AI-Assisted Coding

Assignment-6.3

Name: C . Vaishnav Reddy

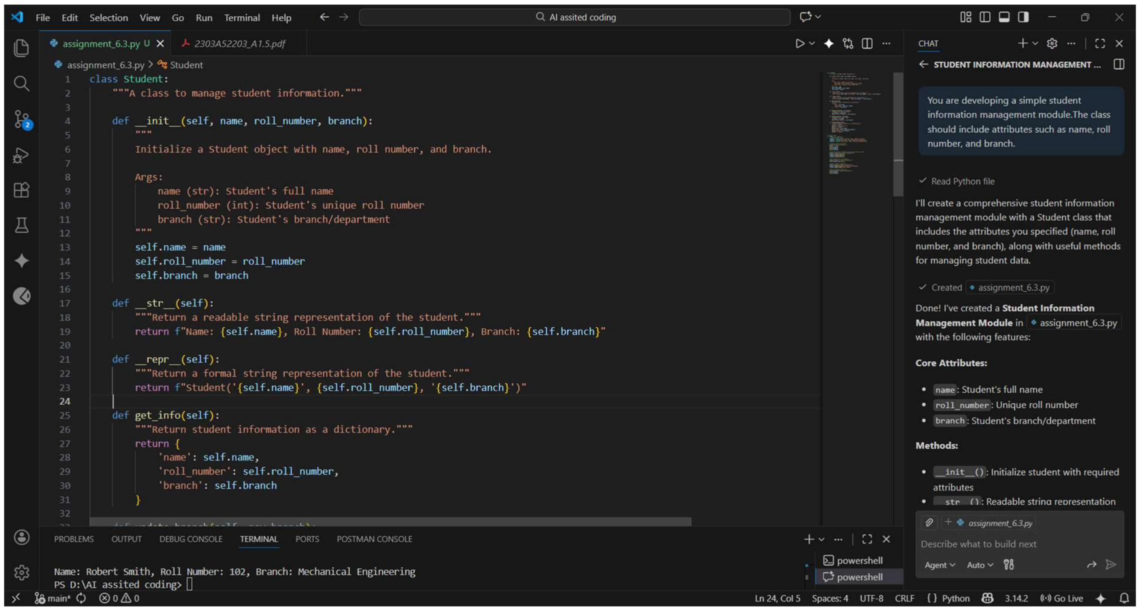
Rollno: 2303A52203

Batch: 45

Task1: Classes (Student Class)

=>Use an AI tool (GitHub Copilot) to complete a Student class.

=>The class should include attributes such as name, roll number, and branch.


File Edit Selection View Go Run Terminal Help
AI assited coding
assignment_6.3.py U
2303A52203_A1.5.pdf
assignment_6.3.py > Student
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
powershell
main* 0 0
Ln 24, Col 5 Spaces: 4 UTF-8 CRLF {} Python 3.14.2 Go Live
CHAT
STUDENT INFORMATION MANAGEMENT ...
Describe what to build next
Agent Auto

```python
class Student:
    """A class to manage student information."""

    def __init__(self, name, roll_number, branch):
        """
        Initialize a Student object with name, roll number, and branch.

        Args:
            name (str): Student's full name
            roll_number (int): Student's unique roll number
            branch (str): Student's branch/department
        """
        self.name = name
        self.roll_number = roll_number
        self.branch = branch

    def __str__(self):
        """Return a readable string representation of the student."""
        return f"Name: {self.name}, Roll Number: {self.roll_number}, Branch: {self.branch}"

    def __repr__(self):
        """Return a formal string representation of the student."""
        return f"Student('{self.name}', {self.roll_number}, '{self.branch}')"

    def get_info(self):
        """Return student information as a dictionary."""
        return {
            'name': self.name,
            'roll_number': self.roll_number,
            'branch': self.branch
        }
```

```
Name: Robert Smith, Roll Number: 102, Branch: Mechanical Engineering
PS D:\AI assited coding>
```

You are developing a simple student information management module.The class should include attributes such as name, roll number, and branch.

✓ Read Python file

I'll create a comprehensive student information management module with a Student class that includes the attributes you specified (name, roll number, and branch), along with useful methods for managing student data.

✓ Created assignment_6.3.py

Done! I've created a **Student Information Management Module** in assignment_6.3.py with the following features:

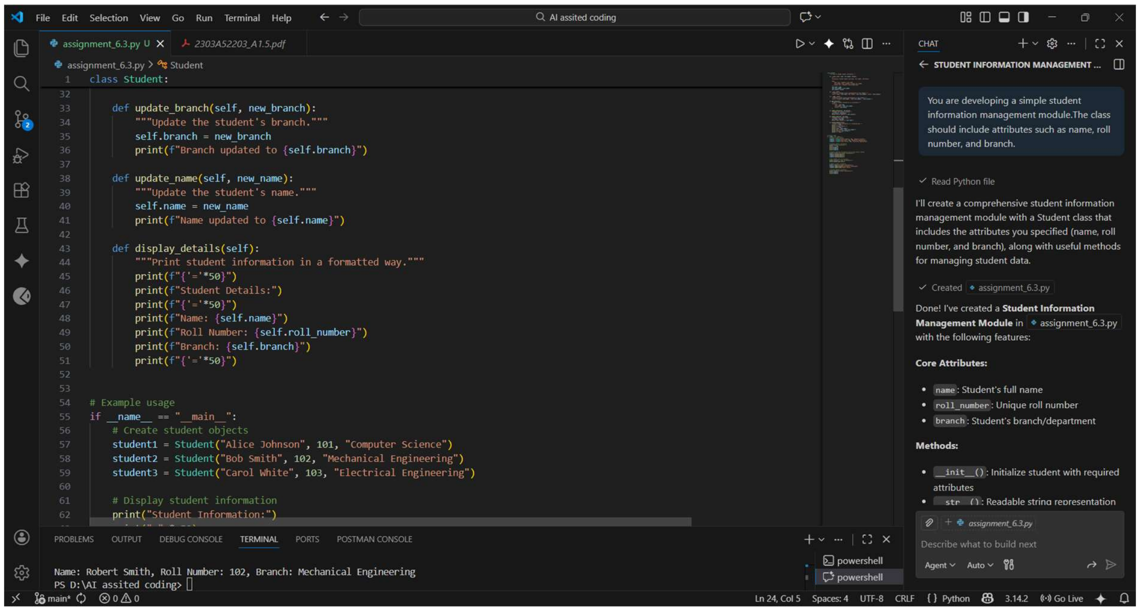
Core Attributes:

- `name`: Student's full name
- `roll_number`: Unique roll number
- `branch`: Student's branch/department

Methods:

- `__init__()`: Initialize student with required attributes
- `__str__()`: Readable string representation

=>Add a method display_details() to print student information.


