**Do you think you can flesh out your own function? I think you can! Let’s give it a go.**

Flesh out the body of the print\_seconds function so that it prints the total amount of seconds given the hours, minutes, and seconds function parameters. Remember that there are 3600 seconds in an hour and 60 seconds in a minute.

def print\_seconds(hours, minutes, seconds):

print(hours\*3600+minutes\*60+seconds)

print\_seconds(1,2,3)

RunReset

Here is your output:

3723

Another milestone in your learning experience. Well done!

Defining Functions Recap

We looked at a few examples of built-in functions in Python, but being able to define your own functions is incredibly powerful. We start a function definition with the def keyword, followed by the name we want to give our function. After the name, we have the parameters, also called arguments, for the function enclosed in parentheses. A function can have no parameters, or it can have multiple parameters. Parameters allow us to call a function and pass it data, with the data being available inside the function as variables with the same name as the parameters. Lastly, we put a colon at the end of the line.

After the colon, the function body starts. It’s important to note that in Python the function body is delimited by indentation. This means that all code indented to the right following a function definition is part of the function body. The first line that’s no longer indented is the boundary of the function body. It’s up to you how many spaces you use when indenting -- just make sure to be consistent. So if you choose to indent with four spaces, you need to use four spaces everywhere in your code.

**Use the get\_seconds function to work out the amount of seconds in 2 hours and 30 minutes, then add this number to the amount of seconds in 45 minutes and 15 seconds. Then print the result.**

def get\_seconds(hours, minutes, seconds):

return 3600\*hours + 60\*minutes + seconds

amount\_a = get\_seconds(2,30,0)

amount\_b = get\_seconds(0,45,15)

result =amount\_a+amount\_b

print(result)

RunReset

Here is your output:

11715

Well done, you! The code you are writing is getting more and

more complex. Keep going!

Returning Values Using Functions

Sometimes we don't want a function to simply run and finish. We may want a function to manipulate data we passed it and then return the result to us. This is where the concept of return values comes in handy. We use the return keyword in a function, which tells the function to pass data back. When we call the function, we can store the returned value in a variable. Return values allow our functions to be more flexible and powerful, so they can be reused and called multiple times.

Functions can even return multiple values. Just don't forget to store all returned values in variables! You could also have a function return nothing, in which case the function simply exits.

**Ready to try it yourself? See if you can reduce the code duplication in this script.**

In this code, identify the repeated pattern and replace it with a function called month\_days, that receives the name of the month and the number of days in that month as parameters. Adapt the rest of the code so that the result is the same. Confirm your results by making a function call with the correct parameters for both months listed.

# REPLACE THIS STARTER CODE WITH YOUR FUNCTION

def month\_days(month,days):

print(month+" has " + str(days) + " days.")

month\_days("june",30)

month\_days("july",31)

RunReset

Here is your output:

june has 30 days.

july has 31 days.

Nice work! You're getting acquainted with some interesting

coding practices to reduce code duplication.

This function to calculate the area of a rectangle is not very readable. Can you refactor it, and then call the function to calculate the area with base of 5 and height of 6? **Tip:** a function that calculates the area of a rectangle should probably be called rectangle\_area, and if it's receiving base and height, that's what the parameters should be called.

def rectangle\_area(base, height):

area = base\*height # the area is base\*height

print("The area is " + str(area))

rectangle\_area(5,6)

RunReset

Here is your output:

The area is 30

Wonderful! You're learning how to write self-documenting

code, that will be easier to read & reuse.

100%

## Practice Quiz: Functions

**TOTAL POINTS 5**

1.Question 1

This function converts miles to kilometers (km).

1. Complete the function to return the result of the conversion
2. Call the function to convert the trip distance from miles to kilometers
3. Fill in the blank to print the result of the conversion
4. Calculate the round-trip in kilometers by doubling the result, and fill in the blank to print the result

# 1) Complete the function to return the result of the conversion

def convert\_distance(miles):

km = miles \* 1.6 # approximately 1.6 km in 1 mile

return km

my\_trip\_miles = 55

# 2) Convert my\_trip\_miles to kilometers by calling the function above

my\_trip\_km = convert\_distance(my\_trip\_miles)

# 3) Fill in the blank to print the result of the conversion

print("The distance in kilometers is " + str(my\_trip\_km))

# 4) Calculate the round-trip in kilometers by doubling the result,

# and fill in the blank to print the result

round\_trip=(my\_trip\_km\*2)

print("The round-trip in kilometers is " + str(round\_trip))

RunReset

**Correct**

Woohoo! You’ve figured out how to make the functions do what

they need to do, and remembered some things from the

previous lessons. Way to go!.

**1 / 1 point**

2.Question 2

This function compares two numbers and returns them in increasing order.

1. Fill in the blanks, so the print statement displays the result of the function call in order.

Hint: if a function returns multiple values, don't forget to store these values in multiple variables

# This function compares two numbers and returns them

# in increasing order.

def order\_numbers(number1, number2):

if number2 > number1:

return number1, number2

else:

return number2, number1

# 1) Fill in the blanks so the print statement displays the result

# of the function call

smaller, bigger = order\_numbers(100, 99)

print(smaller, bigger)

RunReset

**Correct**

Nice! You remembered how to accept multiple return values

from a function. You’re ready for the next lesson!

**1 / 1 point**

3.Question 3

What are the values passed into functions as input called?



Variables



Return values



Parameters



Data types

**Correct**

Nice job! A parameter, also sometimes called an argument, is a value passed into a function for use within the function.

**1 / 1 point**

4.Question 4

Let's revisit our lucky\_number function. We want to change it, so that instead of printing the message, it returns the message. This way, the calling line can print the message, or do something else with it if needed. Fill in the blanks to complete the code to make it work.

def lucky\_number(name):

number = len(name) \* 9

msg = "Hello " + name + ". Your lucky number is " + str(number)

return msg

print(lucky\_number("Kay"))

print(lucky\_number("Cameron"))

RunReset

**Correct**

Way to go! The function now returns the message, for the

calling line to use it in any way it wants to.

**1 / 1 point**

5.Question 5

What is the purpose of the def keyword?



Used to define a new function



Used to define a return value



Used to define a new variable



Used to define a new parameter

**Correct**

Awesome! When defining a new function, we must use the def keyword followed by the function name and properly indented body.