CS 1110 Lab 3 – Spring 2024

Lab Introduction

For this lab and future labs, you will be working with a partner. When you and your partner have completed the lab, notify a TA or instructor, who will ask you questions about your lab and check you both off as completed. Turn in a copy of your lab03.py file on Gradescope to the Lab03 assignment. Lab grades are based on putting forth a good effort and working with a partner.

Lab Task

1. More on Functions
2. Personal Budgeting
3. Convert Temperature
4. Conceptual Questions

More on Functions

Arguments vs Parameters

When using functions in code, one common misunderstanding is that parameters and arguments are synonyms. While they are used in similar contexts, the two are not the same thing. The key difference in arguments and parameters is that arguments are the values passed into a function when it is **called**, and parameters are the variables defined in a function's header when it is **created**. In the example shown, 5 and 6 are arguments, while "first_number" and "second_number" are parameters. Note that arguments can be variables, they just have to be used in the function call rather than the definition.

```python
def add_two_numbers(first_number, second_number):
    return first_number + second_number
```

```python
add_two_numbers(5,6)
```

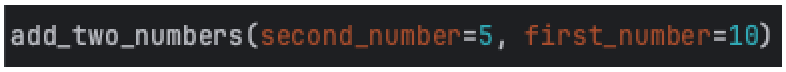
Default Parameters

Default parameters, also known as optional parameters, are parameters that have a value that they will default to if nothing is passed in for them when a function is called. To use a default parameter, assign a parameter a value with the assignment operator (=) in your function definition (ex: 'second_number').

```python
def add_two_numbers(first_number, second_number=10):
    return first_number + second_number
```

Named Arguments

Named arguments, also known as keyword arguments, can be used to override the order in which parameters are declared in a function definition. Standard function calls use *positional arguments*, which are arguments that are set equal to the parameter whose position in the functional definition corresponds to their position in the function call. Named arguments are arguments where you explicitly tell the computer which parameter an argument is meant to correspond to. Be careful when using these; unnamed arguments will still be positional, meaning that you could assign multiple values to a parameter if you are not careful. Finally, similarly to default parameters, named arguments must come **after** positional (unnamed) arguments in the function call.



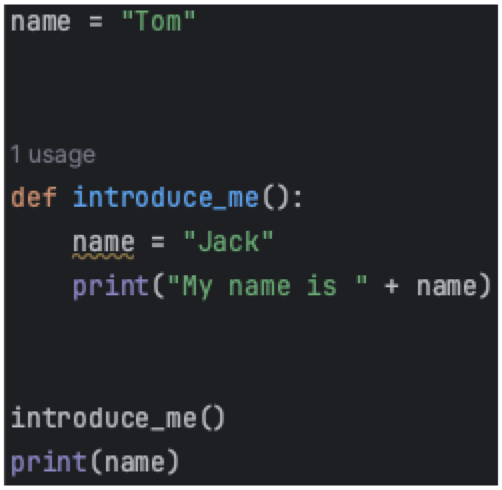
```python
add_two_numbers(second_number=5, first_number=10)
```

Scope

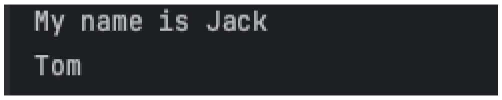
In Python, variables are only available for modification from within the "region" in which they are created. This region is known as the variables scope. Variables created in the main body of the code (not in a function) are known as global variables, and they can be accessed (but not necessarily modified) from anywhere in the code. Every time a function is defined, it is given its own local scope. Variables created inside of functions are known as local variables and can only be accessed within the function (their scope) for any purpose.

The Global Keyword

The 'global' keyword is a keyword used in python that allows users to modify a variable from the global scope inside the function. If a user wants to access a variable from the global scope inside their function, they can do so without using the global keyword, but they will be unable to change the value of this variable in their function. Additionally, their function cannot have a variable of the same name defined within it because Python will assume that the user is referring to the variable in the local scope. In the following example, the function uses the name "Jack" from the local scope, but when name is printed after the function has been called, it is still "Tom". The name variable cannot be changed from within a function without using the 'global' keyword. To use the global keyword in python, simply write 'global' and then the variable or variables that you will be using as globals in your function.