File Edit Selection View Go Run Terminal Help
AI assited coding
assignment_6.3.py U
2303A52203_A1.5.pdf
assignment_6.3.py > Student
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
powershell
main*
Ln 24, Col 5 Spaces: 4 UTF-8 CRLF {} Python 3.14.2 Go Live
CHAT
Describe what to build next
Agent Auto

```python
class Student:

    def update_branch(self, new_branch):
        """Update the student's branch."""
        self.branch = new_branch
        print(f"Branch updated to {self.branch}")

    def update_name(self, new_name):
        """Update the student's name."""
        self.name = new_name
        print(f"Name updated to {self.name}")

    def display_details(self):
        """Print student information in a formatted way."""
        print(f"{'='*50}")
        print(f"Student Details:")
        print(f"{'='*50}")
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Branch: {self.branch}")
        print(f"{'='*50}")


# Example usage
if __name__ == "__main__":
    # Create student objects
    student1 = Student("Alice Johnson", 101, "Computer Science")
    student2 = Student("Bob Smith", 102, "Mechanical Engineering")
    student3 = Student("Carol White", 103, "Electrical Engineering")

    # Display student information
    print("Student Information:")
```

```
Name: Robert Smith, Roll Number: 102, Branch: Mechanical Engineering
PS D:\AI assited coding> 
```

STUDENT INFORMATION MANAGEMENT ...

You are developing a simple student information management module.The class should include attributes such as name, roll number, and branch.

✓ Read Python file

I'll create a comprehensive student information management module with a Student class that includes the attributes you specified (name, roll number, and branch), along with useful methods for managing student data.

✓ Created assignment_6.3.py

Done! I've created a **Student Information Management Module** in assignment_6.3.py with the following features:

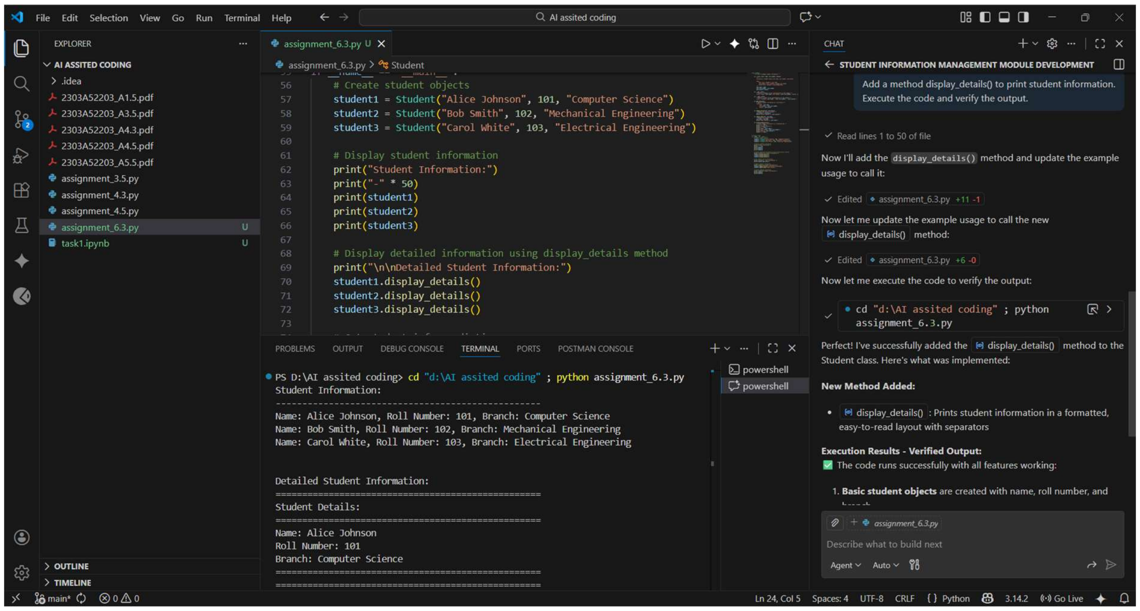
Core Attributes:

- `name`: Student's full name
- `roll_number`: Unique roll number
- `branch`: Student's branch/department

Methods:

- `__init__()`: Initialize student with required attributes
- `__str__()`: Readable string representation

=>Execute the code and verify the output.



```python
    # Create student objects
    student1 = Student("Alice Johnson", 101, "Computer Science")
    student2 = Student("Bob Smith", 102, "Mechanical Engineering")
    student3 = Student("Carol White", 103, "Electrical Engineering")

    # Display student information
    print("Student Information:")
    print("-" * 50)
    print(student1)
    print(student2)
    print(student3)

    # Display detailed information using display_details method
    print("\n\nDetailed Student Information:")
    student1.display_details()
    student2.display_details()
    student3.display_details()
```

```
PS D:\AI assited coding> cd "d:\AI assited coding" ; python assignment_6.3.py
Student Information:
--------------------------------------------------
Name: Alice Johnson, Roll Number: 101, Branch: Computer Science
Name: Bob Smith, Roll Number: 102, Branch: Mechanical Engineering
Name: Carol White, Roll Number: 103, Branch: Electrical Engineering


Detailed Student Information:
==================================================
Student Details:
==================================================
Name: Alice Johnson
Roll Number: 101
Branch: Computer Science
==================================================
==================================================
```

STUDENT INFORMATION MANAGEMENT MODULE DEVELOPMENT

Add a method display_details() to print student information. Execute the code and verify the output.

