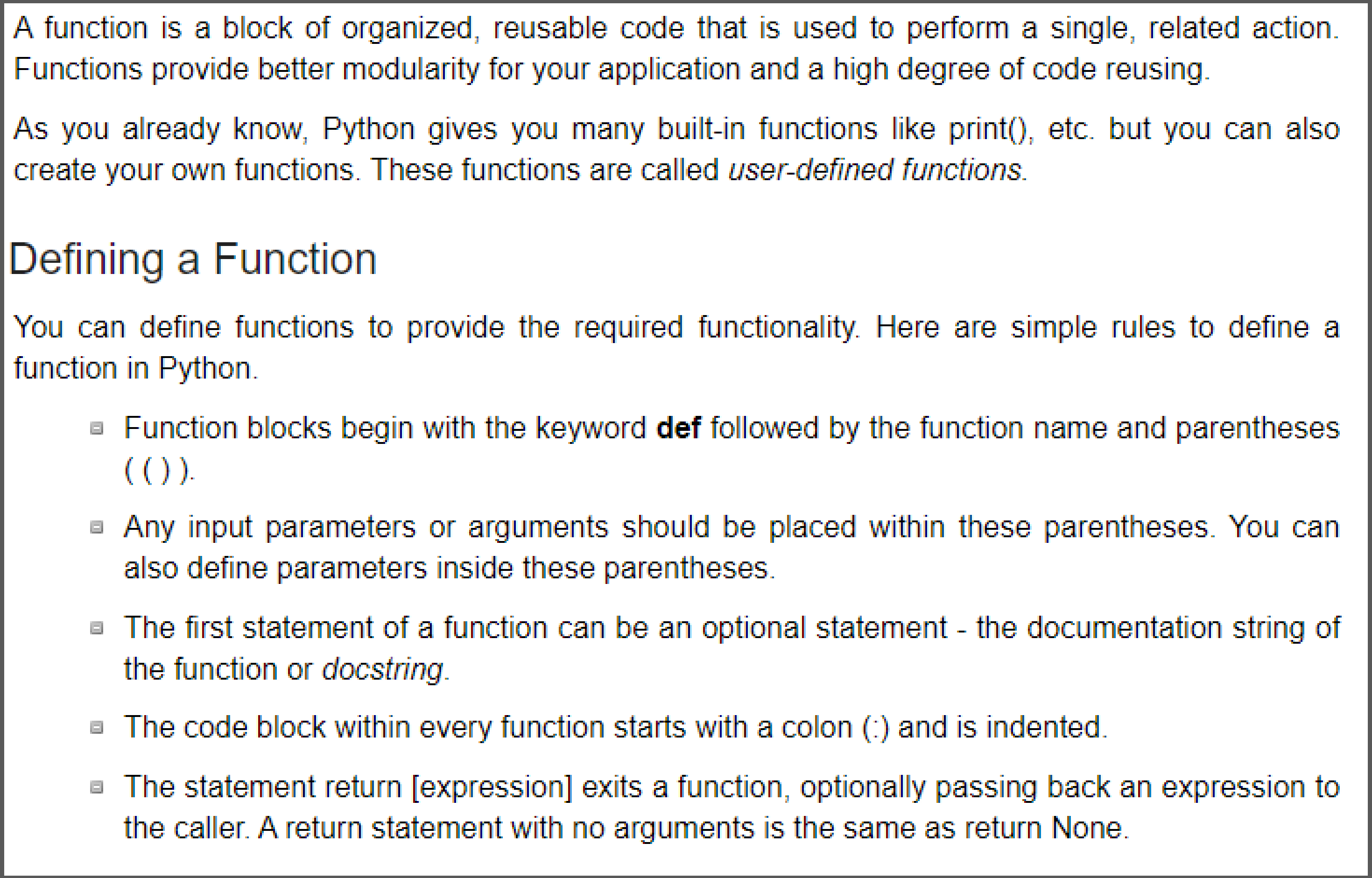
Network Programming in Python I

Justin Ellis MBA

Review

Any Questions?

