FUNCTIONS

Objectives:

- Why functions exist
- What happens when we enter and leave functions
- Parameters and return values
- How to create and use a function

As you know, a computer program is a list of instructions for the computer to perform in order to solve some problem. But, what happens if we have “a lot” of instructions? It’s just like what happens when any business or group of people has a lot of work to do – they get organized and delegate the work. As another analogy, a program contains functions for the same reason that an essay contains paragraphs.

A restaurant kitchen is a busy place. Some chefs specialize on just one type of food, such as sauces or desserts. A menu is often a recipe of smaller recipes. The same thing occurs in the software industry. A large computer program is written by several people, and each team member is responsible for writing a portion.

Functions provide us a way to *organize a solution into logical pieces* that communicate with each other. A function is basically one self-contained part of a computer program. Over the years, computer scientists have used various words to describe this concept: besides the word “function” you may also hear of a “procedure,” “sub-program,” or “method.” They all essentially mean the same thing, and the nuances are not important at this point.

Another purpose of a function in a computer program is to encapsulate some code that might need to be used several times. We *avoid having to retype the same code* in more than one place in the program.

Here is an analogy to illustrate: When you hear a song, there is usually a portion that gets repeated several times, and this turns out to be the most memorable part of the song. It’s called the chorus. And when you see the lyrics of the song printed out or as a computer file, you can see the words for the chorus printed near the top of the lyrics. However, between each of the verses of the song, there may be a special notation [Chorus] in brackets. This means “the chorus goes here.” Why did the person who typed the lyrics decide to write [Chorus] instead of typing out the lyrics of the chorus again?

The reason is to save printed space. It is unnecessary to retype the chorus because it doesn’t change during the song. When you are first learning the song, and you encounter [Chorus] in the lyrics, all you have to do is glance back at the top of the sheet to find the words to the Chorus. And then what do you do when you are finished with the chorus a second time? You need to remember where you left off in the rest of the song.

Exactly the same thing is going on in computer programs that contain functions.

Basic concepts regarding functions

Actually, you have already been using Python's built-in functions. Now we get to write our own. The first thing to understand about functions is where they fit into your program.

In Python, we *define* functions at the top of a program. After defining all the functions we wish to write, this is followed by the "main program." When you run a program, it begins with the first statement of the main program. We only enter a function if and when it is called. Most of the time, a function is called from somewhere in the main program. But it is possible for one function to call another.

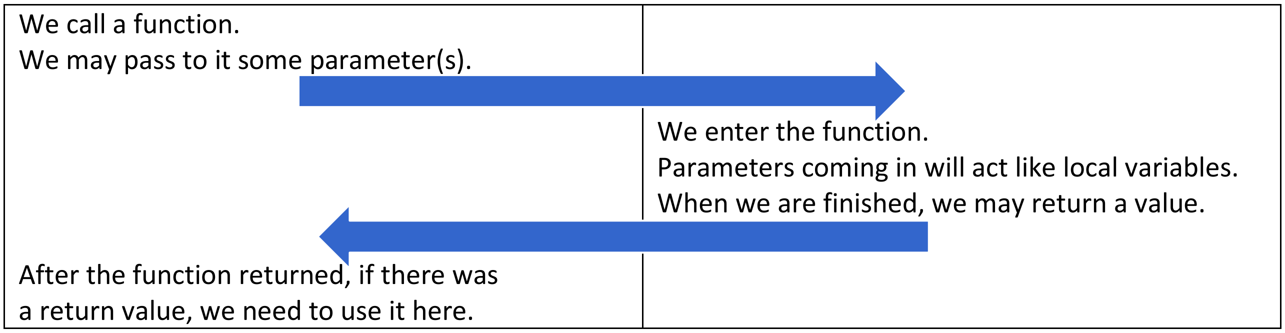
When you write a function, it is important to preface it with a comment, explaining its purpose, and how it works, just like you would include a general comment about an entire program.

Functions often communicate data with the main program. Usually we pass *parameters* to a function. Parameters are like a form of "input" to a function. Inside the function, we are free to create any "local" variables to assist us in our calculations. Finally, a function usually needs to *return* some data back to the place where the function was called. A function can have any number of parameters, but it can only return one value. However, in Python, we can return a list, so this is our loophole in case we actually need to return more than one number.

There is one subtlety to note about parameters inside a function. Any changes we make to them inside the function will have no effect once the function is finished. Consider this example. The function and main program are written here side by side for clarity, though in reality the function would be on top.

```
# main program                def fun(n):
a = 4                             n = n + 1
fun(a)                            print(n)
print(a)
```

In this example, since a = 4, the value 4 is passed to the function into the parameter n. So, inside the function, the value of n is initially 4. Then we increment it to 5 and print this out. When the function returns to the main program, a is still 4. So, when we print a, we print the number 4. Incrementing n inside the function had no effect on a in the main program. However, lists and dictionaries are exceptions to this scope rule. If you pass a list or a dictionary as a parameter to a function, then the function can change individual values.