✓ Read lines 1 to 50 of file

Now I'll add the `display_details()` method and update the example usage to call it:

✓ Edited assignment_6.3.py +11 -1

Now let me update the example usage to call the new display_details() method:

✓ Edited assignment_6.3.py +6 -0

Now let me execute the code to verify the output:

```
cd "d:\AI assited coding" ; python assignment_6.3.py
```

Perfect! I've successfully added the display_details() method to the Student class. Here's what was implemented:

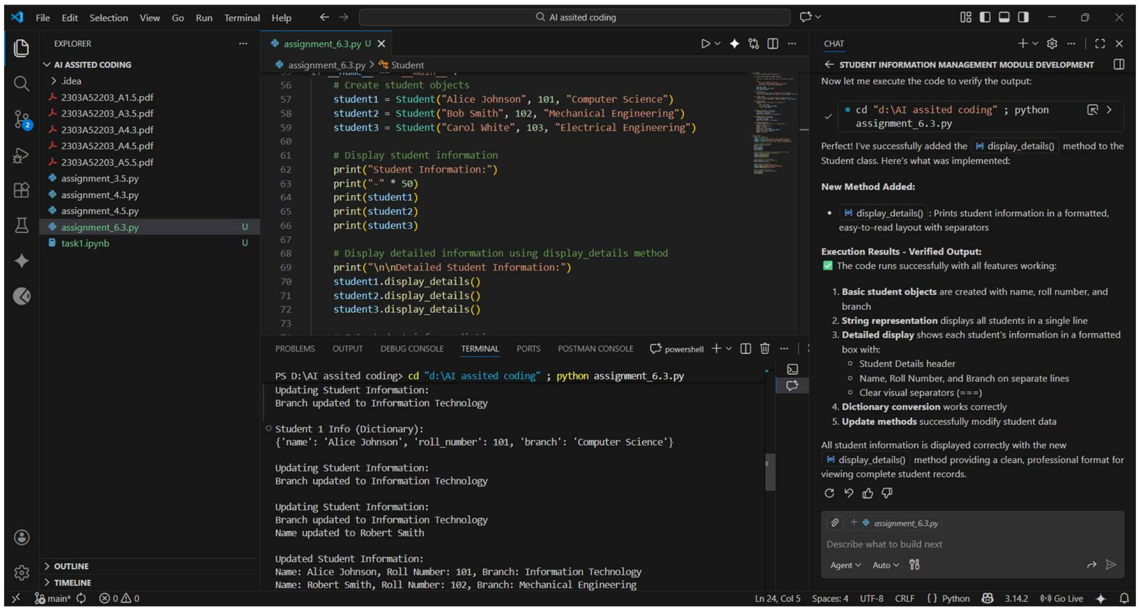
New Method Added:

- display_details() : Prints student information in a formatted, easy-to-read layout with separators

Execution Results - Verified Output:
✅ The code runs successfully with all features working:

1. **Basic student objects** are created with name, roll number, and

=>Analyze the code generated by the AI tool for correctness and clarity.



```python
    # Create student objects
    student1 = Student("Alice Johnson", 101, "Computer Science")
    student2 = Student("Bob Smith", 102, "Mechanical Engineering")
    student3 = Student("Carol White", 103, "Electrical Engineering")

    # Display student information
    print("Student Information:")
    print("-" * 50)
    print(student1)
    print(student2)
    print(student3)

    # Display detailed information using display_details method
    print("\n\nDetailed Student Information:")
    student1.display_details()
    student2.display_details()
    student3.display_details()

```

```
PS D:\AI assited coding> cd "d:\AI assited coding" ; python assignment_6.3.py
Updating Student Information:
Branch updated to Information Technology

Student 1 Info (Dictionary):
{'name': 'Alice Johnson', 'roll_number': 101, 'branch': 'Computer Science'}

Updating Student Information:
Branch updated to Information Technology

Updating Student Information:
Branch updated to Information Technology
Name updated to Robert Smith

Updated Student Information:
Name: Alice Johnson, Roll Number: 101, Branch: Information Technology
Name: Robert Smith, Roll Number: 102, Branch: Mechanical Engineering
```

STUDENT INFORMATION MANAGEMENT MODULE DEVELOPMENT

Now let me execute the code to verify the output:

```
cd "d:\AI assited coding" ; python assignment_6.3.py
```

Perfect! I've successfully added the display_details() method to the Student class. Here's what was implemented:

New Method Added:

- display_details() : Prints student information in a formatted, easy-to-read layout with separators

Execution Results - Verified Output:
✅ The code runs successfully with all features working:

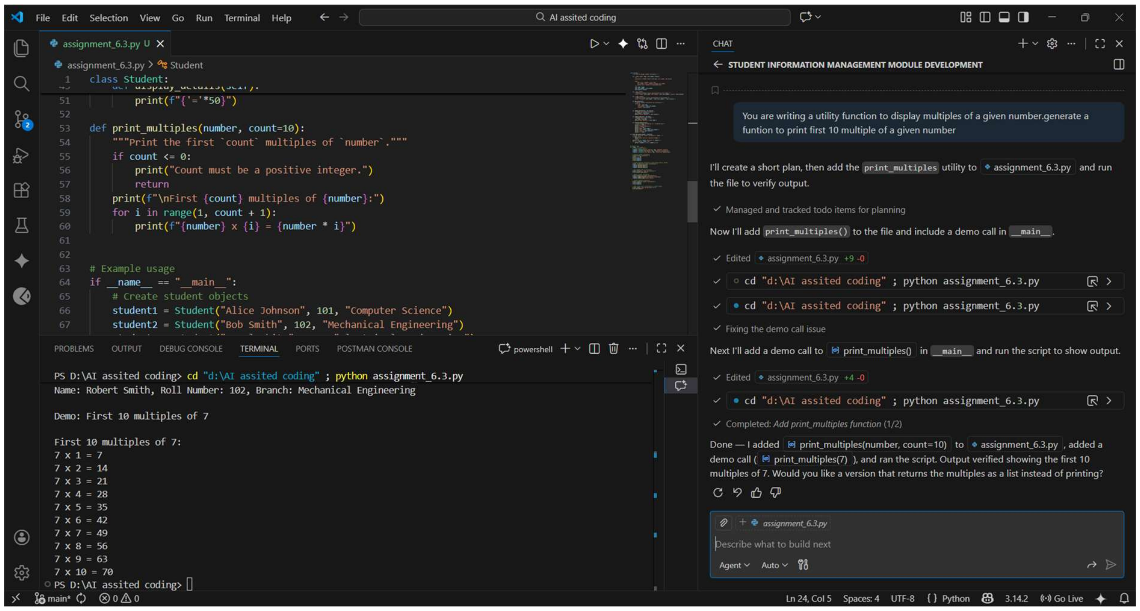
1. **Basic student objects** are created with name, roll number, and branch
2. **String representation** displays all students in a single line
3. **Detailed display** shows each student's information in a formatted box with:
   - Student Details header
   - Name, Roll Number, and Branch on separate lines
   - Clear visual separators (===)
4. **Dictionary conversion** works correctly
5. **Update methods** successfully modify student data

All student information is displayed correctly with the new display_details() method providing a clean, professional format for viewing complete student records.

Describe what to build next

Task Description #2: Loops (Multiples of a Number)

=>generate a funtion to print first 10 multiple of a given number


File Edit Selection View Go Run Terminal Help
AI assited coding
assignment_6.3.py U
assignment_6.3.py > Student
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE powershell
CHAT
main* 0 0
Ln 24, Col 5 Spaces: 4 UTF-8 {} Python 3.14.2 Go Live

```python
class Student:
            print(f"{'='*50}")

def print_multiples(number, count=10):
    """Print the first `count` multiples of `number`."""
    if count <= 0:
        print("Count must be a positive integer.")
        return
    print(f"\nFirst {count} multiples of {number}:")
    for i in range(1, count + 1):
        print(f"{number} x {i} = {number * i}")


# Example usage
if __name__ == "__main__":
    # Create student objects
    student1 = Student("Alice Johnson", 101, "Computer Science")
    student2 = Student("Bob Smith", 102, "Mechanical Engineering")
```

```
PS D:\AI assited coding> cd "d:\AI assited coding" ; python assignment_6.3.py
Name: Robert Smith, Roll Number: 102, Branch: Mechanical Engineering

Demo: First 10 multiples of 7

First 10 multiples of 7:
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
PS D:\AI assited coding> 
```

STUDENT INFORMATION MANAGEMENT MODULE DEVELOPMENT

You are writing a utility function to display multiples of a given number.generate a funtion to print first 10 multiple of a given number

I'll create a short plan, then add the `print_multiples` utility to assignment_6.3.py and run the file to verify output.

✓ Managed and tracked todo items for planning

Now I'll add `print_multiples()` to the file and include a demo call in `__main__`.

✓ Edited assignment_6.3.py +9 -0

```
cd "d:\AI assited coding" ; python assignment_6.3.py
cd "d:\AI assited coding" ; python assignment_6.3.py
```

✓ Fixing the demo call issue

Next I'll add a demo call to print_multiples() in `__main__` and run the script to show output.

✓ Edited assignment_6.3.py +4 -0

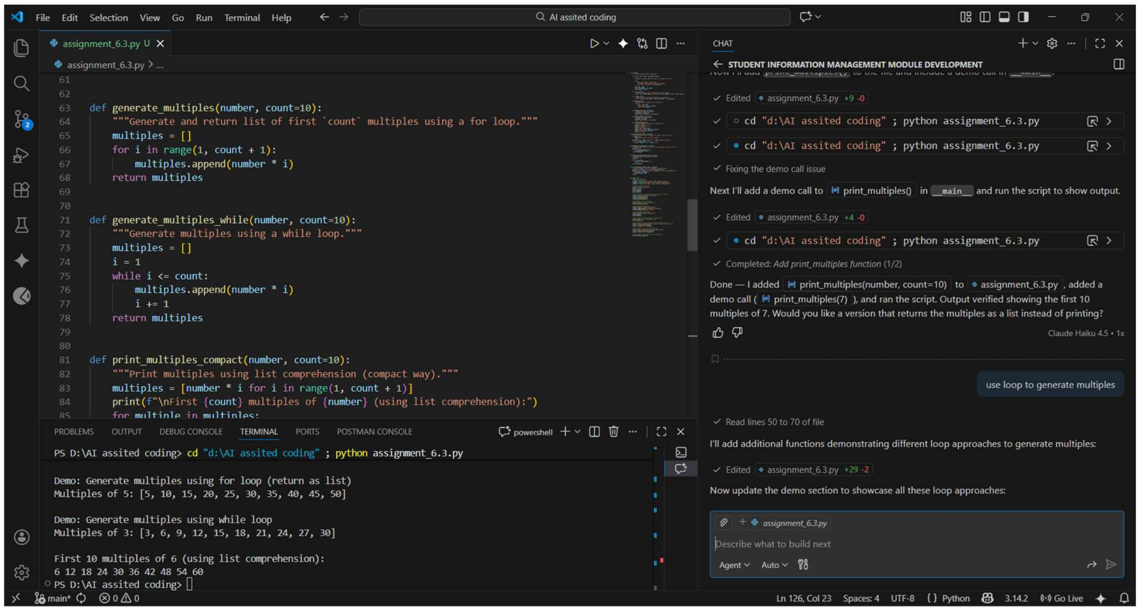
```
cd "d:\AI assited coding" ; python assignment_6.3.py
```

✓ Completed: *Add print_multiples function* (1/2)

Done — I added print_multiples(number, count=10) to assignment_6.3.py, added a demo call (print_multiples(7)), and ran the script. Output verified showing the first 10 multiples of 7. Would you like a version that returns the multiples as a list instead of printing?

