:
    print("Student is below class average.")
```

Output After executing Code:



```
PS C:\today>  c:; cd 'c:\today'; & 'c:\Program Files\Python313\python.exe' 'c:\Users\ADHARSH\.vscode\
extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '49981' '--' 'c:\toda
y\one.py'
Name: Ankitha
Roll Number: 23CS101
Marks: [78, 85, 90]
Average Marks: 84.33
Student is above class average.
```

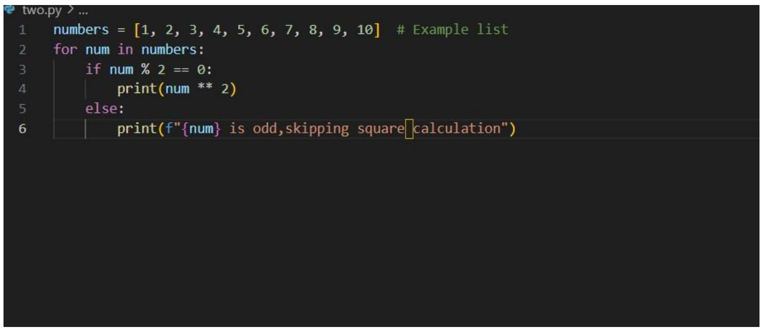
Observations:

- A Student class is created with attributes **name**, **roll number**, and **marks**.
- The display_details() method neatly prints the student's details.
- The is_above_average() method compares the student's marks with a given average and prints the result.
- User input is taken for **name, roll number, marks, and average** at runtime, making the program interactive.

TASK #2:

Prompt Used:

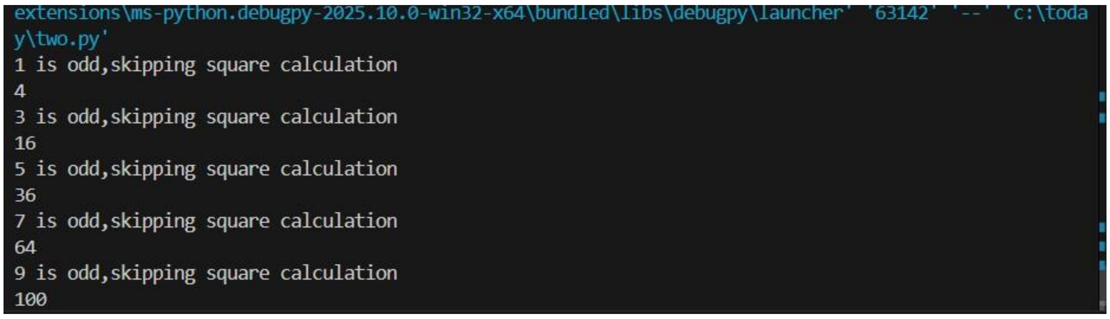
- Write the first two lines of a for loop to iterate through a list o f numbers. Suggest how to calculate and print the square of even numbers only.



```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  # Example list
for num in numbers:
    if num % 2 == 0:
        print(num ** 2)
    else:
        print(f"{num} is odd,skipping square calculation")
```

Code Generated:

Output After executing Code:



```
extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63142' '--' 'c:\toda
y\two.py'
1 is odd,skipping square calculation
4
3 is odd,skipping square calculation
16
5 is odd,skipping square calculation
36
7 is odd,skipping square calculation
64
9 is odd,skipping square calculation
100
```

Observations:

- The function Iterates through numbers.
- We have to give the Condition if num % 2 == 0 checks even numbers.
- It results in Prints their square using num ** 2.

TASK#3:

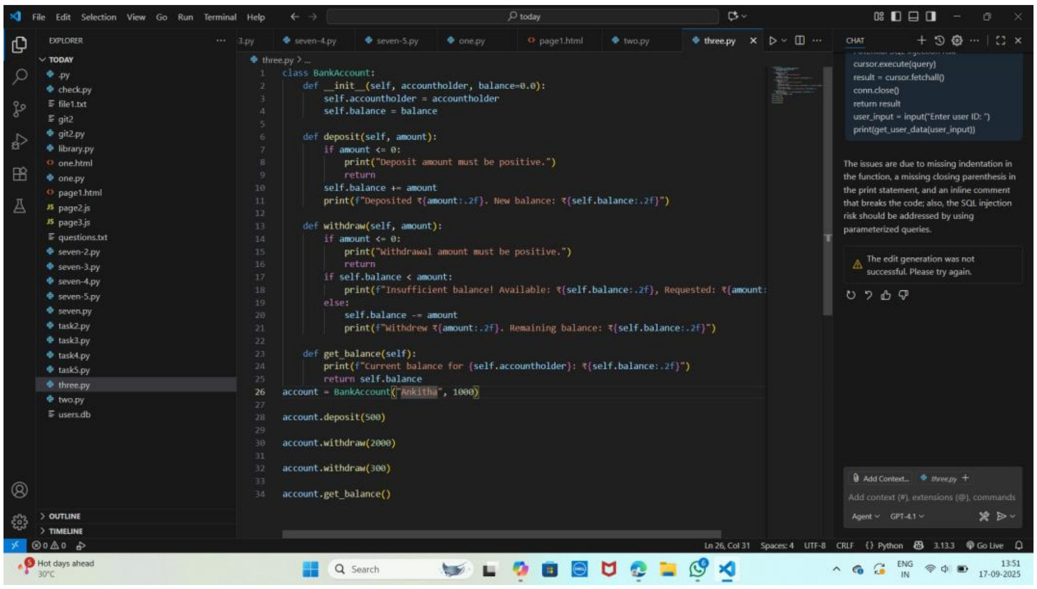
Prompt Used:

•Create a class called Bank Account with attributes accountholder and balance . Complete methods for deposit() ,withdraw() ,and check for insufficient balance.

Code: class BankAccount:

```
    def __init__(self, accountholder, balance=0.0):
        self.accountholder = accountholder
        self.balance = balance

    def deposit(self, amount):
        if amount <= 0:
            print("Deposit amount must be positive.")
            return
        self.balance += amount
        print(f"Deposited ₹{amount:.2f}. New balance: ₹{self.balance:.2f}")
    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal amount must be positive.")
            return
        if self.balance < amount:
            print(f"Insufficient balance! Available: ₹{self.balance:.2f}, Requested: ₹{amount:.2f}")
        else:
            self.balance -= amount
            print(f"Withdrew ₹{amount:.2f}. Remaining balance: ₹{self.balance:.2f}")

    def get_balance(self):
        print(f"Current balance for {self.accountholder}: ₹{self.balance:.2f}")
        return self.balance
account = BankAccount("Ankitha", 1000)

account.deposit(500)

account.withdraw(2000)

account.withdraw(300)

account.get_balance()
```

Code Generated:



```
class BankAccount:
    def __init__(self, accountholder, balance=0.0):
        self.accountholder = accountholder
        self.balance = balance

    def deposit(self, amount):
        if amount <= 0:
            print("Deposit amount must be positive.")
            return
        self.balance += amount
        print(f"Deposited ₹{amount:.2f}. New balance: ₹{self.balance:.2f}")

    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal amount must be positive.")
            return
        if self.balance < amount:
            print(f"Insufficient balance! Available: ₹{self.balance:.2f}, Requested: ₹{amount:
        else:
            self.balance -= amount
            print(f"Withdrew ₹{amount:.2f}. Remaining balance: ₹{self.balance:.2f}")

    def get_balance(self):
        print(f"Current balance for {self.accountholder}: ₹{self.balance:.2f}")
        return self.balance
account = BankAccount("Ankitha", 1000)

account.deposit(500)

account.withdraw(2000)

account.withdraw(300)

account.get_balance()
```

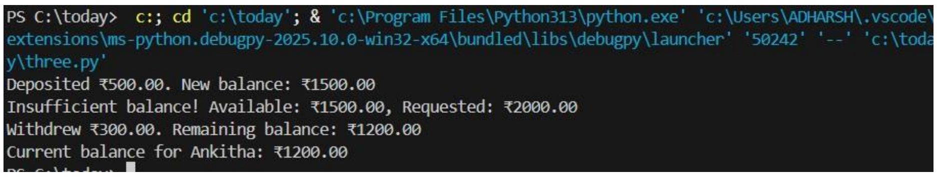
CHAT

```
cursor.execute(query)
result = cursor.fetchall()
conn.close()
return result
user_input = input("Enter user ID: ")
print(get_user_data(user_input))
```

The issues are due to missing indentation in the function, a missing closing parenthesis in the print statement, and an inline comment that breaks the code; also, the SQL injection risk should be addressed by using parameterized queries.

⚠ The edit generation was not successful. Please try again.

Output After executing Code:



```
PS C:\today>  c:; cd 'c:\today'; & 'c:\Program Files\Python313\python.exe' 'c:\Users\ADHARSH\.vscode\
extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '50242' '--' 'c:\toda
y\three.py'
Deposited ₹500.00. New balance: ₹1500.00
Insufficient balance! Available: ₹1500.00, Requested: ₹2000.00
Withdrew ₹300.00. Remaining balance: ₹1200.00
Current balance for Ankitha: ₹1200.00
PS C:\today>
```

Observation:

- We used function deposit(): increases balance. we can able to use the function
- withdraw(): prevents overdrawing using if conditions . its results in check_balance():
- shows current balance**.**

TASK#4:

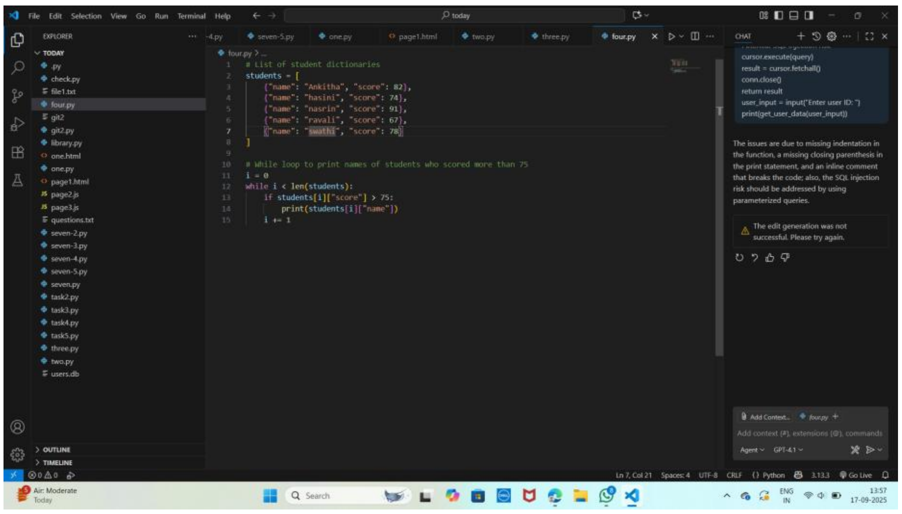
Prompt Used:

- 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:

```
students = [

  {"name": "Ankitha", "score": 82},

  {"name": "hasini", "score": 74},

  {"name": "nasrin", "score": 91},

  {"name": "ravali", "score": 67},

  {"name": "swathi", "score": 78}

]



# While loop to print names of students who scored more than 75

i = 0

while i < len(students):

  if students[i]["score"] > 75:

    print(students[i]["name"])

  i += 1
```

Code Generated:



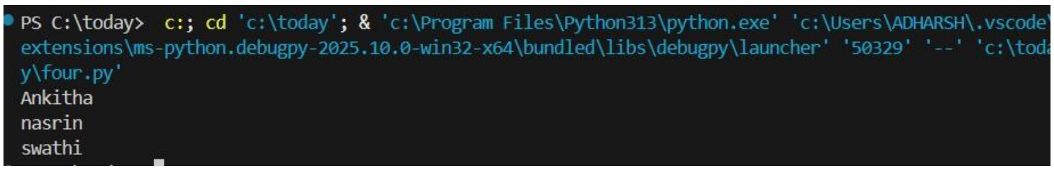
```python
# List of student dictionaries
students = [
    {"name": "Ankitha", "score": 82},
    {"name": "hasini", "score": 74},
    {"name": "nasrin", "score": 91},
    {"name": "ravali", "score": 67},
    {"name": "swathi", "score": 78}
]

# While loop to print names of students who scored more than 75
i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

```
cursor.execute(query)
result = cursor.fetchall()
conn.close()
return result
user_input = input("Enter user ID: ")
print(get_user_data(user_input))
```

The issues are due to missing indentation in the function, a missing closing parenthesis in the print statement, and an inline comment that breaks the code; also, the SQL injection risk should be addressed by using parameterized queries.

The edit generation was not successful. Please try again.

Output After executing Code:



```
PS C:\today>  c:; cd 'c:\today'; & 'c:\Program Files\Python313\python.exe' 'c:\Users\ADHARSH\.vscode\
extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '50329' '--' 'c:\toda
y\four.py'
Ankitha
nasrin
swathi
```

Observations:

- We Uses while loop with counter i.
- The loop Checks if score > 75.

It will Prints qualifying students.

TASK#5: PROMPT:

- Begin writing a class Shopping Cart 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:

```
    lass ShoppingCart:     def

__init__(self):          self.items =

[]   def add_item(self, name,

price):

          self.items.append((name, price))

    print(f"added {name} to the cart") def

remove_item(self, name):

          initial_len = len(self.items)        self.items = [item for item

     in  self.items  if  item[0]  !=  name]              if  len(self.items)  <

     initial_len:

             print(f"removed shoes from the cart{name}")

          else:

             print(f"{name} not found in the cart")   def

     calculate_total(self, discount=0):

          total = sum(price for _, price in self.items)

     if discount > 0:          total -= total *

     (discount / 100)        return total
```

