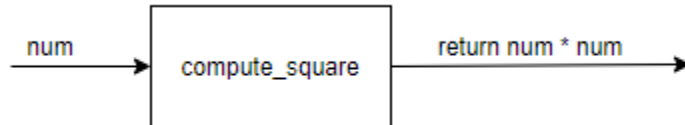


CS128 Lab 8 – Functions that return values

Most functions require parameters, which are values that control how the function does its job. A function can also return a value. For example, the **compute_square** function takes a value as a parameter and returns the value squared.



When you call a function that returns a value, your code should do something with the value such as assign it to a variable, or pass it as an argument to another function.

```
def compute_square(num):  
    return num * num  
  
#main algorithm  
  
num_squared = compute_square(7)  
print('7 squared is', num_squared)  
  
print('20 squared is', compute_square(20))
```


In the example code, notice the first call to **compute_square** is on the right hand side of an assignment statement, which means the value returned from the function is stored in the variable **num_squared**.

The second call to **compute_square** is embedded as an argument to a call to the **print** function.

Note

A function *terminates immediately* after the execution of a **return** statement. You should not have code after the return statement, since it will not be executed.

```
def compute_square(num):  
    return num * num  
    print("this line is never reached")
```



You can have more than one return statement in a function, although each should be the last statement in the control flow block.

```
def get_largest( num1 , num2):  
    if num1 > num2:  
        return num1  
    else:  
        return num2
```

You should not have code that follows either return statement:

```
def get_largest( num1 , num2):  
    if num1 > num2:  
        return num1  
        print("this line is never reached") ← error  
    else:  
        return num2  
        print("this line is never reached") ← error
```

You should not have code after the if/else in the following example since both branches contain a return statement.

```
def get_largest( num1 , num2):  
    if num1 > num2:  
        return num1  
    else:  
        return num2  
  
    print("this line is never reached") ← error
```

TASK1: You will not write code for this task. Answer the questions related to the sample functions. You might want to use scratch paper to record values (a, b, c, num) while working on this task. **DO NOT** TYPE THE FUNCTION INTO CODIO OR PYTHON TUTOR.

```
def mystery(a, b, c):  
    num = a + b + c  
    if a < b :  
        num = num * 2  
    elif b < c :  
        num = num * 5  
    else:  
        num = num * 10  
  
    return num
```

QUESTION 1: What value is stored in **val1** after the following code is executed?

val1 = mystery(6, 4, 2)

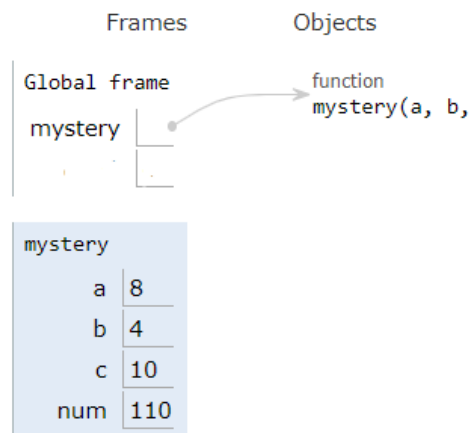
QUESTION 2: What value is stored in **val2** after the following code is executed?

val2 = mystery(3, 2, 5)

Assume the code shown below in the Python Visualizer. The execution is currently at line 10 and is about to return the function result. Notice 3 separate calls to the mystery function (lines 13, 15, 17).

Python 3.6
([known limitations](#))

```
1 def mystery(a, b, c):  
2     num = a + b + c  
3     if a < b :  
4         num = num * 2  
5     elif b < c :  
→ 6         num = num * 5  
7     else:  
8         num = num * 10  
9  
→ 10    return num  
11  
12  
13 tmp1 = mystery( 4, 8, 10)  
14  
15 tmp2 = mystery(8, 4, 10)  
16  
17 tmp3 = mystery(10, 4, 8)
```



QUESTION 3: Look at the function frame on the right hand side of the screen print. What value will be returned when line 10 executes?

QUESTION 4: Which global variable (tmp1, tmp2 or tmp3) is assigned the value returned by the mystery function after line 10 executes? If you look at the function frame parameter values for a, b, and c, you should be able to determine which of the 3 calls was made.

QUESTION 5: What is printed when the following code executes?

```
def mystery2(a, b, c):  
    if a < b and b < c:  
        return a  
    elif b < a and b < c:  
        return b  
    elif c < a and c < b:  
        return c  
    else:  
        return -1  
  
print("test#1:", mystery2( 6, 4, 8) )  
print("test#2:", mystery2( 5, 7, 5) )
```

```
def mystery3(a, b):  
  
    if a == b:  
        result = "equal"  
    elif a < b:  
        result = "less than"  
    else:  
        result = "greater than"  
  
    return result  
    print('a{} {} b{}'.format(a, result, b))  
  
compare_string = mystery3(5,7)
```

QUESTION 6: What is printed when the code executes?

QUESTION 7: Describe the error in the **mystery3** function.

GET A SIGNATURE FOR TASK1.

In the remainder of this lab you will create functions to calculate the cost of doing a room renovation.

