



Python Programming

asaad.nayyef@uomus.edu.iq



Week 8: Functions and Function Arguments

Introduction to Functions

A function is a group of statements that exist within a program for the purpose of performing a specific task. Functions are the next step toward creating optimized code as a software developer. If the same block of code is reused repeatedly, a function allows the programmer to write the block of code once, name the block, and use the code as many times as needed by calling the block by name. Functions can read in values and return values to perform tasks, including complex calculations.

Learning objectives

By the end of this section you should be able to

1. Identify function calls in a program.
2. Define a parameterless function that outputs strings.
3. Describe benefits of using functions.

Function Names

Before we discuss the process of creating and using functions, we should mention a few things about function names. Just as you name the variables that you use in a program, you also name the functions. A function's name should be descriptive enough so that anyone reading your code can reasonably guess what the function does.

Python requires that you follow the same rules that you follow when naming variables, which we recap here:

1. You cannot use one of Python's key words as a function name.
2. A function name cannot contain spaces.
3. The first character must be one of the letters a through z, A through Z, or an underscore character (_).
4. After the first character you may use the letters a through z or A through Z, the digits 0 through 9, or underscores.
5. Uppercase and lowercase characters are distinct.

Defining and Calling a Function

A function is defined using the `def` keyword. The first line contains `def` followed by the function name, parentheses (with any parameters), and a colon. **A function must be defined before the function is called.**

To create a function, you write its definition. Here is the general format of a function definition

in Python:

```
def function_name():  
    statement  
    statement  
    etc.
```



Python Programming

asaad.nayyef@uomus.edu.iq



```
def message():
    print('Welcome to')
    print('Al Mustaqbal University.')
# Call the message function.
message()
Welcome to
Al Mustaqbal University.
```

```
# This program has two functions. First we
# define the main function.
```

```
def main():
    print('I have a message for you.')
    message()
    print('Goodbye!')
```

```
# Next we define the message function.
```

```
def message():
    print('Welcome to')
    print('Al Mustaqbal University.')
```

```
# Call the main function.
main()
```

I have a message for you.
Welcome to
Al Mustaqbal University.
Goodbye!

Passing Arguments to Functions

An argument is any piece of data that is passed into a function when the function is called. A parameter is a variable that receives an argument that is passed into a function. Sometimes it is useful not only to call a function, but also to send one or more pieces of data into the function. Pieces of data that are sent into a function are known as *arguments*. The function can use its arguments in calculations or other operations.

If you want a function to receive arguments when it is called, you must equip the function with one or more parameter variables. A parameter variable, often simply called a parameter, is a special variable that is assigned the value of an argument when a function is called. Here is an example of a function that has a parameter variable:

```
def show_double(number):
    result = number * 2
    print(result)
```

This function's name is `show_double`. Its purpose is to accept a number as an argument and display the value of that number doubled. Look at the function header and notice the word `number` that appear inside the parentheses. This is the name of a parameter variable. This variable will be assigned the value of an argument when the function is called.

```
# This program demonstrates an argument being
# passed to a function.
def main():
```



Python Programming

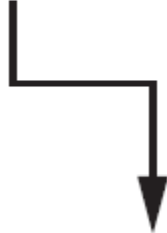
asaad.nayyef@uomus.edu.iq



```
value = 5
show_double(value)
# The show_double function accepts an argument
# and displays double its value.
def show_double(number):
    result = number * 2
    print(result)
# Call the main function.
main()
10
```

The value variable is passed as an argument

```
def main():
    value = 5
    show_double(value)
```

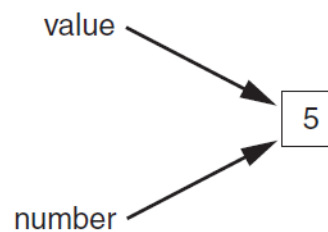


```
def show_double(number):
    result = number * 2
    print(result)
```