Functions-Tutorialspoint



A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

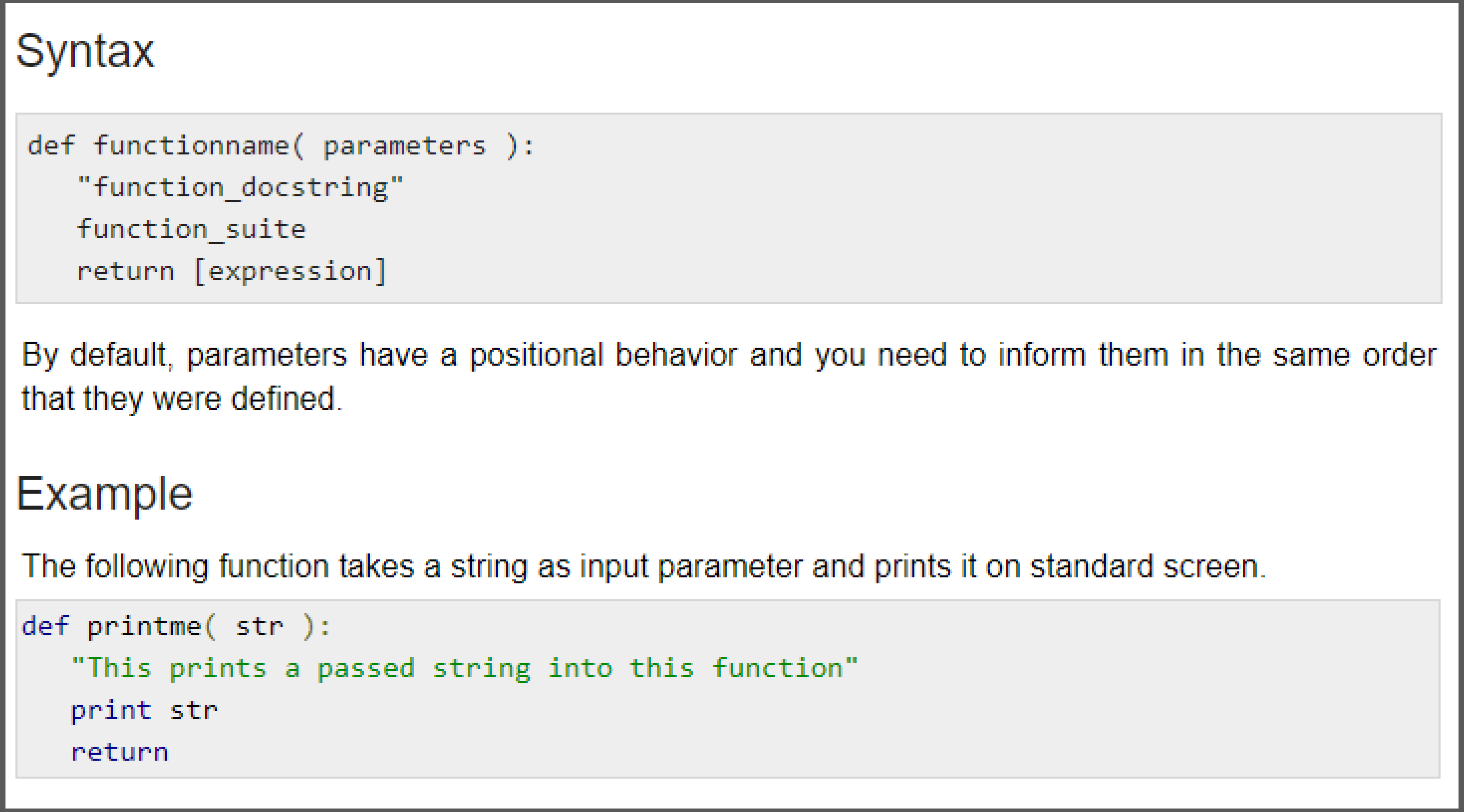
As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions*.