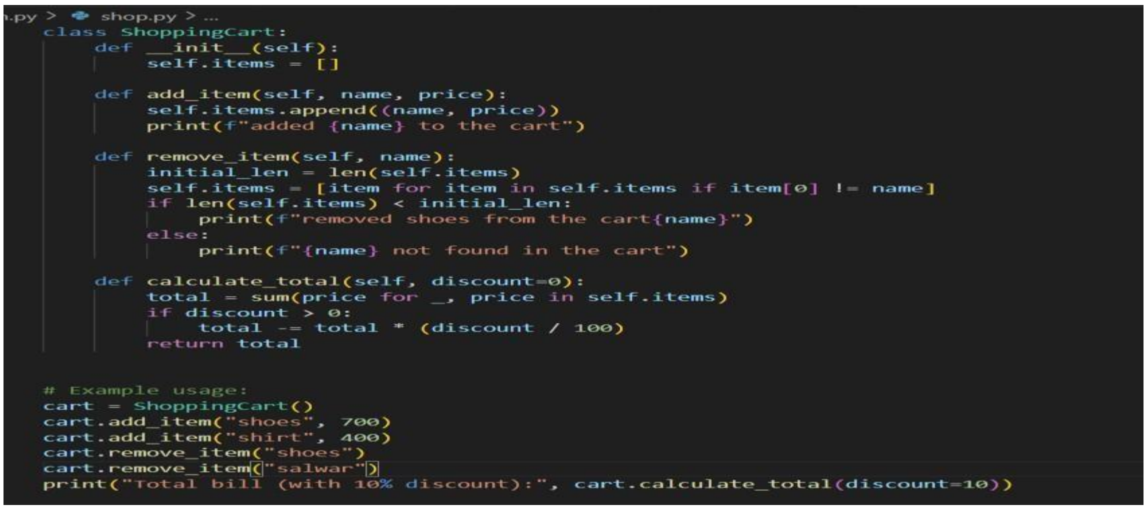
# Example usage: cart = ShoppingCart() cart.add_item("shoes", 700)

cart.add_item("shirt", 400) cart.remove_item("shoes")

cart.remove_item("salwar") print("Total bill (with 10% discount):",

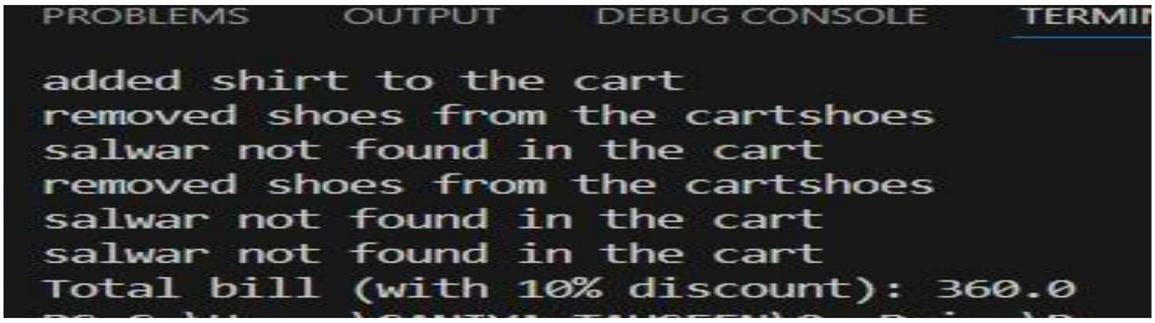
cart.calculate_total(discount=10))

Code Generated:



```python
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, name, price):
        self.items.append((name, price))
        print(f"added {name} to the cart")

    def remove_item(self, name):
        initial_len = len(self.items)
        self.items = [item for item in self.items if item[0] != name]
        if len(self.items) < initial_len:
            print(f"removed shoes from the cart{name}")
        else:
            print(f"{name} not found in the cart")

    def calculate_total(self, discount=0):
        total = sum(price for _, price in self.items)
        if discount > 0:
            total -= total * (discount / 100)
        return total

# Example usage:
cart = ShoppingCart()
cart.add_item("shoes", 700)
cart.add_item("shirt", 400)
cart.remove_item("shoes")
cart.remove_item("salwar")
print("Total bill (with 10% discount):", cart.calculate_total(discount=10))
```

Output After executing Code:



```
added shirt to the cart
removed shoes from the cartshoes
salwar not found in the cart
removed shoes from the cartshoes
salwar not found in the cart
salwar not found in the cart
Total bill (with 10% discount): 360.0
```

Observations:

- If we want to add item use function-add_item(): adds item to cart.
- If we want to remove item use function remove_item(): removes by name.
- If we want to calculate the total use function calculate_total(): loops through cart, applies discounts with if-elif.