The value variable and the number parameter reference the same value

```
def main():
    value = 5
    show_double(value)

def show_double(number):
    result = number * 2
    print(result)
```





Python Programming

asaad.nayyef@uomus.edu.iq



Passing Multiple Arguments

Often, it's useful to write functions that can accept multiple arguments. Program shows a function named `show_sum`, that accepts two arguments. The function adds the two arguments and displays their sum.

```
# This program demonstrates a function that accepts
# two arguments.
def main():
    print('The sum of 12 and 45 is')
    show_sum(12, 45)

# The show_sum function accepts two arguments
# and displays their sum.
def show_sum(num1, num2):
    result = num1 + num2
    print(result)

# Call the main function.
main()
The sum of 12 and 45 is
57
```

Two arguments passed to two parameters

```
def main():
    print('The sum of 12 and 45 is')
    show_sum(12, 45)
```

```
def show_sum(num1, num2):
    result = num1 + num2
    print(result)
```

num1 → 12

num2 → 45



Python Programming

asaad.nayyef@uomus.edu.iq



Making Changes to Parameters

When an argument is passed to a function in Python, the function parameter variable will reference the argument's value. However, any changes that are made to the parameter variable will not affect the argument.

*# This program demonstrates what happens when you
change the value of a parameter.*

```
def main():  
    value = 99  
    print('The value is', value)  
    change_me(value)  
    print('Back in main the value is', value)  
  
def change_me(arg):  
    print('I am changing the value.')  
    arg = 0  
    print('Now the value is', arg)
```

Call the main function.
main()

The value is 99

I am changing the value.

Now the value is 0

Back in main the value is 99

Global Variables and Global Constants

A global variable is accessible to all the functions in a program file.

When a variable is created by an assignment statement that is written outside all the functions in a program file, the variable is *global*. A global variable can be accessed by any statement in the program file, including the statements in any function.

Create a global variable.
my_value = 10
*# The show_value function prints
the value of the global variable.*

```
def show_value():  
    print(my_value)
```

Call the show_value function.
show_value()

10



Python Programming

asaad.nayyef@uomus.edu.iq



```
# Create a global variable.
number = 0
def main():
    global number
    number = int(input('Enter a number: '))
    show_number()

def show_number():
    print('The number you entered is', number)
# Call the main function.
main()
Enter a number: 77
The number you entered is 77
```

Examples

```
# Ask the user for their university name
name = input("What's your university name? ")
# Print the output
print(f"hello, {name}")
```

What's your university name? Al Mustaqbal University

hello, Al Mustaqbal University

We can better our code to create our own special function that says “hello” for us!

```
def hello():
    print("hello")
name = input("What's your university name? ")
hello()
print(name)
```

What's your university name? Al Mustaqbal University

hello

Al Mustaqbal University

```
# Create our own function
```

```
def hello(to):
    print("3 hello,", to)
# Output using our own function name = input("What's your name? ")
hello(name)
# Output using our own function
name = input("What's your name4? ")
hello(name)
```

3 hello, Al Mustaqbal University

What's your name4? Al Mustaqbal University

3 hello, Al Mustaqbal University

We can change our code to add a default value to hello:

```
# Create our own function
def hello(to="world"):
    print("hello,", to)
# Output using our own function
name = input("What's your university name5? ")
hello(name)
```



Python Programming

asaad.nayyef@uomus.edu.iq



```
# Output without passing the expected arguments  
hello()
```

What's your university name5? Al Mustaqbal University

hello, Al Mustaqbal University

hello, world

```
def main():  
# Output using our own function  
    name = input("What's your university name6? ")  
    hello(name)  
# Output without passing the expected arguments  
    hello()  
# Create our own function  
def hello(to="world"):  
    print("hello,", to)  
main()  
hello, Al Mustaqbal University  
hello, world
```