We call a function.
We may pass to it some parameter(s).

We enter the function.
Parameters coming in will act like local variables.
When we are finished, we may return a value.

After the function returned, if there was
a return value, we need to use it here.

Let’s look at some examples. In each case, the function being defined appears in boldface.

Example #1

```
# A program that can count multiple times.

# count_up - This function will print the numbers
# from 1 up to some given maximum.
def count_up(maximum):
    for i in range(1, maximum + 1):
        print(i)

# Main program:  let’s count to 10, then to 20.
count_up(10)
count_up(20)
```

Example #2

```
# Let’s find the sum of some ranges of integers.

# find_sum - find the sum from low..high
def find_sum(low, high):
    sum = 0
    for i in range(low, high + 1):
        sum += i
    return sum

# Main program
print(find_sum(10, 25))
print(find_sum(4, 72))
print(find_sum(75, 150))
```

Discussion: For each of the two above example programs, answer these questions.

1. In the example program above, what is the name of the function?
2. How many times is the function called?
3. Does the function take any parameters? If so, how many, and what is/are the type(s) of the parameters?
4. Does the function return a value? If so, what kind of value?

Please note that there are two parts to writing any function:

First, we need to *define* the function itself. We accomplish this by using the Python keyword `def`. As we design the function, we need to consider the following.

- Does the function need parameters? If so, how many, and what should they be called?
- What calculation does the function need to do? Is any I/O necessary?
- What should the function return, if anything?

The second step is to actually *use* the function elsewhere in the program. In other words, somewhere in the main program we should call the function. Here are some considerations.

- What values should we pass to the function?
- What should we do with the answer that the function returns to us? For example, it turns out that a function call is often on the right side of an assignment statement.

Example #3: Using a function to create some lines to "Old Mother Hubbard". What is the advantage of using a function like this?

```
def substitute (place, thing, action):
    print ("She went to the", place)
    print ("To buy him", thing)
    print ("But when she came back,")
    print ("He", action, "\n")


#-------main program-----------------------------------------
intro = '''Old Mother Hubbard
Went to the cupboard,
To give the poor dog a bone;
But when she came there
The cupboard was bare,
And so the poor dog had none.
'''
print (intro)
substitute ("alehouse", "some beer", "sat in a chair")
substitute ("tailor's", "a coat", "was riding a goat")
substitute ("hatter's", "a hat", "was feeding her cat")
substitute ("barber's", "a wig", "was dancing a jig")
substitute ("cobler's", "some shoes", "was reading the news")
```

Example #4: We convert a Celsius temperature to Fahrenheit using a function.

```
# Conversion function
def convert_to_f(celsius_temp):
    return 9/5 * celsius_temp + 32

# --------------------------------
# Main program
c = float(input("Enter a Celsius temperature:  "))

f = convert_to_f(c)

print("The Fahrenheit temperature is ", f)
```

Parameters and return value

A function may or may not need parameters. And it may or may not return a value. Therefore, there are 4 possible scenarios that we should consider. They are all plausible, but each one describes a completely different purpose of the function.

1. Parameter(s) and return value Send data, receive data This is the most common scenario. The function is calculating something for the main program.	2. Parameter(s) but no return value Send data, but receive no data Most likely because we want to output something or write to a file while inside the function. The function is not trying to perform a calculation that we need later.
3. No parameter(s), but a return value Send no data, but receive data Most likely because the function is getting data from the user, an input file, or is generating random data – and this is needed by the rest of the program.	4. Neither parameter(s) nor return value Send no data, receive no data No data to be sent in either direction. Usually this means we are just printing a message, or we are doing some isolated step of an overall algorithm.

Most functions return a value. Because it's so easy for a function to return a value, it is common for a function not to perform output itself. Instead, the output would be performed after the function call.

What does a function call look like? It depends on whether the function you are calling returns a value. If not, then the function call just sits by itself. Otherwise, you need to do something to the return value. "Do something" means that the function call needs to be on the right side of an assignment, or it needs to be nested inside another function call as a parameter.

Here are some examples of what function calls look like:

Function call	Meaning
`fun(25)`	Go to the function called fun() and take the number 25 with you. Apparently, there is no return value to bring back.
`x = fun(25)`	The value returned by fun(25) is put into the variable x.
`print(fun(25))`	The value returned by fun(25) is printed.
`fun2(fun(25))`	The value returned by fun(25) is passed to a second function fun2().

In the above examples, notice that the first one is different from the other three. All we do is pass the number 25 to a function called `fun`. We don't expect the function to return anything. If it did, it would be thrown away! In the other three examples, we do something with the return value.

