

Assignment 6.3 Ai Assisted Coding

Htno:2303a51305

Btno:05

Task 1: Classes (Student Class).

Prompt: Write a Python program to create a Student class.

Requirements:

- The class should have attributes: name, roll_number, and branch.
- Use a constructor (**init**) to initialize the values.
- Add a method `display_details()` that prints all student details clearly.
- Create at least one student object and call the method.
- Keep the code simple and beginner-friendly.

Input:

The screenshot shows a Google Colab notebook titled "Untitled29.ipynb". On the left, there's a sidebar with a list of requirements:

- Add a method `display_details()` that prints all student details clearly.
- Create at least one student object and call the method.
- Keep the code simple and beginner-friendly.

The main code editor contains the following Python code:

```
class Student:  
    def __init__(self, name, roll_number, branch):  
        self.name = name  
        self.roll_number = roll_number  
        self.branch = branch  
  
    def display_details(self):  
        print(f"Student Name: {self.name}")  
        print(f"Roll Number: {self.roll_number}")  
        print(f"Branch: {self.branch}")  
  
# Create a student object  
student1 = Student("snehanjali", "101", "Computer Science")  
  
# Call the display_details method  
student1.display_details()
```

To the right of the code editor, there's a "Release notes" panel with the following content:

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

At the bottom right, it shows "9:44 AM" and "Python 3".

Output:

The screenshot shows a Google Colab interface. On the left, there's a notebook titled "Student Details" containing the following code:

```
os
-----
Student Details
...
Name: Snehanjali Uppu
Roll Number: 2303A51305
Branch: CSE
```

On the right, there's a sidebar with "Release notes". The sidebar includes a link to a blog post about analyzing a bank failure with Colab. It also lists several updates from January 2026:

- Launched Data Explorer - a new feature to search Kaggle datasets, models, and more directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, it is now available for use in Antigravity, Curse, and via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for Python package upgrades
 - accelerate 1.11.0 → 1.12.0
 - astropy 7.1.1 → 7.2.0
 - bigframes 2.28.0 → 2.31.0
 - cachetools 5.5.2 → 6.2.4

Explanation: The program defines a `Student` class with attributes (name, roll number, branch) initialized using a constructor.

The `display_details()` method prints the student information, showing basic OOP concepts like class, object, and methods.

Task 2: Loops of a number.

Prompt: Write a Python function that prints the first 10 multiples of a given number.

Requirements:

- Use a loop structure.
- The number should be passed as a parameter.
- Print the multiples clearly.
- Keep the code simple and beginner-friendly.

Then write another version of the same program using a different loop (while loop instead of for loop).

Input:

Task 2: Loops (Multiples of a Number)

Here's the Python function using a `for` loop:

```
3] Os ⏎ def print_multiples_for_loop(number):
    print(f"Multiples of {number} (using for loop):")
    for i in range(1, 11): # Loop from 1 to 10 (inclusive)
        print(f"{number} x {i} = {number * i}")

    # Example usage:
    print_multiples_for_loop(5)

'... Multiples of 5 (using for loop):'
```

Output:

```
3] Os ⏎
    print(f"\n{number} x {i} = {number * i}")

# Example usage:
print_multiples_for_loop(5)

'... Multiples of 5 (using for loop):'
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

2) Input:

And here's the Python function using a `while` loop:

```
[1]  Os  def print_multiples_while_loop(number):
    print(f"\nMultiples of {number} (using while loop):")
    i = 1
    while i <= 10:
        print(f"{number} x {i} = {number * i}")
        i += 1

    # Example usage:
    print_multiples_while_loop(7)

...
Multiples of 7 (using while loop):
7 x 1 = 7
7 x 2 = 14
```

2) Output:

```
[1]  Os  print_multiples_while_loop()