Returning Values

```
def main():  
    x = int(input("What's x? "))  
    print("x squared is", square(x))  
def square(n):  
    return n * n  
main()  
What's x? 5  
x squared is 25
```

```
def maxnum(x, y):  
    if x > y:  
        return x  
    return y  
  
print(maxnum(10, 20))  
20  
def maxnum(x, y):  
    return max(x, y)  
  
print(maxnum(10, 20))  
20  
def maxnum(x, y, z):  
    return max(max(x, y), z)  
print(maxnum(10, 20, 50))  
50
```



Python Programming

asaad.nayyef@uomos.edu.iq



In Python, arguments are passed **by object reference**, which behaves like **call by reference** for mutable objects and **call by value** for immutable objects.

1. Call by Value (Immutable Objects: int, float, string, tuple)

When you pass immutable objects (like numbers or strings), Python creates a new copy of the value, so changes inside the function **do not affect** the original variable.

```
def modify_value(x):  
    x = 20  # Changes local copy, does not affect original variable  
    print("Inside function:", x)
```

```
a = 10  
modify_value(a)  
print("Outside function:", a)  # Original value remains unchanged
```

Inside function: 20

Outside function: 10

2. Call by Reference (Mutable Objects: list, dict, set)

When you pass mutable objects (like lists or dictionaries), Python passes a **reference** to the same object, so modifications inside the function **affect** the original variable.

```
def modify_list(lst):  
    lst.append(4)  # Modifies the original list  
    print("Inside function:", lst)
```

```
my_list = [1, 2, 3]  
modify_list(my_list)  
print("Outside function:", my_list)  # Original list is modified  
Inside function: [1, 2, 3, 4]
```

Outside function: [1, 2, 3, 4]

```
def print_greeting():  
    print(out_str)  
hour = int(input("Enter thr Hour: "))  
min = int(input("Enter thr Min: "))  
if hour < 12:  
    out_str = "Good morning"  
else:  
    out_str = "Good day"  
print_greeting()
```

Enter thr Hour: 8

Enter thr Min: 30

Good morning



Python Programming

asaad.nayyef@uomus.edu.iq



```
# This program gets three test scores and displays
# their average. It congratulates the user if the
# average is a high score.
# Global constant for a high score
HIGH_SCORE = 95
def main():
    #Get the three test scores.
    test1 = int(input('Enter the score for test 1: '))
    test2 = int(input('Enter the score for test 2: '))
    test3 = int(input('Enter the score for test 3: '))
    # Calculate the average test score.
    average = (test1 + test2 + test3) / 3
# Print the average.
    print('The average score is', average)
# If the average is a high score,
# congratulate the user.
    if average >= HIGH_SCORE:
        print('Congratulations!')
        print('That is a great average!')
# Call the main function
main()
```

Enter the score for test 1: 99

Enter the score for test 2: 98

Enter the score for test 3: 97

The average score is 98.0

Congratulations!

That is a great average!

```
def main():
# Print the table headings.
    print('Number\tSquare')
    print('-----')
# Print the numbers 1 through 10
# and their squares.
    for number in range(1, 11):
        square = number**2
        print(number, '\t', square)
# Call the main function.
main()
```

Number Square

```
-----
1      1
2      4
3      9
4     16
5     25
```



Python Programming

asaad.nayyef@uomos.edu.iq



6	36
7	49
8	64
9	81
10	100

```
# This program uses a loop to display a
# table of numbers and their squares.
def main():
    # Get the ending limit.
    print('This program displays a list of numbers')
    print('(starting at 1) and their squares.')
    end = int(input('How high should I go? '))
    # Print the table headings.
    print()
    print('Number\tSquare')
    print('-----')
    # Print the numbers and their squares.
    for number in range(1, end + 1):
        square = number**2
        print(number, '\t', square)

# Call the main function.
main()
```

This program displays a list of numbers

(starting at 1) and their squares.

How high should I go? 5

Number Square

```
-----
1      1
2      4
3      9
4     16
5     25
```