

2303A51563

batch=10

Task Description #1: Classes (Student Class)

Scenario

You are developing a simple student information management module.

Task

- Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
- The class should include attributes such as name, roll number, and branch.
- Add a method display_details() to print student information.
- Execute the code and verify the output.
- Analyze the code generated by the AI tool for correctness and clarity.

Expected Output #1

- A Python class with a constructor (__init__) and a display_details() method.
- Sample object creation and output displayed on the console.
- Brief analysis of AI-generated code.

Code

class Student:

"""

A class to represent a student with basic information.

"""

def __init__(self, name, roll_number, branch):

"""

Constructor to initialize student attributes.

Parameters:

name (str): Name of the student

roll_number (int): Roll number of the student

branch (str): Branch/Department of the student

"""

self.name = name

self.roll_number = roll_number

self.branch = branch

def display_details(self):

"""

Displays the student details in a readable format.

"""

print("----- Student Details -----")

print("Name :", self.name)

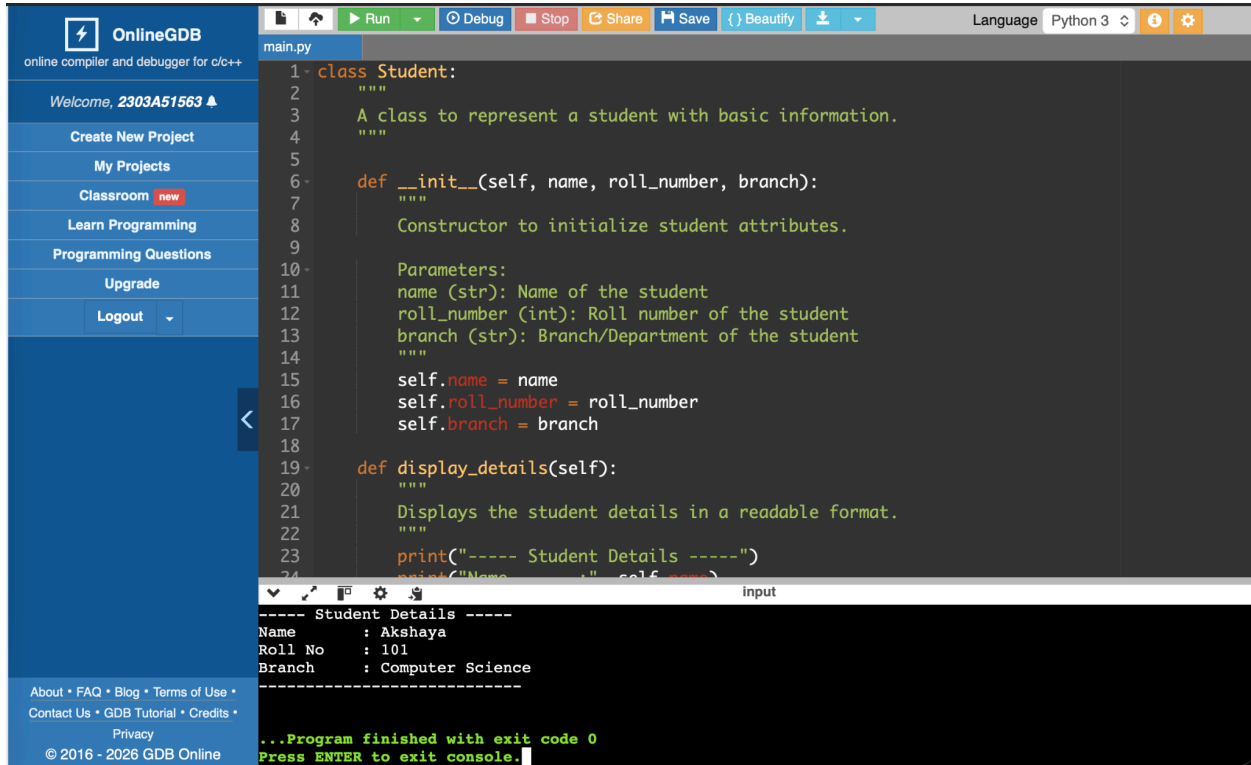
print("Roll No :", self.roll_number)

print("Branch :", self.branch)

print("-----")

```
# ✓ Sample Object Creation and Execution
student1 = Student("Akshaya", 101, "Computer Science")
```

```
# Display student details
student1.display_details()
```