It is a common mistake to write a function call like "fun(25)" when the function in fact returns a value.

The following example program illustrates the mistake of discarding a return value. It contains a function that performs a simple calculation.

```
# for a given value, return 7 less than 3 times the square.
def f(x):
    return 3 * x ** 2 - 7

# main program
value = int(input("Please enter a number:  "))
f(value)
print("After applying the function, the answer is ", result)
```

This program will not run at all. It contains an error because the variable result was never assigned. The way to fix the error is to change the statement `f(value)` to this:

```
result = f(value)
```

It's worth stating again: <u>A common mistake is to forget to make use of the value returned by a function. Often you must assign the returned value to a variable at the place you call the function</u>. Think about it: Why call a function if you are going to throw away the answer?

<u>Recap</u>

Let's review some important facts about functions:

- The parameters and return value are optional. It all depends on the purpose of the function.

- Variables declared inside a function are “local” only, and cannot be used outside the function. Similarly, changes to variables inside a function have no effect outside the function.
- You can only return one value from a function, but it can be anything, including a list.
- We have already been using built-in functions like print( ).
- The function is introduced with the word `def`, and the body of the function is indented. Unindenting signals Python that the function is over.

Any of the practice problems we saw earlier can be written as a function instead of a program. Try it! Let me suggest:

- Put the algorithm’s calculations inside the function. The function should not do I/O itself. The function’s “input” is really the parameters, and its “output” is the result it returns. If you’d like, you can write a separate function to obtain the interactive input from the user and/or a separate function to display the output.
- The main program should pass the appropriate (input) data to the function. And most importantly, it needs to use the function’s return value to output to the user.

<u>Practice</u>:

1. Consider the following Python program. Trace its execution by hand and explain what happens. What is the output?

```
def fun(x):
    return 2*x – 1

def fun2(x, y):
    return fun(x) + y

def getNumber():
    value = int(input(“Please enter an integer:  “))
    return value

# main program
a = 3
b = fun(a)
print(b)

c = fun2(a, b)
print(c)

d = getNumber()
print(d)
```

2. Consider the following Python program. Explain what it accomplishes.

```
def compute(n):
    return n + 2

# main program
L = [ 12, 87, 5, 13, 94 ]

for number in L:
    print("The result for ", number, " is ", compute(number))
```

3. What is the output when the following program runs?

```
def a():
    print("a", end = "")

def b():
    a()
    print("b", end = "")

def c():
    a()
    b()
    print("c", end = "")

def d():
    a()
    b()
    c()
    print("d", end = "")

# -----------------------
# main program
# -----------------------

a()
print()

b()
print()
```

```
c()
print()

d()
print()
```

4. Write a function that takes two parameters, the length and width of a rectangle. The function should return the area of the rectangle. What should the function be called? Similarly, write a second function that returns the rectangle's perimeter. Call the functions from the main program with the values 10 and 8.

5. Let's create a short program that includes the use of a function. The program will ask the user to enter the freezing and boiling points of a substance. We need to convert these values from Fahrenheit to Celsius.

   First, we need a conversion function. The parameter is the Fahrenheit temperature. The return value is the corresponding Celsius temperature.

   The main program will do the following steps: It should ask the user for the freezing and boiling temperatures in Fahrenheit. Next, it needs to call the convert function on the freezing temperature to convert it to Celsius. Then it should do the same for the boiling temperature. Finally, output both Celsius temperatures.

6. Design a program that converts an amount of money from US dollars into euros. Use a function to perform the actual conversion, and do all the I/O in the main program. Assume that 1 Euro is worth $1.14.

7. Write a function that takes a positive integer parameter. The function will interpret this integer as a year in the Gregorian calendar, and return True or False depending on whether that year is a leap year or not.

8. Write a function that takes two positive integer parameters: a month number (1-12) and a year. The function should return the number of days in the given month. Why is the year included as one of the parameters? Because the month number might be 2, and in this case we need to call the leap year function to see if we should return 28 or 29.

9. Write a function that takes a positive integer parameter. The function will return the sum of the digits of this number.

10. Write a function that takes a string parameter, and returns the number of vowels in the string.

11. Write a function that takes a string parameter s. The function will return a modified version of s: Add 1 to the ASCII value of each character s, and return the new string. Use the built-in chr() and ord() functions.

12. Write a function that takes a string parameter. This string will contain three positive integers separated by hyphens like this: "27-32-8". The string represents the record of an NHL team, showing the number of wins, losses, and overtime losses. The function should return the number of points earned according to this record. This is determined by awarding 2 points for each win and 1 point for each overtime loss. Thus, for "27-32-8", the function should return 62.

