# ASSIGNMENT 6.3

Task Description #1: Classes (Student Class)Scenario

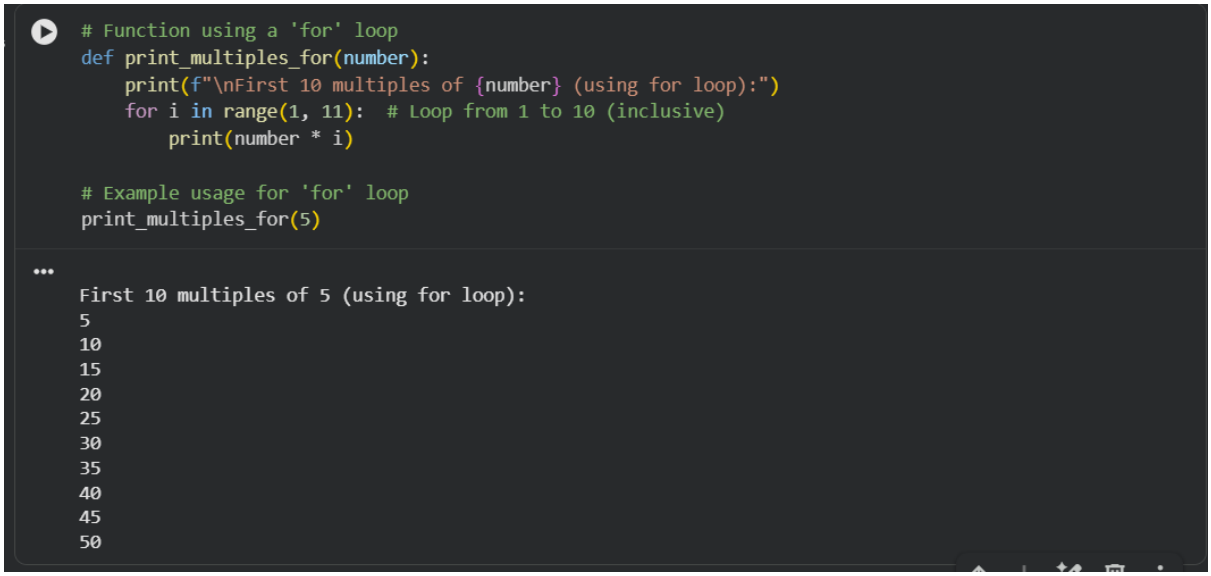
You are developing a simple student information management module

```
[1]
✓ 0s
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 sample student object
student1 = Student("Alice Smith", "CSE001", "Computer Science")

# Display student information
student1.display_details()

...
Student Name: Alice Smith
Roll Number: CSE001
Branch: Computer Science
```

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

Scenario

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



```
# Function using a 'for' loop
def print_multiples_for(number):
    print(f"\nFirst 10 multiples of {number} (using for loop):")
    for i in range(1, 11):  # Loop from 1 to 10 (inclusive)
        print(number * i)

# Example usage for 'for' loop
print_multiples_for(5)

...
First 10 multiples of 5 (using for loop):
5
10
15
20
25
30
35
40
45
50
```

```python
# Function using a 'while' loop
def print_multiples_while(number):
    print(f"\nFirst 10 multiples of {number} (using while loop):")
    count = 1  # Initialize a counter
    while count <= 10:  # Loop condition: continue as long as count is 10 or less
        print(number * count)
        count += 1  # Increment the counter in each iteration

# Example usage for 'while' loop
print_multiples_while(7)
```

```
First 10 multiples of 7 (using while loop):
7
14
21
28
35
42
49
56
63
70
```

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

```python
def classify_age_nested(age):
    """
    Classifies an input age into 'Child', 'Teenager', 'Adult', or 'Senior'
    using nested if-elif-else conditional statements.
    """
    if age < 13:
        return 'Child'
    elif age < 20:
        return 'Teenager'
    elif age < 65:
        return 'Adult'
    else:
        return 'Senior'
print(f"Age 5 is classified as: {classify_age_nested(5)}")
print(f"Age 15 is classified as: {classify_age_nested(15)}")
print(f"Age 30 is classified as: {classify_age_nested(30)}")
print(f"Age 70 is classified as: {classify_age_nested(70)}")
print(f"Age 12 is classified as: {classify_age_nested(12)}")
print(f"Age 19 is classified as: {classify_age_nested(19)}")
print(f"Age 64 is classified as: {classify_age_nested(64)}")
print(f"Age 65 is classified as: {classify_age_nested(65)}")
```

```
Age 5 is classified as: Child
Age 15 is classified as: Teenager
Age 30 is classified as: Adult
Age 70 is classified as: Senior
Age 12 is classified as: Child
Age 19 is classified as: Teenager
Age 64 is classified as: Adult
Age 65 is classified as: Senior
```

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

Scenario

You need to calculate the sum of the first n natural numbers.

[6]
✓ 0s

```
def sum_to_n_for(n):
    """
    Calculates the sum of the first 'n' natural numbers using a for loop.
    """
    total_sum = 0
    for i in range(1, n + 1):
        total_sum += i
    return total_sum

# Example usage with a sample input
n_value = 10
result = sum_to_n_for(n_value)
print(f"The sum of the first {n_value} natural numbers is: {result}")

n_value_2 = 5
result_2 = sum_to_n_for(n_value_2)
print(f"The sum of the first {n_value_2} natural numbers is: {result_2}")
```

```
The sum of the first 10 natural numbers is: 55
The sum of the first 5 natural numbers is: 15
```

[7]
✓ 0s

```
def sum_to_n_alternative(n):
    """
    Calculates the sum of the first 'n' natural numbers using the mathematical formula.
    """
    if n < 0:
        raise ValueError("n must be a non-negative integer.")
    return n * (n + 1) // 2

# Example usage with a sample input
n_value = 10
result = sum_to_n_alternative(n_value)
print(f"The sum of the first {n_value} natural numbers (alternative) is: {result}")

n_value_2 = 5
result_2 = sum_to_n_alternative(n_value_2)
print(f"The sum of the first {n_value_2} natural numbers (alternative) is: {result_2}")
```

```
The sum of the first 10 natural numbers (alternative) is: 55
The sum of the first 5 natural numbers (alternative) is: 15
```

Task Description #5: Classes (Bank Account Class)

Scenario

You are designing a basic banking application

```python
class BankAccount:
    def __init__(self, account_number, account_holder, initial_balance=0.0):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = initial_balance
        print(f"Account {self.account_number} created for {self.account_holder} with initial balance $

    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.")

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

    def check_balance(self):
        print(f"Current balance for account {self.account_number}: ${self.balance:.2f}")
my_account = BankAccount("123456789", "John Doe", 1000.00)
```

```python
    def check_balance(self):
        print(f"Current balance for account {self.account_number}: ${self.balance:.2f}")
my_account = BankAccount("123456789", "John Doe", 1000.00)

my_account.check_balance()

my_account.deposit(500.00)

my_account.withdraw(200.00)

my_account.withdraw(2000.00)

my_account.check_balance()
my_account.deposit(-100.00)
my_account.withdraw(0.00)
```

```
Account 123456789 created for John Doe with initial balance $1000.00
Current balance for account 123456789: $1000.00
Deposited $500.00. New balance: $1500.00
Withdrew $200.00. New balance: $1300.00
Insufficient funds. Current balance: $1300.00
Current balance for account 123456789: $1300.00
Deposit amount must be positive.
Withdrawal amount must be positive.
```