Richard Hayes Crowley

06/09/2021

CSC\_242\_Lab\_02 Q&A

1. In the start code for this application, what is the purpose of the implementation of this keyword? **Global**

**The global keyword is used to indicate that a variable within a function is to reference a variable declared in the global (or, modular namespace’s) scope. So when changes are made to the variable within a function, those changes are also made to the global variable declared outside of the function’s scope. That way, the updated variable value would be accessible throughout the module by other functions.**

1. Both of the functions defined in the starter code, namely **getCustomerData()** and **calculateElecticBill()** , were parameter - less. But one of these functions had a **return** statement. Which of these functions had the **return** statement and why?

**In the starter code, only the calculateElectricBill had a return statement, and I assume this is because there was no global variable(s) initialized for the function to mutate and for other functions to read. I elected to return dictionaries from my functions and read the mutated values from there, however if I were to use global variables I would have declared the variables “total”, “penalty”, and “surcharge” in the outer scope and re-declared them with the *global* keyword within my functions.**

1. Which of these logical operators ( **not** , **and** , **or** ) did you use in your program code?

**I only needed to use comparison operators (.e.g, greater than or equal to >= etc.) in my program code to meet the assignment’s acceptance criteria. However, I could use a conditional statement such as “if surcharge in billDictionary” or “if penalty in billDictionary” to determine whether or not to print a surcharge or penalty in my program’s output, as those values may be zero.**

1. In the Python programming language, are these two **if()** statement logical conditions equivalent? Explain your answer.

**x = 5**

**if (x > 4 and x < 10 ) :**

**print ("success")**

**if (4 < x < 10) :**

**print ("success")**

**Yes, these two conditional statements are equivalent because they measure the same two things (that x is greater than four and less than ten) *inclusively*. If the first statement said “or” instead of “and”, then that would be an exclusive statement. But in this case, the results are the same (prints “success” twice if you run it).**

1. New versions of Python now allow the use of a **match** - **case** statement. An example of such a statement is shown below?

**def Departments(val) :**

**code = val**

**match code :**

**case 10 :**

**print ("Accounting")**

**case 20 :**

**print ("Finance")**

**case \_ :**

**print("other department")**

**Departments(30)**

Which would be better to use in our program, for computing the Electricity charge, a **match** - **case** statement or an **if** - **elif** - **else** statement? Explain your answer.

**A match case (or a “switch case” if you were talking to a JavaScript person) is useful when you have large number of conditions and want to avoid writing a bunch of if-elif statements. It is considered a “cleaner” way to handle a large number of conditional statements.**

**In our application, I don’t think there’s much benefit to using a match case because we are only using a few if-elif statements (in my case, I’m only using four, e.g., if-elif-elif-else), and it isn’t very cluttered. However if I had a couple more conditions to test for, I would reach for a match-case since it is easier to read and is cleaner code.**