Describe what to build next

=>Ask the AI to generate the same functionality using another controlled looping structure (e.g.,while instead of for).



```python
def generate_multiples(number, count=10):
    """Generate and return list of first `count` multiples using a for loop."""
    multiples = []
    for i in range(1, count + 1):
        multiples.append(number * i)
    return multiples


def generate_multiples_while(number, count=10):
    """Generate multiples using a while loop."""
    multiples = []
    i = 1
    while i <= count:
        multiples.append(number * i)
        i += 1
    return multiples


def print_multiples_compact(number, count=10):
    """Print multiples using list comprehension (compact way)."""
    multiples = [number * i for i in range(1, count + 1)]
    print(f"\nFirst {count} multiples of {number} (using list comprehension):")
    for multiple in multiples:
```

```
PS D:\AI assited coding> cd "d:\AI assited coding" ; python assignment_6.3.py

Demo: Generate multiples using for loop (return as list)
Multiples of 5: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

Demo: Generate multiples using while loop
Multiples of 3: [3, 6, 9, 12, 15, 18, 21, 24, 27, 30]

First 10 multiples of 6 (using list comprehension):
6 12 18 24 30 36 42 48 54 60
PS D:\AI assited coding>
```

CHAT

STUDENT INFORMATION MANAGEMENT MODULE DEVELOPMENT

✓ Edited assignment_6.3.py +9 -0

✓ cd "d:\AI assited coding" ; python assignment_6.3.py

✓ cd "d:\AI assited coding" ; python assignment_6.3.py

✓ Fixing the demo call issue

Next I'll add a demo call to print_multiples() in `__main__` and run the script to show output.

✓ Edited assignment_6.3.py +4 -0

✓ cd "d:\AI assited coding" ; python assignment_6.3.py

✓ Completed: *Add print_multiples function* (1/2)

Done — I added print_multiples(number, count=10) to assignment_6.3.py, added a demo call (print_multiples(7)), and ran the script. Output verified showing the first 10 multiples of 7. Would you like a version that returns the multiples as a list instead of printing?

Claude Haiku 4.5 • 1x

use loop to generate multiples

✓ Read lines 50 to 70 of file

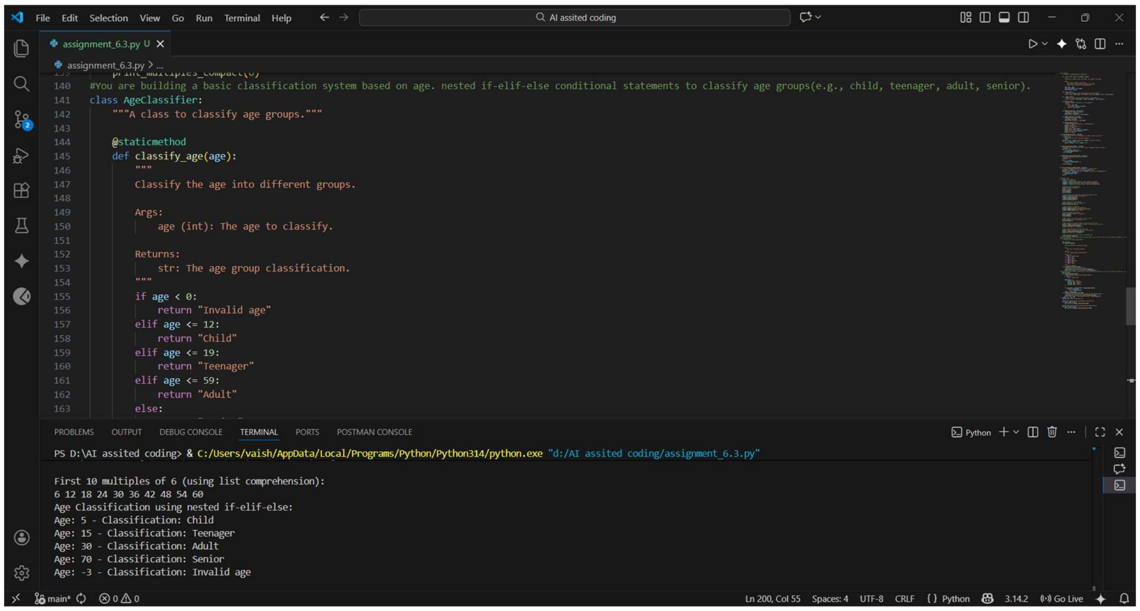
I'll add additional functions demonstrating different loop approaches to generate multiples:

✓ Edited assignment_6.3.py +29 -2

Now update the demo section to showcase all these loop approaches:

Task Description #3: Conditional Statements (Age Classification)

