LAB ASSIGNMENT-7.2

<AI ASSISTED CODING>

Name : Bommineni Spandana

Hall ticket No. : 2403A52005

Batch No. : 24BTCAIAIB01

Course Title : AI Assisted Coding

➢ Task Description#1

- Task #1 – Syntax Error in Conditionals

```python
a = 10
if a = 10:
    print("Equal")
```

➢ Expected Output#1

- Corrected function with syntax fix.

➢ Prompt :

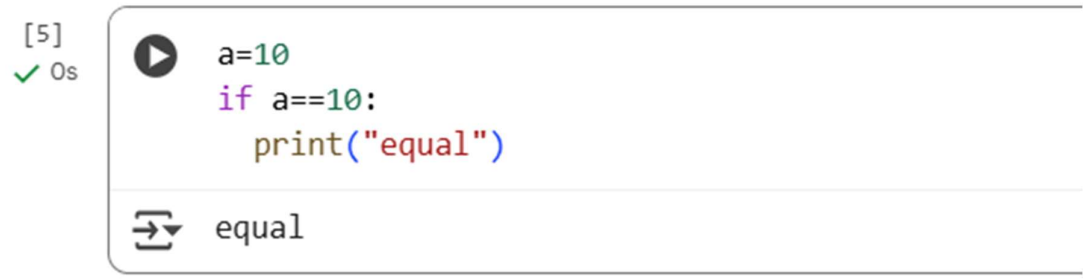
The following Python code contains a syntax error in its conditional statement: a = 10 if a = 10: print("Equal") Identify the error in the if statement and rewrite the code to produce the expected output

Error of the code :

```
[4]
0s
a=10
if a=10:
  print("equal")

  File "/tmp/ipython-input-2241231774.py", line 2
    if a=10:
       ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of
```

Correction of the code :



➢ Explanation :

This code snippet initializes a variable a with the value 10. Then, it checks if the value of a is equal to 10 using an if statement. If the condition is true, it prints the string "equal" to the console.

➢ Comments :

Initialize variable 'a' with value 10

Check if 'a' is equal to 10

If 'a' is equal to 10, print "equal"

➢ Task Description#2 (Loops)

- Task #2 – Loop Off-By-One Error.

```
def sum_upto_n(n):
    total = 0
    for i in range(1, n):
        total += i
    return total
```

➢ Expected Output#2

- AI fixes increment/decrement error.

➢ Prompt :

Calculate the sum of integers from 1 to n: def sum_upto_n(n): total = 0 for i in range(1, n): total += i return total The function currently returns an incorrect

result due to a loop boundary issue. Can you identify and fix the off-by-one error so the function correctly includes n in the summation

Error of the code

```
def sum_upto_n(n):
  total = 0
  for i in range(1, n ):
    total += i
  return total
n = 5
sum_result = sum_upto_n(n)
print(f"The sum of integers from 1 to {n} is: {sum_result}")
```

```
The sum of integers from 1 to 5 is: 10
```

Correction of the code

[2] ✓ 0s

```
def sum_upto_n(n):
  total = 0
  for i in range(1, n + 1):
    total += i
  return total
n = 5
sum_result = sum_upto_n(n)
print(f"The sum of integers from 1 to {n} is: {sum_result}")
```

```
The sum of integers from 1 to 5 is: 15
```

➢ Explanation :

The code calculates the sum of integers from 1 to 5, which is 1 + 2 + 3 + 4 + 5 = 15, and then prints this result.

➢ Comments :

Calculates the sum of integers from 1 to n (inclusive)
Initialize a variable to store the sum
Loop from 1 up to and including n
Add the current number to the total
Return the final sum

Set the value of n
Call the function and store the result.

- **Task Description#3**
  - Error: AttributeError.

```
class User:
    def __init__(self, name):
        self.name = name

u = User("Alice")
print(u.getName())
```

- **Expected Output#3**
  - Identify the missing method and correct the code.
- **Prompt :**

Consider the following Python class definition and usage:
class User: def _init_(self, name): self.name = name u = User("Alice")
print(u.getName())
Running this code results in an AttributeError.
Can you identify the cause of the error and modify the class so that it correctly prints the user's name.

