Carleton University
Department of Systems and Computer Engineering
ECOR 1041 — Computation and Programming — Winter 2021

Lab 5 - Control Flow: Making Choices.

Objective

To gain experience: (1) developing Python functions that use `if` statements to perform different computations, based on conditions, and (2) using Boolean operators to form "complex" conditions.

Submitting Lab Work for Grading

You don't have to finish the lab by the end of your lab period. The deadline for submitting lab work to cuLearn for grading is 11:55 pm (Ottawa time) the day after your scheduled lab. Solutions that are emailed to your instructor or a TA will not be graded, even if they are emailed before the deadline.

Please read *Important Considerations When Submitting Files to cuLearn*, on the last page of the course outline.

Exercise 1 (Based on an exercise by Cay Horstmann and Rance Necaise)

Calculating the tip when you go to a restaurant is not difficult, but a restaurant wants to suggest a tip based on the diners' satisfaction with the level of service they receive. In this exercise, you'll use the function design recipe from Chapter 3 of *Practical Programming* to develop a function named `tip`. The function has two parameters. The first argument is the cost of the meal. The second parameter is satisfaction level. (Use these ratings: 1 = Totally satisfied, 2 = Somewhat satisfied, 3 = Dissatisfied). The function returns the amount of the tip, calculated as follows:

- If the diners are totally satisfied, calculate a 20% tip.
- If the diners are somewhat satisfied, calculate a 15% tip.
- If the diners are dissatisfied, calculate a 5% tip.

Step 1: Launch Wing 101. Create a new editor window and save it. Use lab5ex1.py as the file name.

Step 2: Type this function header and empty docstring.

```
def tip():
    """
    """
```

Step 3 (Examples): In the docstring, type two or three example function calls and return values. Use a calculator to determine the value the function should return for a given set of arguments.

Step 4 (Header): Add parameters (use descriptive names) and type annotations to the function header.

Step 5 (Description): In the docstring, write a sentence or two that describes what the function does (but not how it does it). Mention the parameters in your description and describe the return value. Remember to describe any preconditions on the parameter values.

Step 6 (Body): Design and code the function body. Hint: you don't need to use the Boolean operators (`and`, `or` and `not`).

Save the code, then click Run. Correct any syntax errors.

Step 7 (Test): Use the Python shell to test your function (use the docstring examples). After you've finished testing, close the editor window for lab5ex1.py.

Exercise 2 (Based on an exercise by Cay Horstmann and Rance Necaise)

A supermarket awards coupons depending on how much a customer spends on groceries. For example, if you spend $50, you'll get a coupon worth 8% of that amount ($4). The following table shows the percentage used to calculate the coupon awarded for different amounts spent:

Amount Spent	Coupon Percentage
Less than $10	No coupon
Between $10 to $60	8%
More than $60 to $150	10%
More than $150 to $210	12%
More than $210	14%

Step 1: Create a new editor window and save it. Use lab5ex2.py as the file name.

Step 2: Use the function design recipe (examples, header, description, body, test) to develop a function named `coupon`. The function has one parameter, which is the amount spent on groceries. It returns the value of the coupon. Hint: you don't need to use the Boolean operators (`and`, `or` and `not`).

Your function definition must have type annotations and a complete docstring. Use the Python shell to test your function. After you've finished testing, close the editor window for lab5ex2.py.

Exercise 3 (Based on an exercise by Cay Horstmann and Rance Necaise)

A non-governmental organization needs to calculate the amount of financial assistance it will provide to needy families. The formula is as follows:

- If the annual household income is at least $30,000 but below $40,000 and the household has at least three children, the amount is $1,500 per child.
- If the annual household income is at least $20,000 but below $30,000 and the household has at least two children, the amount is $1,000 per child.
- If the annual household income is below $20,000, the amount is $2,000 per child.

Step 1: Create a new editor window and save it. Use lab5ex3.py as the file name.

Step 2: Use the function design recipe (examples, header, description, body, test) to develop a function named `assistance`. The function has two parameters. The first parameter is the household income. The second parameter is the number of children in the household. It returns the amount of assistance the family should receive. Hint: a function that uses Boolean operators (`and`, `or` and `not`) will be much shorter than one that does not.

Your function definition must have type annotations and a complete docstring. Use the Python shell to test your function. After you've finished testing, close the editor window for lab5ex3.py.

Exercise 4

When bakers get together for a party, they like to eat pastries. A bakers' party is successful when the number of pastries is between 40 and 60, inclusive. Unless it is the weekend, in which case there is no upper bound on the number of pastries.

Step 1: Complete this truth table (fill in the values in the "Is this party successful?" column).

Is it the weekend?	Number of pastries	Is the party successful?
true	Below 40	
true	Between 40 and 60	
true	Above 60	
false	Below 40	
false	Between 40 and 60	
false	Above 60	

Step 2: Create a new editor window and save it. Use lab5ex4.py as the file name.

Step 3: Use the function design recipe to develop a function named `bakers_party`. The function takes two arguments. The first argument is `True` if it's the weekend, `False` if the day is

a weekday. The second argument is the number of pastries (an integer). The function returns `True` if a party with the given arguments is successful, otherwise it returns `False`.