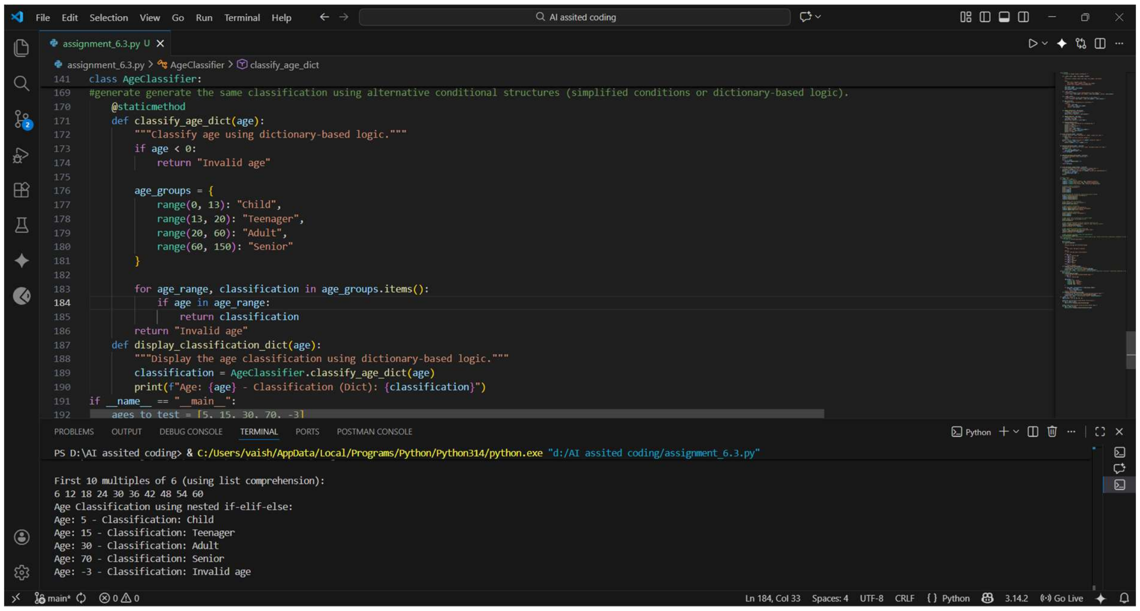
You are building a basic classification system based on age. nested if-elif-else conditional statements to classify age groups(e.g., child, teenager, adult, senior).



```python
#You are building a basic classification system based on age. nested if-elif-else conditional statements to classify age groups(e.g., child, teenager, adult, senior).
class AgeClassifier:
    """A class to classify age groups."""

    @staticmethod
    def classify_age(age):
        """
        Classify the age into different groups.

        Args:
            age (int): The age to classify.

        Returns:
            str: The age group classification.
        """
        if age < 0:
            return "Invalid age"
        elif age <= 12:
            return "Child"
        elif age <= 19:
            return "Teenager"
        elif age <= 59:
            return "Adult"
        else:
```

```
PS D:\AI assited coding> & C:/Users/vaish/AppData/Local/Programs/Python/Python314/python.exe "d:/AI assited coding/assignment_6.3.py"

First 10 multiples of 6 (using list comprehension):
6 12 18 24 30 36 42 48 54 60
Age Classification using nested if-elif-else:
Age: 5 - Classification: Child
Age: 15 - Classification: Teenager
Age: 30 - Classification: Adult
Age: 70 - Classification: Senior
Age: -3 - Classification: Invalid age
```

#generate generate the same classification using alternative conditional structures (simplified conditions or dictionary-based logic).

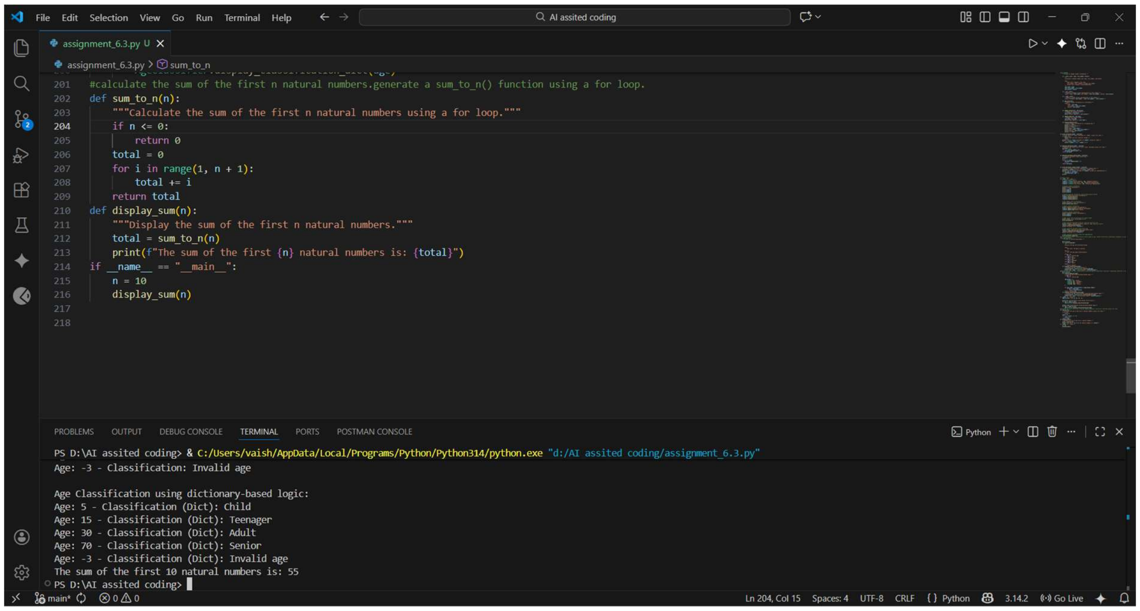


```python
class AgeClassifier:
#generate generate the same classification using alternative conditional structures (simplified conditions or dictionary-based logic).
    @staticmethod
    def classify_age_dict(age):
        """Classify age using dictionary-based logic."""
        if age < 0:
            return "Invalid age"

        age_groups = {
            range(0, 13): "Child",
            range(13, 20): "Teenager",
            range(20, 60): "Adult",
            range(60, 150): "Senior"
        }

        for age_range, classification in age_groups.items():
            if age in age_range:
                return classification
        return "Invalid age"
    def display_classification_dict(age):
        """Display the age classification using dictionary-based logic."""
        classification = AgeClassifier.classify_age_dict(age)
        print(f"Age: {age} - Classification (Dict): {classification}")
if __name__ == "__main__":
    ages_to_test = [5, 15, 30, 70, -3]
```

```
PS D:\AI assited coding> & C:/Users/vaish/AppData/Local/Programs/Python/Python314/python.exe "d:/AI assited coding/assignment_6.3.py"

First 10 multiples of 6 (using list comprehension):
6 12 18 24 30 36 42 48 54 60
Age Classification using nested if-elif-else:
Age: 5 - Classification: Child
Age: 15 - Classification: Teenager
Age: 30 - Classification: Adult
Age: 70 - Classification: Senior
Age: -3 - Classification: Invalid age
```

Task Description #4: For and While Loops (Sum of First n Numbers)