Task 2: Write a function to convert a distance given in feet and inches to a distance in feet only. For example: 3 feet 6 inches would result in 3.5 feet. You must use the function signature provided below.

```
def convert_to_feet (feet, inches) :
```

After you write the **convert_to_feet** function, run the cell to confirm the function results:

```
7 feet 0 inches = 7.0 feet
5 feet 6 inches = 5.5 feet
12 feet 9 inches = 12.75 feet
```

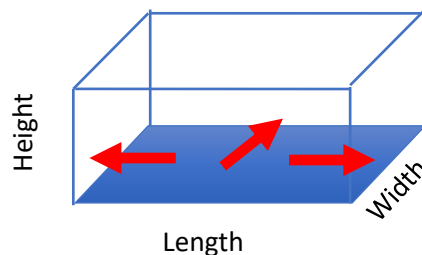
GET A SIGNATURE FOR TASK2.

Task 3: Write a method called **calc_ceiling_space** that takes the length and width of a rectangular room as parameters and returns the area of the ceiling. Run the cell to confirm the expected result matches the actual result returned by the **calc_ceiling_space** function:

```
24.0 length 10.0 width = 240.0 ceiling space
7.5 length 11.5 width = 86.25 ceiling space
```

GET A SIGNATURE FOR TASK3.

Task 4: Write a function called **calc_wall_space** that determines all the wall space in a rectangular room. The three dimensions (length, width, and height) are given as parameters to the function. The function should return a single value that specifies the total wall space in the room.



There are four walls that need to be painted in this room. It is the area of those four walls that the method should calculate. Three of the walls are indicated by arrows (the left and right smaller walls, as well as the larger back wall). The fourth wall is the front wall, which is the same size as the back wall.

For example, **calc_wall_space (15.0, 12.0, 10.0)** is calculated from 2 walls of 15 length by 10 height, each is 150 sq.ft. for a total of 300 sq. ft. and 2 walls that are 12 width by 10 height, each is 120 sq. ft. for a total of 240 sq. ft. Added together, this gives 540 sq. ft. Confirm the expected result matches the actual result returned by the **calc_wall_space** function:

```
15.0 length 12.0 width 10.0 height = 540.0 wall space
11.75 length 10.5 width 8.0 height = 356.0 wall space
```

GET A SIGNATURE FOR TASK4.

Task 5: Write a function called **calc_room_space** that takes in the length, width, and height of a room as parameters and calculates the total area that needs painting. (Assume that the ceiling and the walls will be painted the same color). The **calc_room_space** function should directly call the **calc_ceiling_space** and **calc_wall_space** functions with the appropriate values. Do not duplicate the ceiling and wall space calculations in the **calc_room_space** function body, call the existing functions to do the work for you.

NOTE: A Python cell may refer to functions defined in other cells as long as you have already run the code to define the functions. You do not need to copy the **calc_ceiling_space** and **calc_wall_space** functions into the same cell as **calc_room_space**, although you can if it makes it easier to understand.

Confirm the expected result matches the actual result returned by the **calc_room_space** function:

```
15.0 length 12.0 width 10.0 height = 720.0 room space
8.5 length 9.6 width 10.7 height = 468.93999999999994 room space
```

GET A SIGNATURE FOR TASK5.

Task 6: Write a function called **paint_amount** that takes as parameters the room length, width and height and determines how many gallons of paint you need to paint the room – including walls and ceiling. Assume that you will cover 350 square feet of wall space with a single gallon of paint. Also note that you cannot buy fractional portions of paint, so this method will have to go up to the next gallon. (If you calculate that you need 1.35 gallons, this method should return 2 gallons.)

HINT: The **math.ceil()** function rounds a number up. For example, **math.ceil(8.2)** returns 9.

The **paint_amount** function should directly call the **calc_room_space** function with appropriate values.

If you have problems with this task, try using the Python Visualizer to step through your code. Confirm the expected result matches the actual result returned by the **paint_amount** function:

```
58 length 30 width 10 height = 10 paint
16.75 length 19.5 width 10.0 height = 4 paint
```

GET A SIGNATURE FOR TASK6.

Task 7: Finally, put all the pieces together to create a method called **renovaton_cost** that takes as parameters the room dimensions length, width, height in feet and inches, and calculates the total cost of painting the room. The function signature is:

def renovation_cost(length_ft, length_inches, width_ft, width_inches, height_ft, height_inches):

Nobody ever says 13.4166 feet when they mean 13 feet 5 inches, so each room dimension is passed as two values: feet and inches. You will need to call **convert_to_feet** to convert each room dimension in just feet (versus feet and inches), for example converting 13 feet 5 inches to 13.14166 feet. Once you determine the length, width and height in feet, call **paint_amount** to determine the amount of paint needed for the room. Assume paint costs 14.99 a gallon and return the total cost of painting the room.

Confirm the expected result matches the actual result returned by the **renovation_cost** function:

```
12 ft 6 inch length 10 ft 9 inch width 8 ft 6 inch height = 29.98 cost
15 ft 7 inch length 12 ft 5 inch width 10 ft 2 inch height = 44.97 cost
```

GET A SIGNATURE FOR TASK7.