The screenshot displays the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area shows a Python file named 'main.py' with the following code:

```
1 class Student:
2     """
3     A class to represent a student with basic information.
4     """
5
6     def __init__(self, name, roll_number, branch):
7         """
8         Constructor to initialize student attributes.
9
10        Parameters:
11        name (str): Name of the student
12        roll_number (int): Roll number of the student
13        branch (str): Branch/Department of the student
14        """
15        self.name = name
16        self.roll_number = roll_number
17        self.branch = branch
18
19    def display_details(self):
20        """
21        Displays the student details in a readable format.
22        """
23        print("----- Student Details -----")
24        print("Name : ", self.name)
```

The output console at the bottom shows the execution results:

```
----- Student Details -----
Name      : Akshaya
Roll No   : 101
Branch    : Computer Science
-----
...Program finished with exit code 0
Press ENTER to exit console.
```

The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python file named 'main.py' with the following code:

```
12 roll_number (int): Roll number of the student
13 branch (str): Branch/Department of the student
14 """
15 self.name = name
16 self.roll_number = roll_number
17 self.branch = branch
18
19 def display_details(self):
20     """
21     Displays the student details in a readable format.
22     """
23     print("----- Student Details -----")
24     print("Name       :", self.name)
25     print("Roll No    :", self.roll_number)
26     print("Branch     :", self.branch)
27     print("-----")
28
29
30 # Sample Object Creation and Execution
31 student1 = Student("Akshaya", 101, "Computer Science")
32
33 # Display student details
34 student1.display_details()
35
```

Below the code editor is an 'input' field and the program's output:

```
----- Student Details -----
Name       : Akshaya
Roll No    : 101
Branch     : Computer Science
-----

...Program finished with exit code 0
Press ENTER to exit console.
```

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

Scenario

You are writing a utility function to display multiples of a given number.

Task

- Prompt the AI tool to generate a function that prints the first 10 multiples of a given number using a loop.
- Analyze the generated loop logic.
- Ask the AI to generate the same functionality using another controlled looping structure (e.g., while instead of for).

Expected Output #2

- Correct loop-based Python implementation.
- Output showing the first 10 multiples of a number.
- Comparison and analysis of different looping approaches.

Code

```
def print_multiples_for(num):
```

```
    """
```

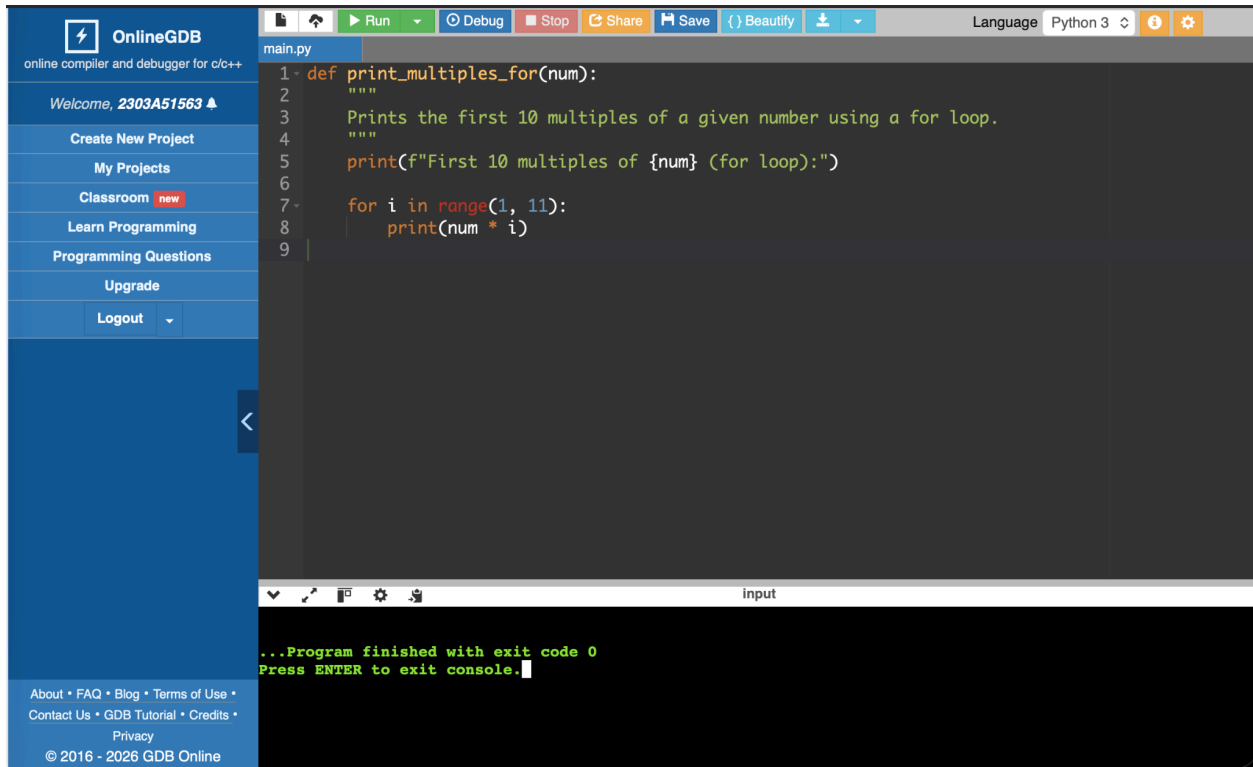
```
    Prints the first 10 multiples of a given number using a for loop.
```

```
    """
```

```
    print(f"First 10 multiples of {num} (for loop):")
```

```
    for i in range(1, 11):
```

```
        print(num * i)
```