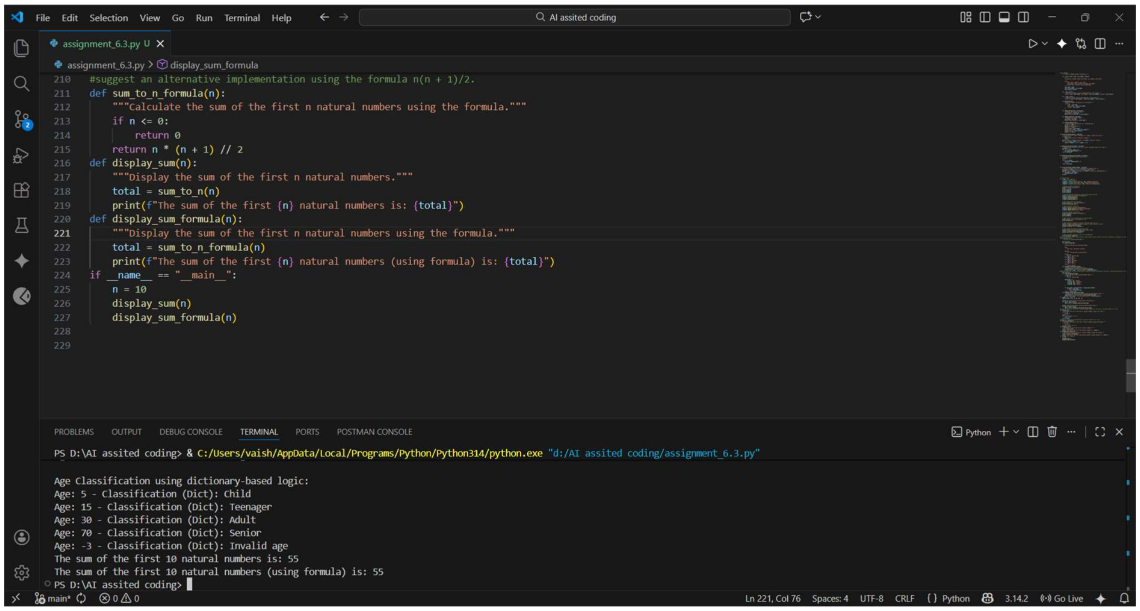
# generate a sum_to_n() function using a for loop.



```python
#calculate the sum of the first n natural numbers.generate a sum_to_n() function using a for loop.
def sum_to_n(n):
    """Calculate the sum of the first n natural numbers using a for loop."""
    if n <= 0:
        return 0
    total = 0
    for i in range(1, n + 1):
        total += i
    return total
def display_sum(n):
    """Display the sum of the first n natural numbers."""
    total = sum_to_n(n)
    print(f"The sum of the first {n} natural numbers is: {total}")
if __name__ == "__main__":
    n = 10
    display_sum(n)
```

```
PS D:\AI assited coding> & C:/Users/vaish/AppData/Local/Programs/Python/Python314/python.exe "d:/AI assited coding/assignment_6.3.py"
Age: -3 - Classification: Invalid age

Age Classification using dictionary-based logic:
Age: 5 - Classification (Dict): Child
Age: 15 - Classification (Dict): Teenager
Age: 30 - Classification (Dict): Adult
Age: 70 - Classification (Dict): Senior
Age: -3 - Classification (Dict): Invalid age
The sum of the first 10 natural numbers is: 55
PS D:\AI assited coding>
```

#suggest an alternative implementation using the formula n(n + 1)/2.

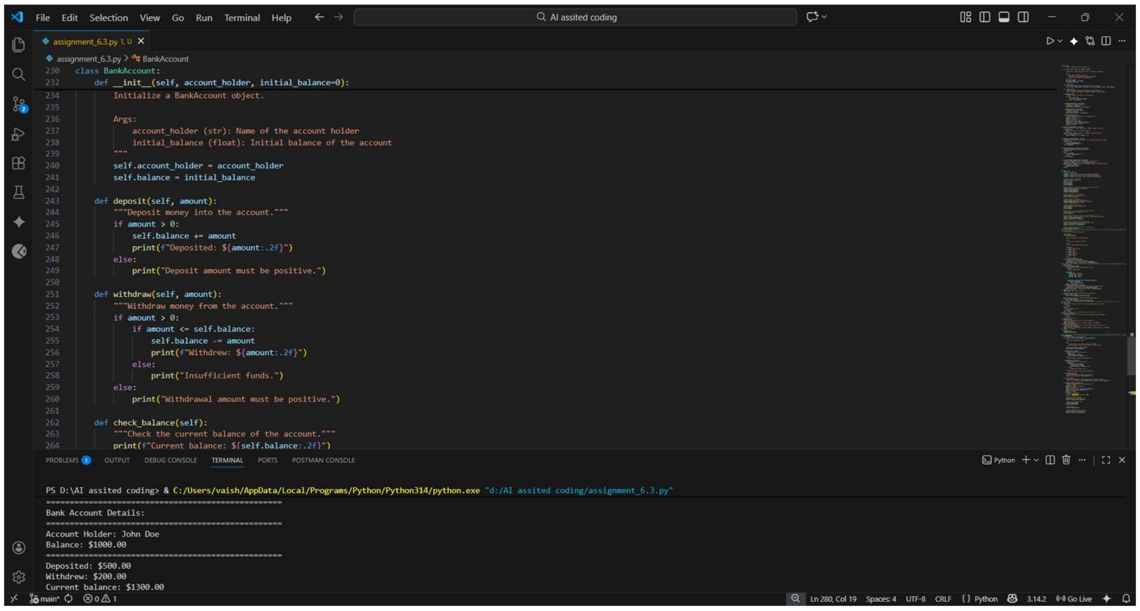


```python
#suggest an alternative implementation using the formula n(n + 1)/2.
def sum_to_n_formula(n):
    """Calculate the sum of the first n natural numbers using the formula."""
    if n <= 0:
        return 0
    return n * (n + 1) // 2
def display_sum(n):
    """Display the sum of the first n natural numbers."""
    total = sum_to_n(n)
    print(f"The sum of the first {n} natural numbers is: {total}")
def display_sum_formula(n):
    """Display the sum of the first n natural numbers using the formula."""
    total = sum_to_n_formula(n)
    print(f"The sum of the first {n} natural numbers (using formula) is: {total}")
if __name__ == "__main__":
    n = 10
    display_sum(n)
    display_sum_formula(n)
```

```
PS D:\AI assited coding> & C:/Users/vaish/AppData/Local/Programs/Python/Python314/python.exe "d:/AI assited coding/assignment_6.3.py"

Age Classification using dictionary-based logic:
Age: 5 - Classification (Dict): Child
Age: 15 - Classification (Dict): Teenager
Age: 30 - Classification (Dict): Adult
Age: 70 - Classification (Dict): Senior
Age: -3 - Classification (Dict): Invalid age
The sum of the first 10 natural numbers is: 55
The sum of the first 10 natural numbers (using formula) is: 55
PS D:\AI assited coding>
```