Multiples of 7 (using while loop):
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
```

Explanation: A loop is used to repeat an action 10 times and calculate multiples of a number using multiplication.

for loop is best for fixed iterations, while while loop gives more control using a condition.

Task 3: Conditional Statements (Age Classification)

Prompt: Write a Python function to classify a person based on age.

Requirements:

- Use nested if-elif-else statements.

- Age groups:

Child: 0–12

Teenager: 13–19

Adult: 20–59

Senior: 60 and above

The function should take age as input and print the category. Then generate another version using an alternative approach (simplified conditions or dictionary-based logic). Keep the code beginner-friendly.

Input: using if else statements

The screenshot shows a Google Colab notebook titled "Untitled29.ipynb". The code defines a function `classify_person_if_else(age)` that prints the age group based on the input age. The code includes comments explaining the logic and handling invalid input. The notebook interface shows the code in the left panel and a "Release notes" sidebar on the right. The sidebar highlights the launch of Data Explorer, Gemini 3 availability, and other improvements. The status bar at the bottom indicates the time as 10:01AM and the Python version as Python 3.

```
def classify_person_if_else(age):
    """
    Classifies a person based on age using if-elif-else statements.
    """
    if not isinstance(age, (int, float)) or age < 0:
        print(f"Invalid age: {age}. Age must be a non-negative number.")
        return

    if age <= 12:
        print(f"Age {age}: Child")
    elif age < 19:
        print(f"Age {age}: Teenager")
    elif age < 59:
        print(f"Age {age}: Adult")
    else: # age >= 60
        print(f"Age {age}: Senior")

# Example usage:
print("--- Using if-elif-else ---")
classify_person_if_else(5)      # Child
classify_person_if_else(15)     # Teenager
classify_person_if_else(30)     # Adult
```

Output:

A screenshot of a Google Colab notebook titled "Untitled29.ipynb". The code cell contains the following Python code:

```
print(f"Age {age}: Senior")  
# Example usage:  
print("--- Using if-elif-else ---")  
classify_person_if_elif_else(5) # Child  
classify_person_if_elif_else(15) # Teenager  
classify_person_if_elif_else(30) # Adult  
classify_person_if_elif_else(70) # Senior  
classify_person_if_elif_else(-5) # Invalid age  
classify_person_if_elif_else("ten") # Invalid age  
... --- Using if-elif-else ---  
Age 5: Child  
Age 15: Teenager  
Age 30: Adult  
Age 70: Senior  
Invalid age: -5. Age must be a non-negative number.  
Invalid age: ten. Age must be a non-negative number.
```

The output shows the classification of various ages and handles invalid inputs.

2)input:using alternative methods

A screenshot of a Google Colab notebook titled "Untitled29.ipynb". The code cell contains the following Python code:

```
def classify_person_alternative(age):  
    """  
    Classifies a person based on age using an alternative approach  
    (list of age ranges).  
    """  
    if not isinstance(age, (int, float)) or age < 0:  
        print(f"Invalid age: {age}. Age must be a non-negative number.")  
        return  
  
    # Define age groups as (upper_bound, category_name)  
    age_groups = [  
        (12, "Child"),  
        (19, "Teenager"),  
        (59, "Adult"),  
        (float('inf'), "Senior") # Use infinity for the upper bound of the last group  
    ]  
  
    for upper_bound, category in age_groups:  
        if age <= upper_bound:  
            print(f"Age {age}: {category}")  
            return
```

The sidebar on the right displays the "Release notes" and "Python package upgrades" sections.

Output:

The screenshot shows a Google Colab notebook titled "Untitled29.ipynb". In the code cell, the following output is displayed:

```
--- Using Alternative Approach (list of ranges) ---
Age 8: Child
Age 18: Teenager
Age 45: Adult
Age 85: Senior
Invalid age: -10. Age must be a non-negative number.
Invalid age: twenty. Age must be a non-negative number.
```

On the right side of the interface, there is a "Release notes" panel. It includes a link to the blog for more information about new features, tips and tricks, and featured notebooks such as "Analyzing a Bank Failure with Colab". Below this, the date "2026-01-20" is shown, followed by a list of changes:

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Under "Python package upgrades", the following changes are listed:

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

The bottom status bar shows the time as 10:01AM and the Python version as Python 3.

Explanation: Conditional statements (**if-elif-else**) check age ranges and assign a category like child, teenager, adult, or senior.

The logic works top-to-bottom, and alternative methods like dictionary mapping make the system more flexible.

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

Prompt: Write a Python function `sum_to_n(n)` to calculate the sum of the first n natural numbers.

Requirements:

- Use a for loop.
- Keep the code simple.
- Show example function call and output.

Then provide an alternative implementation using:

- 1) a while loop
- 2) a mathematical formula

Explain the differences briefly.

1 Input:

The screenshot shows the Google Colab interface. In the main code editor area, there is a comment block that reads: "First, here's the Python function using a for loop:". Below this, the function definition is provided:

```
def sum_to_n_for_loop(n):
    """
    Calculates the sum of the first n natural numbers using a for loop.
    """
    if not isinstance(n, int) or n < 1:
        print("Input must be a positive integer.")
        return None

    total_sum = 0
    for i in range(1, n + 1):
        total_sum += i
    return total_sum

# Example usage:
print(f"Sum of first 10 natural numbers (for loop): {sum_to_n_for_loop(10)}")
print(f"Sum of first 5 natural numbers (for loop): {sum_to_n_for_loop(5)}")
print(f"Sum of first 0 natural numbers (for loop): {sum_to_n_for_loop(0)}") # Inv.
```

In the bottom right corner of the code editor, there is a timestamp "✓ 10:07 AM" and a "Python 3" indicator. To the right of the code editor, there is a "Release notes" sidebar with the following content:

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

Output:

The screenshot shows the Google Colab interface. The code editor contains the same Python function as before. When the "Run all" button is clicked, the output pane displays the results of the function calls:

```
if not isinstance(n, int) or n < 1:
    print("Input must be a positive integer.")
    return None

total_sum = 0
for i in range(1, n + 1):
    total_sum += i
return total_sum

# Example usage:
print(f"Sum of first 10 natural numbers (for loop): {sum_to_n_for_loop(10)}")
print(f"Sum of first 5 natural numbers (for loop): {sum_to_n_for_loop(5)}")
print(f"Sum of first 0 natural numbers (for loop): {sum_to_n_for_loop(0)}") # Inv.

...
Sum of first 10 natural numbers (for loop): 55
Sum of first 5 natural numbers (for loop): 15
Input must be a positive integer.
Sum of first 0 natural numbers (for loop): None
```

To the right of the code editor, there is a "Release notes" sidebar with the following content:

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data search Kaggle directly from a
- Gemini 3 is now
- In addition to b available for us via the Open V
- H100 is being r
- Launched a ne

Python package upgr

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

2) input:

The screenshot shows the Google Colab interface. In the center, there is a code editor window titled "Untitled29.ipynb" containing Python code for calculating the sum of natural numbers using a while loop. To the right of the code editor is a "Release notes" sidebar. The sidebar includes a link to the blog for more information about new features, tips and tricks, and featured notebooks such as "Analyzing a Bank Failure with Colab". Below this, a section titled "2026-01-20" lists several updates: launching Data Explorer, Gemini 3 availability, H100 roll-out, and a new modern design for the Colab UI. At the bottom of the sidebar, there is a list of Python package upgrades. The bottom right corner of the interface shows the current time as 10:07 AM and the Python version as Python 3.

```
def sum_to_n_while_loop(n):
    """
    Calculates the sum of the first n natural numbers using a while loop.
    """
    if not isinstance(n, int) or n < 1:
        print("Input must be a positive integer.")
        return None

    total_sum = 0
    counter = 1
    while counter <= n:
        total_sum += counter
        counter += 1
    return total_sum

# Example usage:
print(f"Sum of first 10 natural numbers (while loop): {sum_to_n_while_loop(10)}")
print(f"Sum of first 5 natural numbers (while loop): {sum_to_n_while_loop(5)}")
```

Output:

The screenshot shows the Google Colab interface with the same code editor and release notes sidebar as the previous screenshot. The main difference is in the output pane at the bottom, which displays the results of the code execution. It shows two lines of text: "... Sum of first 10 natural numbers (while loop): 55" and "Sum of first 5 natural numbers (while loop): 15".