Use the truth table to help you create example function calls and design the function body.

Your function definition must have type annotations and a complete docstring. After you've finished testing, close the editor window for lab5ex4.py.

Exercise 5

The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is between 60 and 90 degrees F (inclusive). Unless it is summer, then the upper limit is 100 degrees instead of 90.

Step 1: Build a truth table that summarizes whether squirrels are playing, based on the season and the temperature. The table header is:

Is it summer?	Temperature	Are the squirrels playing?

Step 2: Create a new editor window and save it. Use lab5ex5.py as the file name.

Step 3: Use the function design recipe to develop a function named `squirrel_play`. This function takes two arguments. The first argument is a boolean value that specifies the season (`True` if it's summer, otherwise `False`.) The second argument is a temperature (an integer). The function returns `True` if the squirrels are playing, given the temperature and the season; otherwise it returns `False`.

Use the truth table to help you create example function calls and design the function body.

Your function definition must have type annotations and a complete docstring. Use the Python shell to test your function. After you've finished testing, close the editor window for lab5ex5.py.

Exercise 6

Any fan of the late author Douglas Adams (*The Hitchhiker's Guide to the Galaxy*) knows that 42 is a truly great number.

Step 1: Create a new editor window and save it. Use lab5ex6.py as the file name.

Step 2: Use the function design recipe to develop a function named `great_42`. This function takes two integer values, `a` and `b`. It returns `True` if either value is 42, or if their sum or the absolute value of their difference is 42.

Before you write any code, build a truth table. The table header is:

a	b	Returned value

Use the table to help you create example function calls and design the function body.

Hint 1: the built-in function `abs(x)` computes the absolute value of `x`.

Hint 2: the shortest solution has no `if` statement and instead has a single statement:

```
return expression
```

(*expression* will contain Boolean operators).

Your function definition must have type annotations and a complete docstring. Use the Python shell to test your function. After you've finished testing, close the editor window for lab5ex6.py.

Exercise 7

Step 1: Create a new editor window and save it. Use lab5ex7.py as the file name.

Step 2: Use the function design recipe to develop a function named `blackjack`. This function takes two positive integer values, `a` and `b`. The function returns whichever value is closest to 21 without being over 21. It returns 0 if both values are over 21. For example, if the values of `a` and `b` are 19 and 20, the function returns 20. If the values of `a` and `b` are 19 and 21, the function returns 21. If the values of `a` and `b` are 17 and 22, the function returns 17.

Your function definition must have type annotations and a complete docstring. Use the Python shell to test your function. After you've finished testing, close the editor window for lab5ex7.py.

Exercise 8

Step 1: Create a new editor window and save it. Use lab5ex8.py as the file name.

Step 2: Use the function design recipe to develop a function named `multiply_uniques`. This function takes three integer values, `a`, `b` and c, and returns their product. However, if one of the values is the same as another of the values, that value is not used when the product is calculated. For example, if the values of `a`, `b` and `c` are 3, 2 and 3, respectively, the product is 2 (the two 3's aren't used). If the values of `a`, `b` and `c` are 3, 3 and 3, the product is 0 (the three 3's aren't used).

Your function definition must have type annotations and a complete docstring. Use the Python shell to test your function. After you've finished testing, close the editor window for lab5ex8.py.

Extra Practice

Use PyTutor to visualize the execution of your solutions. Remember, PyTutor doesn't have a shell. After copying your function definitions into the PyTutor editor, type assignment statements that call the functions and assign the returned values to variables.

Wrap Up

Please read *Important Considerations When Submitting Files to cuLearn*, on the last page of the course outline. The submission deadlines for this lab are:

Lab Section	Lab Date/Time	Submission Deadline (Ottawa Time)
3A	Monday, 8:35 - 11:25	Tuesday, Feb. 9, 2021, 23:55
2A	Tuesday, 14:35 - 17:25	Wednesday, Feb. 10, 2021, 23:55
1A	Friday, 12:35 - 14:25	Saturday, Feb. 13, 2021, 23:55
4A	Friday, 18:05 - 20:55	Saturday, Feb. 13, 2021, 23:55

1. Login to cuLearn. Go to the cuLearn page **for your lab section** (not the main course page).
2. Click the link Submit Lab 5 for grading.
3. Click Add submission.
4. Submit your solutions by dragging the eight files containing your functions (lab5ex1.py through lab5ex8.py) to the File submissions box.
5. Note that you have to confirm that your solutions are your own work before you can submit them. Click Save changes. The Submissions status page will be displayed. The line labelled *Submission status* will confirm that your submission is now *Submitted for grading*. The line labelled *File submissions* will list the names of the files you've submitted.
6. You are permitted to make changes to your solutions and resubmit the files as many times as you want, up to the deadline. Only the most recent submission is saved by cuLearn. To submit revised solutions, follow steps 7 through 9.
7. In cuLearn, click the link Submit Lab 5 for grading.
8. Click Remove submission to delete your previous submission. Click Continue to confirm that you want to do this.
9. Click Edit submission. Repeat steps 4 and 5 to submit your revised solutions.

History: Feb. 5, 2021 (initial release)