Task Description #5: Classes (Bank Account Class)

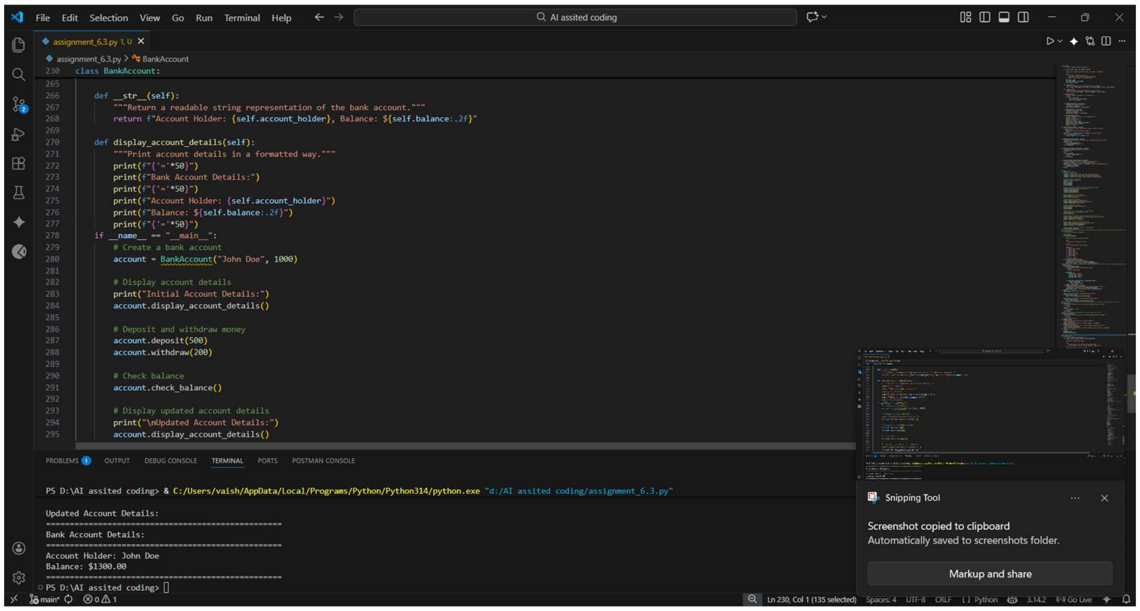
#you are designing a banking application generate a Bank Account class with methods such as deposit(), withdraw(),and check_balance()



```python
class BankAccount:
    def __init__(self, account_holder, initial_balance=0):
        Initialize a BankAccount object.

        Args:
            account_holder (str): Name of the account holder
            initial_balance (float): Initial balance of the account
        """
        self.account_holder = account_holder
        self.balance = initial_balance

    def deposit(self, amount):
        """Deposit money into the account."""
        if amount > 0:
            self.balance += amount
            print(f"Deposited: ${amount:.2f}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        """Withdraw money from the account."""
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew: ${amount:.2f}")
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

    def check_balance(self):
        """Check the current balance of the account."""
        print(f"Current balance: ${self.balance:.2f}")
```

```
PS D:\AI assited coding> & C:/Users/vaish/AppData/Local/Programs/Python/Python314/python.exe "d:/AI assited coding/assignment_6.3.py"
==================================================
Bank Account Details:
==================================================
Account Holder: John Doe
Balance: $1000.00
==================================================
Deposited: $500.00
Withdrew: $200.00
Current balance: $1300.00
```

#Add meaningful comments and explain the working of the code


File Edit Selection View Go Run Terminal Help
AI assited coding
assignment_6.3.py 1, U
assignment_6.3.py > BankAccount
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
Snipping Tool
Markup and share
main* 0 1
Ln 230, Col 1 (135 selected) Spaces: 4 UTF-8 CRLF Python 3.14.2 Go Live

```python
class BankAccount:

    def __str__(self):
        """Return a readable string representation of the bank account."""
        return f"Account Holder: {self.account_holder}, Balance: ${self.balance:.2f}"

    def display_account_details(self):
        """Print account details in a formatted way."""
        print(f"{'='*50}")
        print(f"Bank Account Details:")
        print(f"{'='*50}")
        print(f"Account Holder: {self.account_holder}")
        print(f"Balance: ${self.balance:.2f}")
        print(f"{'='*50}")
    if __name__ == "__main__":
        # Create a bank account
        account = BankAccount("John Doe", 1000)

        # Display account details
        print("Initial Account Details:")
        account.display_account_details()

        # Deposit and withdraw money
        account.deposit(500)
        account.withdraw(200)

        # Check balance
        account.check_balance()

        # Display updated account details
        print("\nUpdated Account Details:")
        account.display_account_details()
```

```
PS D:\AI assited coding> & C:/Users/vaish/AppData/Local/Programs/Python/Python314/python.exe "d:/AI assited coding/assignment_6.3.py"

Updated Account Details:
==================================================
Bank Account Details:
==================================================
Account Holder: John Doe
Balance: $1300.00
==================================================
PS D:\AI assited coding> 
```

Screenshot copied to clipboard
Automatically saved to screenshots folder.