```
... Sum of first 10 natural numbers (while loop): 55
Sum of first 5 natural numbers (while loop): 15
```

3)input:

Finally, the most efficient approach using a mathematical formula (Gauss's formula):

```
[9] 0s
def sum_to_n_formula(n):
    """
    Calculates the sum of the first n natural numbers using a mathematical formula
    """
    if not isinstance(n, int) or n < 1:
        print("Input must be a positive integer.")
        return None

    return n * (n + 1) // 2 # Using integer division

# Example usage:
print(f"Sum of first 10 natural numbers (formula): {sum_to_n_formula(10)}")
print(f"Sum of first 5 natural numbers (formula): {sum_to_n_formula(5)}")
```

... Sum of first 10 natural numbers (formula): 55

Release notes

Please follow our [blo](#) features, tips and tric [Analyzing a Bank Fail](#)

2026-01-20

- Launched Data search Kaggle directly from a
- Gemini 3 is now
- In addition to b available for us via the Open V
- H100 is being i
- Launched a ne

Python package upg!

- accelerate 1.11.
- astropy 7.1.1 ->

Output:

```
... Sum of first 10 natural numbers (formula): 55
... Sum of first 5 natural numbers (formula): 15
```

Release notes

Please follow our [blo](#) features, tips and tric [Analyzing a Bank Fail](#)

2026-01-20

- Launched Data search Kaggle directly from a
- Gemini 3 is now
- In addition to b available for us via the Open V
- H100 is being i
- Launched a ne

Python package upg!

- accelerate 1.11.
- astropy 7.1.1 ->

Explanation:

- In the **for loop**, the program repeats from 1 to n and keeps adding numbers to **total**.
- In the **while loop**, the same addition happens but the loop runs based on a condition (**i <= n**).

Task 5: Classes (Bank Account Class)

Prompt: Write a Python program to create a BankAccount class.

Requirements:

- Attributes: account_holder, balance

- Methods:

deposit(amount) – add money

withdraw(amount) – deduct money if sufficient balance

check_balance() – display current balance

- Include comments in the code.

- Create an object and demonstrate deposit and withdrawal.

- Keep it simple and beginner-friendly.

Input:

The screenshot shows a Google Colab interface with a notebook titled "Untitled29.ipynb". The code cell contains the following Python code for a BankAccount class:

```
[10]  ✓ 0s
class BankAccount:
    # Constructor to initialize account_holder and balance
    def __init__(self, account_holder, initial_balance=0):
        self.account_holder = account_holder
        self.balance = initial_balance
        print(f"Account created for {self.account_holder} with initial balance ${initial_balance:.2f}")

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

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

# Create a BankAccount object
account = BankAccount("Alice", 1000)
# Deposit money
account.deposit(500)
# Withdraw money
account.withdraw(200)
# Check balance
print(account.check_balance())
```

The notebook also includes a "Release notes" sidebar with information about Data Explorer, Gemini 3, and Python package upgrades.

```
else:
    self.balance -= amount
    print(f"Withdraw ${amount:.2f}. New balance: ${self.balance:.2f}")

# Method to display the current balance
def check_balance(self):
    print(f"Account holder: {self.account_holder}, Current balance: ${self.ba
# --- Demonstrate the BankAccount class ---

# Create a new bank account object
my_account = BankAccount("John Doe", 100.00)

# Check initial balance
my_account.check_balance()

# Deposit some money
my_account.deposit(50.50)

# Withdraw some money
my_account.withdraw(20.00)
```

Release notes

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

Output:

```
Account created for John Doe with initial balance $100.00
Account holder: John Doe, Current balance: $100.00
Deposited $50.50. New balance: $150.50
Withdraw $20.00. New balance: $130.50
Insufficient balance. Current balance: $130.50
Account holder: John Doe, Current balance: $130.50
Deposit amount must be positive.
Withdrawal amount must be positive.
```

Release notes

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4

Explanation:

- The **class** represents a bank account.
- The **constructor (`__init__`)** sets account holder name and initial balance.
- `deposit()` increases balance.
- `withdraw()` checks balance before deducting (prevents overdraft).
- `check_balance()` shows account details.

