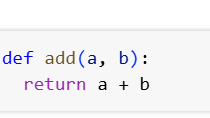
Assignment – 7.3

Task – 1 :



The function you provided was a syntax error because there was a missing colon (:)at the end of the def add(a,b)line . in python ,function definatioons must end with a colon .

Explanation:

**Cell PiFpwgf3nAsq:** This cell defines a simple Python function called add that takes two arguments, a and b.

1. def add(a, b):: This line defines a function named add that accepts two parameters, a and b.
2. return a + b: This line calculates the sum of a and b and returns the result.

**Cell a0bf1822:** This cell attempts to define the same add function but has a syntax error.

1. def add(a, b): This line attempts to define a function named add but is missing the colon : at the end, which is required in Python function definitions.
2. return a + b: This line is intended to calculate and return the sum, but the syntax error on the previous line prevents this line from being executed.

**Cell 8ed59adc:** This cell contains a while loop that was causing an infinite loop and has been corrected.

1. i = 0: This line initializes a variable i to 0. This variable is used as a counter for the loop.
2. while i < 5:: This line starts a while loop. The loop will continue to execute as long as the condition i < 5 is true.
3. print(i): This line prints the current value of the variable i.
4. i += 1: This line increments the value of i by 1 in each iteration. This is the fix for the infinite loop, as it ensures that i will eventually become 5 and the loop condition i < 5 will become false, terminating the loop.

Task-2:

A computer screen shot of a computer code

AI-generated content may be incorrect.

I see the issue in the count\_down function. The loop condition n>=0 and the increment n+=1 will cause an infinite loop. I'll change n+=1 to n-=1 so the loop counts down and eventually terminates.

Explanation:

1. def count\_down\_error(n):: This line defines a function named count\_down\_error that takes one argument, n.
2. while n > 0:: This line starts a while loop. The code inside this loop will continue to execute as long as the condition n > 0 is true.
3. print(n): This line prints the current value of the variable n to the console.
4. n -= 1: This line decrements the value of n by 1 in each iteration of the loop. This step is crucial for the loop to eventually terminate, as it makes n smaller with each iteration until it is no longer greater than 0.

This function, when called with a positive integer n, will print the numbers from n down to 1.

Task-3:

A screen shot of a computer error

AI-generated content may be incorrect.

I can help with that. I'll add the code that causes a division by zero error and then modify it to include a try-except block for safe error handling.

I will now modify the code to include a try-except block to handle the ZeroDivisionError.

Explanation:

1. def divide(a, b):: This line defines a function named divide that takes two arguments, a and b.
2. try:: This keyword starts a try block. The code within the try block is attempted to be executed. If an error occurs during the execution of the code in the try block, Python will look for a corresponding except block to handle the error.
3. return a / b: This line is inside the try block. It attempts to perform the division of a by b and return the result.
4. except ZeroDivisionError:: This line starts an except block that specifically catches the ZeroDivisionError. This error occurs when you try to divide a number by zero. If a ZeroDivisionError happens in the try block, the code inside this except block will be executed.
5. print("Error: Cannot divide by zero."): This line is inside the except block. It prints an error message to the console when a ZeroDivisionError is caught.
6. return None: This line is also inside the except block. It returns the value None when a ZeroDivisionError occurs. You could choose to return a different value or perform another action here depending on what makes sense for your program.
7. print(divide(10, 0)): This line calls the divide function with a=10 and b=0 and prints the return value of the function. Since b is 0, a ZeroDivisionError will occur, the except block will be executed, and the function will return None, which will then be printed.

This code demonstrates how to use a try-except block to gracefully handle potential errors, like division by zero, preventing the program from crashing.

Task – 4:

A screenshot of a computer

AI-generated content may be incorrect.

I see the issues in the rectangle class definition. The \_\_init\_\_ method is missing the self parameter, and there's a typo in the assignment for self.length. I'll add self to the parameters and correct self.length =- length to self.length = length. Also, the method name should be \_\_init\_\_ (with two underscores on each side).

I have fixed the rectangle class definition by adding self to the \_\_init\_\_ method parameters and correcting the assignment.

Explanation:

1. class rectangle():: This line defines a new class named rectangle. In Python, classes are blueprints for creating objects. An object created from this class will represent a rectangle.
2. def \_\_init\_\_(self, length, width):: This line defines a special method within the rectangle class called \_\_init\_\_. This is the constructor method. It is automatically called when you create a new object (an instance) of the rectangle class.
   * self: This is the first parameter of any instance method in a class. It refers to the instance of the class itself (the specific rectangle object being created).
   * length: This is a parameter that will receive the value for the length of the rectangle when you create a new rectangle object.
   * width: This is a parameter that will receive the value for the width of the rectangle when you create a new rectangle object.
3. self.length = length: This line is inside the \_\_init\_\_ method. It takes the value passed as the length argument when the object is created and assigns it to an attribute of the object itself. self.length means the length attribute of the current instance (self).
4. self.width = width: Similar to the previous line, this line takes the value passed as the width argument and assigns it to the width attribute of the current rectangle instance (self.width).

Task – 5:

A screenshot of a computer

AI-generated content may be incorrect.

Error: The index you are trying to access is out of the list's range.

Explanation:

1. numbers = [1, 2, 3]: This line creates a Python list named numbers and initializes it with the elements 1, 2, and 3. The list has indices 0, 1, and 2.
2. try:: This keyword starts a try block. The code within this block is where you put the code that might potentially cause an error. Python will attempt to execute the code inside the try block.
3. print(numbers[5]): This line is inside the try block. It attempts to access and print the element at index 5 of the numbers list. Since the list only has elements up to index 2, this operation will cause an IndexError.
4. except IndexError:: This line starts an except block. It specifies that if an IndexError occurs in the preceding try block, the code within this except block should be executed.
5. print("Error: The index you are trying to access is out of the list's range."): This line is inside the except block. Since the IndexError occurred in the try block, this line is executed, printing the error message to the console.