Error of the code :

```
class User:
    def __init__(self, name):
        self.name = name

u = User("Alice")
print(u.getName())
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-2327273307.py in <cell line: 0>()
      4
      5 u = User("Alice")
----> 6 print(u.getName())

AttributeError: 'User' object has no attribute 'getName'
```

Correction of the code

```
class User:
    def __init__(self, name):
        self.name = name

    def getName(self):
        return self.name

u = User("Alice")
print(u.getName())
```

```
Alice
```

- **Explanation :**

This code demonstrates how to create a basic class to represent a user with a name and how to access that name using a method.

- **Comments :**

Define a class named User
Constructor method to initialize a User object
Assign the provided name to the object's name attribute
Method to get the user's name
Return the value of the name attribute
Create an instance of the User class with the name "Alice"
Call the getName method on the user object and print the result.

- **Task Description#4**

  • Incorrect Class Attribute Initialization.

```
class Car:
    def start():
        print("Car started")

mycar = Car()
mycar.start()
```

- **Expected Output#4**

  • Detect missing self and initialize attributes properly.

- **Prompt :**

Examine the following Python class definition and method call:

class Car: def start(): print("Car started") mycar = Car() mycar.start()

Running this code results in a TypeError because of incorrect method definition.Can you identify the issue with the start() method and rewrite the class so that the method works properly.

Error of the code

```
class Car:
    def start(): # Added 'self' as the first parameter
        print("Car started")

mycar = Car()
mycar.start()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-3633105302.py in <cell line: 0>()
      4
      5 mycar = Car()
----> 6 mycar.start()

TypeError: Car.start() takes 0 positional arguments but 1 was given
```

Correction of the code

```
class Car:
    def start(self): # Added 'self' as the first parameter
        print("Car started")

mycar = Car()
mycar.start()
```

```
Car started
```

➢ Explanation :

This code creates a "Car" object and then tells that car object to "start", which results in the message "Car started" being printed.

➢ Comments :

Define a class named Car

Added 'self' as the first parameter

Print a message indicating the car has started

Create an instance of the Car class

Call the start method on the mycar object

- **Task Description#5**
  - Conditional Logic Error in Grading System.

```
def grade_student(score):
    if score < 40:
        return "A"
    elif score < 70:
        return "B"
    else:
        return "C"
```

- **Expected Output#5**
  - Detect illogical grading and correct the grade levels.
- **Prompt :**

Write a Python function called grade_student(score) that returns a grade based on the score: - Return "A" if the score is less than 40 - Return "B" if the score is between 40 and 69 (inclusive of 40, exclusive of 70) - Return "C" if the score is 70 or above Then, test your function with the following scores: 35, 55, and 85. What grades do you get .

Error of the code

```
def grade_student(score):
    if score < 40:
        return "A"
    elif score < 70:
        return "B"
    else:
        return "C"

# Test cases to show the incorrect output
print(f"Score 35 gets grade: {grade_student(35)}") # Expected: something lower than A, Actual: A
print(f"Score 65 gets grade: {grade_student(65)}") # Expected: something lower than B, Actual: B
print(f"Score 85 gets grade: {grade_student(85)}") # Expected: something higher than C, Actual: C
```

```
Score 35 gets grade: A
Score 65 gets grade: B
Score 85 gets grade: C
```

Correction of the code

```python
def grade_student_corrected(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"

# Test cases for the corrected function
print(f"Score 95 gets grade: {grade_student_corrected(95)}")
print(f"Score 85 gets grade: {grade_student_corrected(85)}")
print(f"Score 75 gets grade: {grade_student_corrected(75)}")
print(f"Score 65 gets grade: {grade_student_corrected(65)}")
print(f"Score 55 gets grade: {grade_student_corrected(55)}")
```

```
Score 95 gets grade: A
Score 85 gets grade: B
Score 75 gets grade: C
Score 65 gets grade: D
Score 55 gets grade: F
```

➢ Explanation :

This code checks a student's score and gives a grade:

- 90 or more → A
- 80–89 → B
- 70–79 → C
- 60–69 → D
- Below 60 → F

Then it prints the grade for different scores. Simple use of if, elif, and else.

➢ Comments :

This function takes a score and returns a letter grade

def grade_student_corrected(score):

```
    # If score is 90 or more, return grade A
    # If score is 80 to 89, return grade B
    # If score is 70 to 79, return grade C
    # If score is 60 to 69, return grade D
# If score is below 60, return grade F
# Test cases to check how the function works
# Should print A
# Should print B
# Should print C
# Should print D
# Should print F
```