13. Airlines use a computer program to print boarding passes. One thing that has to be printed on the boarding pass is the suggested time for the passenger to be ready to board the airplane. So, we can design a program to determine the boarding time, given the takeoff time.
    a. Write a function that converts a time string such as "940a" into a number of minutes since midnight.
    b. Write a function to do the opposite: Given a number of minutes (0-1439) since midnight, return a string representing the time.
    c. Use the two functions above to solve the general problem. Assume that the passenger must be ready to board 25 minutes early.
    d. Write another function that calls the function in part (b) for all possible minutes of the day. It should not return a value but instead write its answers to a file.
    e. Write a function that reads the file created by part (d). For each line in the file, call the function in part (a). Verify that all the return values are 0-1439 in sequence.

14. Write a function that takes 2 string parameters. The function will determine if the two strings are anagrams of each other.

15. Write a function that takes 2 tuple parameters, each having 2 integer parameters: (a,b) and (c,d). These 4 numbers represent timestamps. The tuples refer to two intervals of time. One event begins at time a and ends at time b. A second event begins at time c and ends at time d. The function should return True if the events overlap, and therefore cannot be scheduled in the same room. Return False if there is no overlap, meaning that it's safe to schedule both events in the same room. For example, return False if one event begins at the same time as the other event ends.

16. Write a function that takes a list of real numbers. These represent the scores given by judges at a diving competition. The function should return the average score, after disregarding the highest and lowest scores. Don't change the list.

17. Let's write some functions that each take a list of 5 integers. These integers represent the roll of 5 dice in the game Yahtzee. Assume that the list has already been sorted in ascending order. Design a different function for each part:

a. A function to determine if all 5 numbers are equal.
b. A function to determine if at least 4 numbers are equal.
c. A function to determine if at least 3 numbers are equal.
d. A function to determine if we have a "full house": 3 numbers are equal, and the other 2 numbers are equal to a different value.
e. All 5 numbers form a consecutive sequence of values. In other words, the 5 values are of the form n, n+1, n+2, n+3 and n+4 for some integer n.

18. Write a function that implements the Euclidean algorithm to find the greatest common factor of two positive integers. The function should print the greatest common factor, and return the number of iterations that the algorithm needed to determine the result. In the main program, call your function for all possible pairs of integers between 1 and 100, inclusive. Determine which pair requires the greatest number of steps of the Euclidean algorithm.

19. Write a function to determine if a number is prime. Use this function to help you find twin primes. For each integer n from 1 to 1000, see if n and n+2 are both prime. If so, print the pair.

20. Write a function to solve this problem from chess: Given two tuples of integers (x1, y1) and (x2, y2) representing the row and column numbers of two squares on a chess board, return the minimum number of moves that a king needs to make to travel from the first location to the second.

21. Write a function that takes 3 parameters representing the number of degrees, minutes and seconds of some angle. The degrees and minutes are integers, but the number of seconds is real. The function should return the decimal number of degrees. Also write another function that goes in the other direction: given a real number of degrees, calculate the degrees, minutes and seconds so that the degrees and minutes are integer.

22. Write a function that takes the coefficients of a quadratic equation as parameters. The function should return a tuple representing the solutions of this equation. Assume that the first parameter is not zero, and the roots are real.

23. Go back to the practice problems you saw earlier chapters. Select one of these problems, and write a program to solve it, using a function to perform the essential computation.

24. We can find hierarchical information and multiple "function instances" in everyday life. For example, the nutrition label for Publix's Cool Mint Cookie frozen yogurt shows several levels of nesting. Literally, the ingredients read as follows.

    Cultured lowfat milk, sugar, corn syrup, nonfat dry milk, whey protein concentrate, cultured dairy solids, chocolate fudge [sugar, peanut oil, cocoa (processed with alkai), whey, salt, soy lecithin], choco-coated mint cookies ** {cookie [wheat flour, sugar, partially hydrogenated soybean oil and/or cottonseed oil, cocoa processed with alkai, soy lecithin (an emulsifier), salt, sodium bicarbonate], sugar, coconut oil, cocoa processed with alkai, cocoa, nonfat milk, palm kernel oil, milkfat, natural flavors, soy lecithin, oil of peppermint, color added (yellow 5 lake, blue 1 lake)}, vegetable mono and diglycerides, natural flavors, locust bean gum, guar gum, sodium citrate, calcium sulfate, carrageenan and peppermint extract.

** manufactured in a facility that produces peanuts, tree nuts and milk products.

Copy and paste the above ingredient list to a new blank text file. Separate the list out so that only one ingredient or subingredient appears per line. Indent the ingredients to show the levels of nesting. The above ingredient list uses various types of grouping symbols such as parentheses, brackets and braces. Place each grouping symbol by itself on a line to make the hierarchy easier to read.

Which foodstuffs are mentioned more than once in the list of ingredients? This would be analogous to a function that is called from more than one place in a computer program.