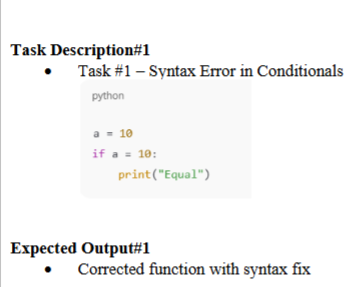
HTNO:2403A51319

ASSIGNMENT:7.2

DEBUGGING



ERROR CODE:

A screenshot of a computer

AI-generated content may be incorrect.

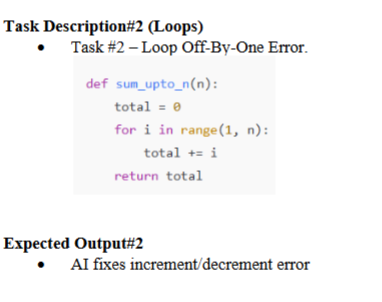
CORRECTED CODE:

A screenshot of a computer

AI-generated content may be incorrect.

EXPLANATION:

1. a = 10: This line assigns the integer value 10 to the variable named a.
2. if a == 10:: This is an if statement, which checks if a condition is true. The condition here is a == 10. The == operator checks if the value of a is equal to 10. Since a was assigned the value 10 in the previous line, this condition is true.
3. print("Equal"): This line is inside the if statement. Because the condition a == 10 is true, the code inside the if block is executed. This line prints the string "Equal" to the console.



ERROR CODE:

A screen shot of a computer code

AI-generated content may be incorrect.

CORRECT CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

EXPLANATION:

1. **n\_value\_str = input("Please enter a positive integer for n: ")**: This line displays the message "Please enter a positive integer for n: " to the user and waits for them to type something and press Enter. Whatever the user types is stored as a string in the variable n\_value\_str.
2. **try...except ValueError:**: This is a try-except block, which is used for error handling. It attempts to run the code inside the try block. If a ValueError occurs (which happens if you try to convert a non-integer string to an integer), the code inside the except ValueError: block is executed.
3. **n\_value = int(n\_value\_str)**: Inside the try block, this line attempts to convert the input string (n\_value\_str) into an integer and stores it in the variable n\_value.
4. **if n\_value < 0:**: This checks if the entered integer is negative. If it is, it prints a message asking the user to enter a positive integer.
5. **else:**: If the entered integer is not negative, the code inside this else block is executed.
6. **result = sum\_upto\_n(n\_value)**: This line calls the sum\_upto\_n function (which should have been defined and executed in a previous cell) and passes the user's input (n\_value) as an argument. The value returned by the function (the sum) is stored in the result variable.
7. **print(f"The sum of numbers up to {n\_value} is: {result}")**: This line prints the final output to the user, showing the input value of n and the calculated sum. The f"" syntax is a f-string, which allows you to easily embed the values of variables directly within the string.
8. **print("Invalid input. Please enter an integer.")**: If the int() conversion in the try block fails (because the user entered something that wasn't an integer), this line inside the except block is executed, informing the user that their input was invalid.

A screenshot of a computer program

AI-generated content may be incorrect.

ERROR CODE:

A screenshot of a computer

AI-generated content may be incorrect.

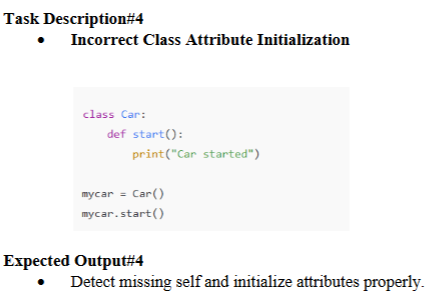
CORRECT CODE:

A screenshot of a computer code

AI-generated content may be incorrect.

EXPLANATION:

1. **class user:**: This line defines a new class named user. Classes are blueprints for creating objects (instances).
2. **def \_\_init\_\_(self, name):**: This defines the constructor method for the user class. The \_\_init\_\_ method is automatically called when you create a new object of the class.
   * self: Refers to the instance of the class being created.
   * name: This is a parameter that will be passed when you create a user object.
3. **self.name = name**: Inside the constructor, this line takes the name value that was passed in and assigns it to an attribute of the object called name. Each user object will have its own name attribute.
4. **u = user("alice")**: This line creates a new instance (object) of the user class and assigns it to the variable u. When creating the object, the string "alice" is passed as the name argument to the \_\_init\_\_ method. So, the u object will have its name attribute set to "alice".
5. **print(u.name)**: This line accesses the name attribute of the u object using dot notation (u.name) and prints its value to the console.



ERROR CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

CORRECT CODE:

A screen shot of a computer program

AI-generated content may be incorrect.

EXPLANATION:

1. **class car:**: This line defines a new class named car. Classes are blueprints for creating objects (instances) that share common properties and behaviors.
2. **def start(self):**: This defines a method within the car class called start.
   * self: This is a required parameter in class methods that refers to the instance of the class on which the method is being called. It allows the method to access and modify the instance's attributes and other methods.
   * print("car started"): This line inside the start method simply prints the string "car started" to the console when the method is called.
3. **mycar = car()**: This line creates a new instance (object) of the car class and assigns it to the variable mycar. This is how you create an actual car object based on the car blueprint.
4. **mycar.start()**: This line calls the start() method on the mycar object. Since mycar is an instance of the car class, it can use the methods defined within that class. When this line is executed, the code inside the start method runs, printing "car started".

A screenshot of a computer program

AI-generated content may be incorrect.

ERROR CODE:

A screen shot of a computer code

AI-generated content may be incorrect.

CORRECTED CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

EXPLANATION:

1. **score\_str = input("Please enter the student's score: ")**: This line prompts the user to enter the student's score and stores the input as a string in the score\_str variable.
2. **try...except ValueError:**: This is a try-except block for error handling. It attempts to execute the code within the try block. If a ValueError occurs (e.g., if the user enters text that cannot be converted to a number), the code in the except ValueError: block is executed.
3. **score = float(score\_str)**: Inside the try block, this line attempts to convert the input string score\_str into a floating-point number (allowing for decimal scores) and stores it in the score variable.
4. **grade = grade\_student(score)**: This line calls the grade\_student function (which you defined earlier) and passes the converted score as an argument. The value returned by the function (the letter grade) is stored in the grade variable.
5. **print(f"A student with a score of {score} gets a grade of: {grade}")**: This line prints the final output, showing the entered score and the calculated grade.
6. **print("Invalid input. Please enter a valid number for the score.")**: If the float() conversion in the try block fails, this line inside the except block is executed, informing the user that their input was invalid.