Functions

What is a Function?

- A *function* is a block of organized, reusable code that is used t
- A *function* provides better modularity for your applications an
- Python provides *built-in functions*
  - These are part of the core language
- Python also allows you to define your own *user-defined functi*

Built-In Functions

- You’ve already been using built-in functions!
  - The *print* function to print a string
    `print(“Hello World!”)`
  - The *input* function to get user input
    `input(“What is your favorite movie?”)`
  - The *int* function to cast from one data type to an integer
    `int(3.1)`
- There are lots of built-in functions. Here are some others:
  - float(*x*) - casts string or integer *x* to a float
  - round(*float*, *int*) - rounds *float* to *int* decimal places
  - max(*arg1*, *arg2*, *argN*) - gets the maximum value of argum
  - min(*arg1*, *arg2*, *argN*) - gets the minimum value of argum
  - len(*s*) – gets the length (number of items) of an object *s*

For reference: https://docs.python.org/3/library/functions.html

User-Defined Functions

- Functions have conventions
  - Name a function based on what it does
  - Whitespace is important!
    - Function body “code blocks” (groups of statements) have t
- Sometimes a function takes an input
  - These are called *parameters*
  - When you call (or use) the function, you pass *arguments*
- Sometimes a function produces an output
  - This is called the function’s *return* value

User-Defined Functions

- You define a *function* using the *def* keyword, followed by the *f*

```
def function_name(param1, …, paramN):
     statements
     return
```

- Parenthesis include optional *parameters*, treating them as va
- Functions optionally *return* a value, which allows us to get th it's done executing.. Whatever follows the *return* keyword will your code where the function was called.

User-Defined Functions

- Let’s define a function *say_hello*
    - It prints the word “Hello!”
    - It has no *parameters*, which means, there is nothing pass

```
def say_hello():
    print(“Hello!”)
```

- Here’s how we use the function *say_hello*

```
say_hello()
```

User-Defined Functions

- Let’s define a function *say_something_specific*
  - It takes one string as a *parameter*
  - It prints that given string

```
def say_something_specific(thing_to_say):
    print(thing_to_say)
```

- Now let’s use the function *say_something_specific*
  - When we call it, we pass “Hello there world!” as an *argur*
  - The function will then print the given string

```
say_something_specific(“Hello there world!”):
```

User-Defined Functions

- Let’s define a function *number_sum*
  - It takes two numbers as *parameters*, separated by a comm
  - It prints and *returns* the sum of those numbers

```
def number_sum(num1, num2):
    sum = num1 + num2
    print(“The sum is:”, sum)
    return sum
```

- Now let’s use the function *number_sum*
  - When we call it, we pass the value of a (which is 5) and th *arguments*
  - *The function* will *return* the sum of the given numbers
  - We’ll store it in a variable result and print it

```
a = 5
b = 3
result = number_sum(a, b)
print(result) #8
```