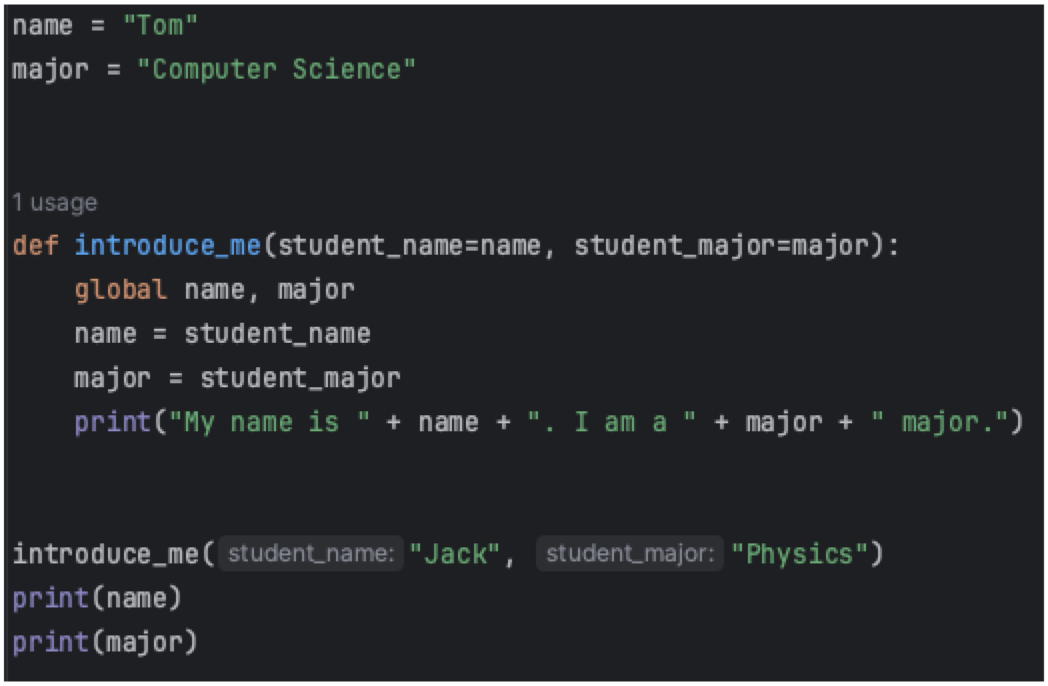
```python
name = "Tom"


1 usage
def introduce_me():
    name = "Jack"
    print("My name is " + name)


introduce_me()
print(name)
```



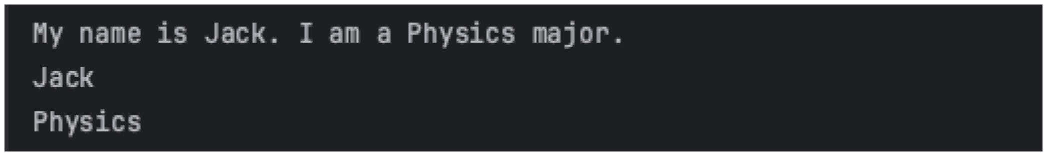
```
My name is Jack
Tom
```

If there are multiple variables that you would like to declare as 'global' variables, separate them with a comma (ex: 'global name, age, year'). You technically do not have to declare a variable as global until the line before you use it, but you should declare all of the global variables you are using at the top of your function to make your code easier to read.

In the following example, we put together many concepts that have been talked about above. The function 'introduce_me' takes in two default parameters, 'student_name' and 'student_major', both of which default to the values stored in the global scope. Inside the function, the global variables, which were declared in the first line of the function, will be assigned the values passed in as arguments (resulting in no change if no arguments were passed), and an introduction will be printed to the console. After calling the function, the values for name and major are printed to show that they were changed by the function.



```
name = "Tom"
major = "Computer Science"


1 usage
def introduce_me(student_name=name, student_major=major):
    global name, major
    name = student_name
    major = student_major
    print("My name is " + name + ". I am a " + major + " major.")


introduce_me( student_name: "Jack",  student_major: "Physics")
print(name)
print(major)
```



```
My name is Jack. I am a Physics major.
Jack
Physics
```

Lab Activity: Personal Budgeting

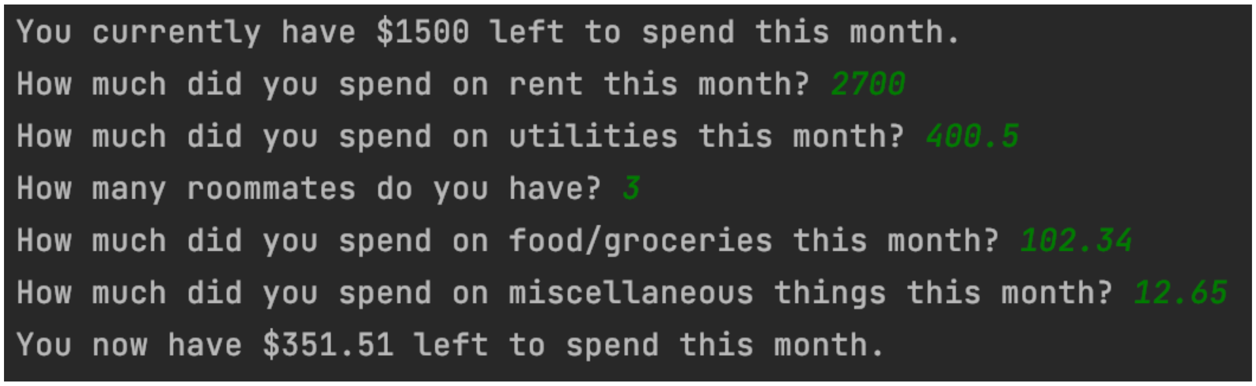
You are looking to write a budget function to keep track of all of your expenses in college. To start, you should make two global variables: one to hold your total monthly budget ('personal_budget') and one to track the remaining funds as you spend money ('remaining_budget'). Now, create three functions to keep track of your budget. The functions are described below:

1) Call the first function 'subtract_from_budget'. It should take in one parameter, your new expense, and subtract this from your global variable tracking the remaining budget. It should not return anything.
2) The second function will deal with our shared bills (rent, utilities, etc.). Call it 'divide_costs'. This function will have two parameters, one required and one optional. The first parameter is the total cost of the bill, and the second optional parameter is the amount of roommates or people you are splitting the bill with. You should set the default value to 1 because if you do not split the bill with anyone, the function should just return

the cost of the bill. For example, if the rent on your apartment is $2,400 split amongst three roommates, you would only pay $800.

3) The third function should be called ‘pay_bills’. In this function, use the input function to prompt the user to input the amount they spent on certain categories (ex: rent, utilities, food, etc.). This function should call the previous functions (‘subtract_from_budget’ and ‘divide_costs’) to track all of the expenses. Finally, print out the user’s remaining amount of money for the month.

Here is an example:



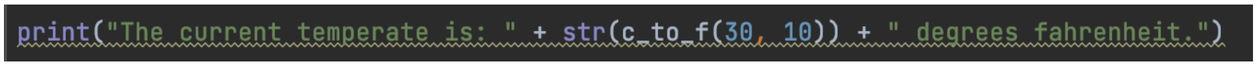
```
You currently have $1500 left to spend this month.
How much did you spend on rent this month? 2700
How much did you spend on utilities this month? 400.5
How many roommates do you have? 3
How much did you spend on food/groceries this month? 102.34
How much did you spend on miscellaneous things this month? 12.65
You now have $351.51 left to spend this month.
```

Lab Activity: Convert Temperature

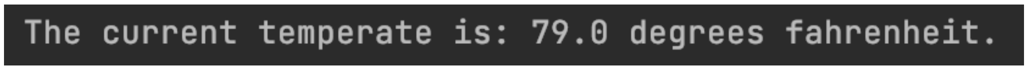
Now, you are given an assignment to convert temperatures from Celsius to Fahrenheit. Many factors contribute to the temperature we feel, but today we will focus on actual temperature and wind chill. Make a function called ‘c_to_f’ with two parameters. The first parameter should be the temperature in Celsius and is required. The second parameter is the wind speed (in mph) and is optional. If the user does not enter a wind speed, we can assume there is no wind chill, so set the default value to 0.

To convert a Celsius temperature to Fahrenheit, multiply the Celsius value by 9/5, then add 32 to the total. To find the effect wind has on the temperature, multiply the wind speed (in mph) by 0.7 and subtract this value from the temperature.

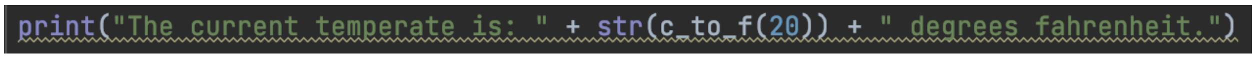
Here are two examples:



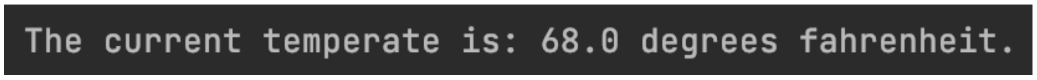
```
print("The current temperate is: " + str(c_to_f(30, 10)) + " degrees fahrenheit.")
```



```
The current temperate is: 79.0 degrees fahrenheit.
```



```
print("The current temperate is: " + str(c_to_f(20)) + " degrees fahrenheit.")
```



```
The current temperate is: 68.0 degrees fahrenheit.
```

Challenge

- Call your ‘pay_bills’ function multiple times. What is happening to your ‘personal_budget’ and ‘remaining_budget’ variables? Why?
- What happens when you use an input function within your ‘c_to_f’ function *and* call the function with a parameter value. Which value gets used?

Conceptual Questions

- What is the difference between required and default parameters?
- Why are default parameters helpful? In what situations are they useful?
- What is the difference between named and positional arguments?
- What is the difference between parameters and arguments?
- How can you tell if the variable is the local scope? What about global scope?
- When is the “global” keyword necessary to use?