ask Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

Task

- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).
- Analyze the generated conditions and logic.
- Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

Expected Output #3

- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work.

Code

```
def classify_age(age):
```

```
    """
```

```
    Classifies a person into an age group based on age.
```

```
    """
```

```
    if age < 13:
```

```
        return "Child"
```

```
    elif age < 20:
```

```
        return "Teenager"
```

```
    elif age < 60:
```

```

    return "Adult"
else:
    return "Senior"

```

The screenshot shows the OnlineGDB web interface. On the left is a blue sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python file named 'main.py' with the following code:

```

1 def classify_age(age):
2     """
3     Classifies a person into an age group based on age.
4     """
5
6     if age < 13:
7         return "Child"
8     elif age < 20:
9         return "Teenager"
10    elif age < 60:
11        return "Adult"
12    else:
13        return "Senior"
14

```

Below the code editor is a console window showing the output: '...Program finished with exit code 0' and 'Press ENTER to exit console.'

ask Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

Task

- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).
- Analyze the generated conditions and logic.
- Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

Expected Output #3

- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work.

Code

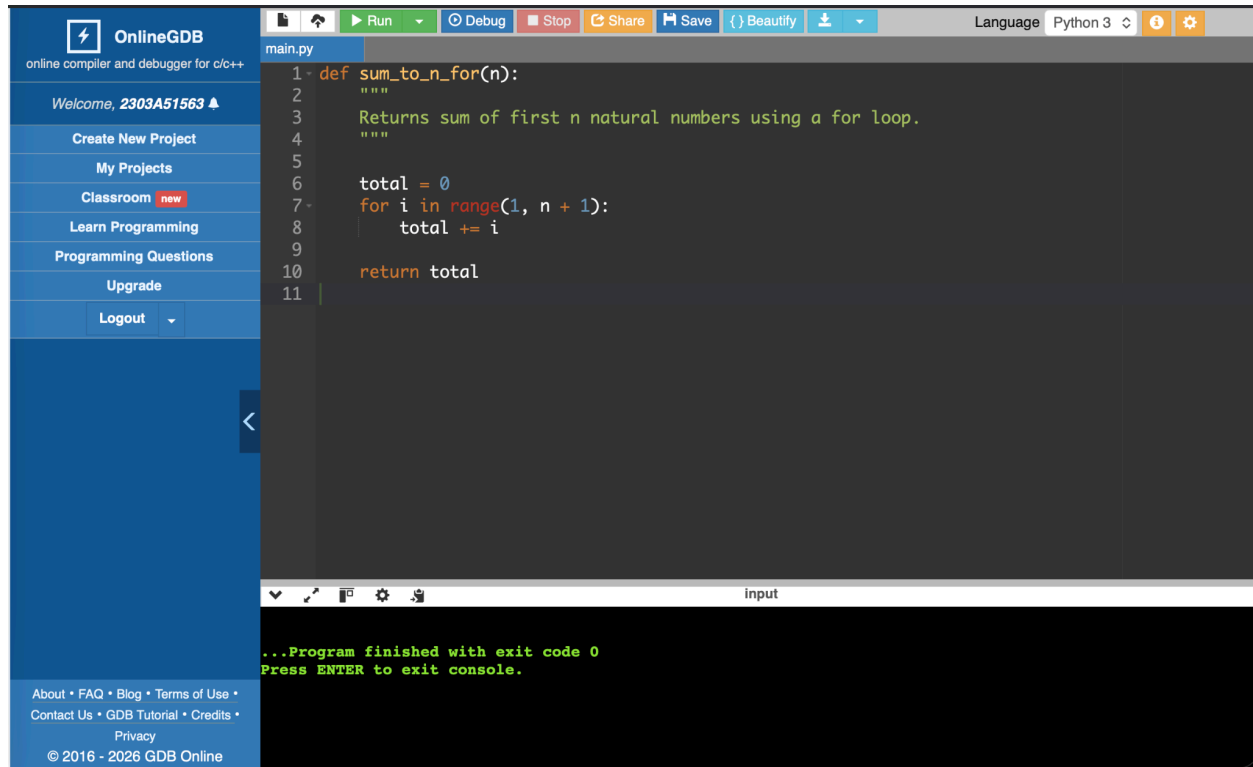
```

def sum_to_n_for(n):
    """
    Returns sum of first n natural numbers using a for loop.
    """

    total = 0

```

```
for i in range(1, n + 1):  
    total += i  
  
return total
```



The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python file named 'main.py' with the following code:

```
1 def sum_to_n_for(n):  
2     """  
3     Returns sum of first n natural numbers using a for loop.  
4     """  
5  
6     total = 0  
7     for i in range(1, n + 1):  
8         total += i  
9  
10    return total  
11
```

Below the code editor is a console window showing the output: '...Program finished with exit code 0' and 'Press ENTER to exit console.' The interface also includes a top toolbar with buttons for Run, Debug, Stop, Share, Save, and Beautify, and a language selector set to Python 3.

Task Description #5: Classes (Bank Account Class)

Scenario


You are designing a basic banking application.

Task

- Use AI tools to generate a Bank Account class with methods such as `deposit()`, `withdraw()`, and `check_balance()`.
- Analyze the AI-generated class structure and logic.
- Add meaningful comments and explain the working of the code.

Expected Output #5

- Complete Python Bank Account class.
- Demonstration of deposit and withdrawal operations with updated balance.
- Well-commented code with a clear explanation

 **OnlineGDB**
online compiler and debugger for c/c++

Welcome, 2303A51563 ▲

Create New Project

My Projects

Classroom new

Learn Programming

Programming Questions

Upgrade

Logout ▼

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online


Run Debug Stop Share Save {} Beautify

main.py

```
1 class BankAccount:
2     """
3     A simple Bank Account class that supports deposit,
4     withdrawal, and balance checking.
5     """
6
7     def __init__(self, account_holder, balance=0):
8         """
9         Constructor initializes account holder name and balance.
10        """
11        self.account_holder = account_holder
12        self.balance = balance
13
14    def deposit(self, amount):
15        """
16        Adds money to the account.
17        """
18        if amount > 0:
19            self.balance += amount
20            print(f"Deposited ₹{amount}. New Balance: ₹{self.balance}")
21        else:
22            print("Deposit amount must be positive!")
23
24    def withdraw(self, amount):
```

input

...Program finished with exit code 0
Press ENTER to exit console.

 **OnlineGDB**
online compiler and debugger for c/c++

Welcome, 2303A51563 ▲

Create New Project

My Projects

Classroom new

Learn Programming

Programming Questions

Upgrade

Logout ▼

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online

Run Debug Stop Share Save {} Beautify

main.py

```
18        self.balance += amount
19        print(f"Deposited ₹{amount}. New Balance: ₹{self.balance}")
20    else:
21        print("Deposit amount must be positive!")
22
23    def withdraw(self, amount):
24        """
25        Withdraws money if sufficient balance is available.
26        """
27        if amount > self.balance:
28            print("Insufficient balance!")
29        elif amount <= 0:
30            print("Withdrawal amount must be positive!")
31        else:
32            self.balance -= amount
33            print(f"Withdrew ₹{amount}. Remaining Balance: ₹{self.balance}")
34
35    def check_balance(self):
36        """
37        Displays the current account balance.
38        """
39        print(f"Account Balance for {self.account_holder}: ₹{self.balance}")
40
41
```

input

...Program finished with exit code 0
Press ENTER to exit console.