Defining a Function

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Functions-Tutorialspoint



Syntax

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

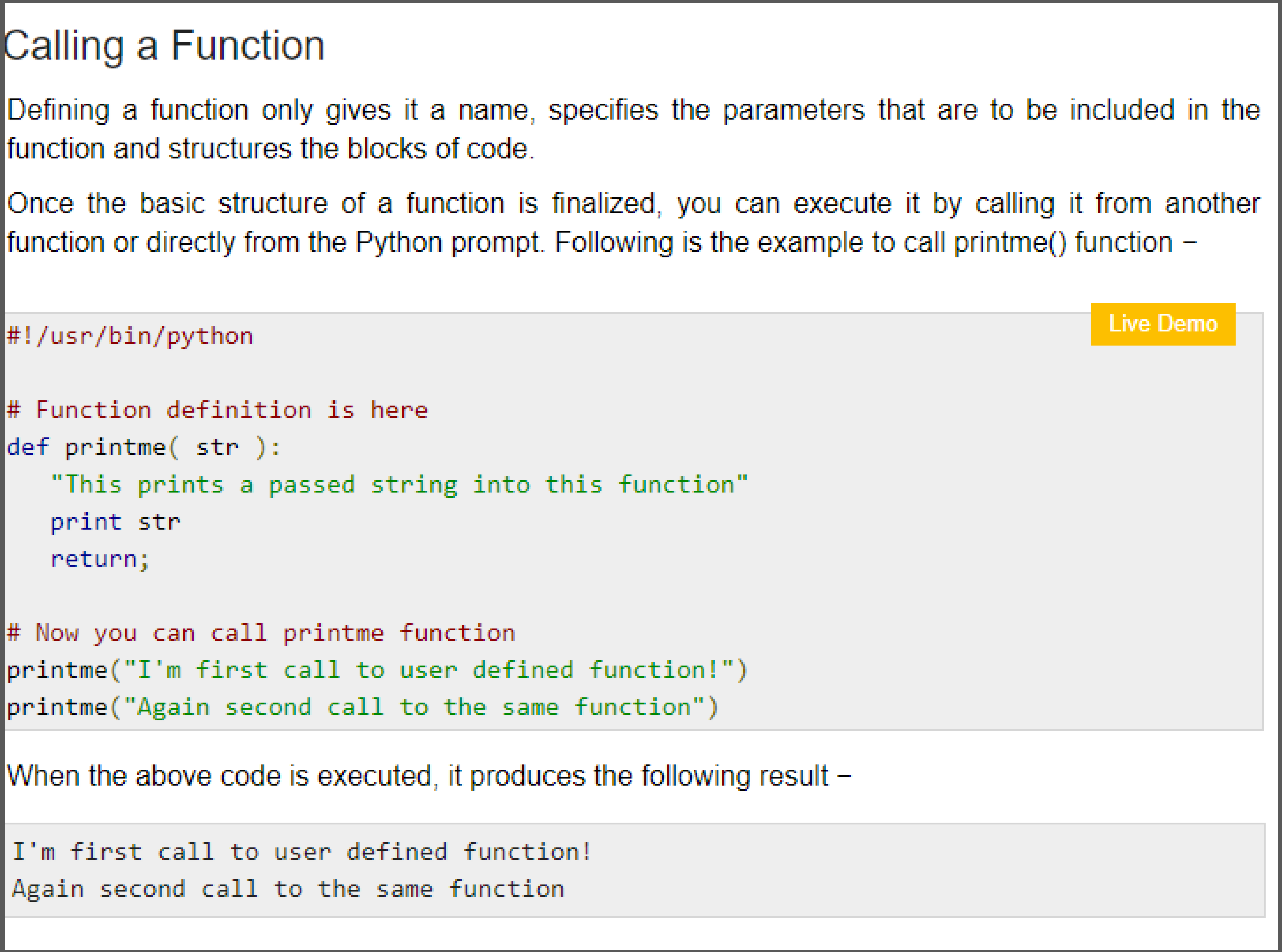
By default, parameters have a positional behavior and you need to inform them in the same order that they were defined.

Example

The following function takes a string as input parameter and prints it on standard screen.

```
def printme( str ):
   "This prints a passed string into this function"
   print str
   return
```

Functions-Tutorialspoint



Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

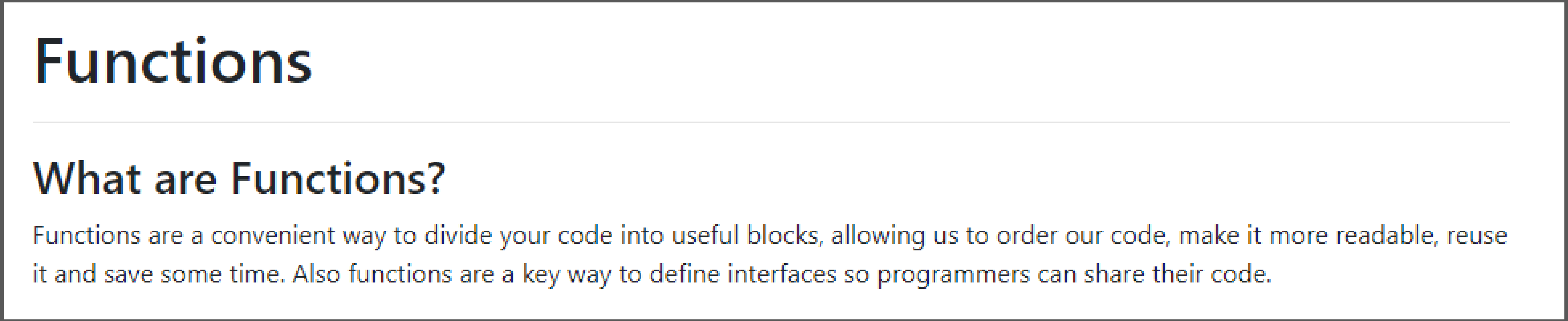
Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function –

```
#!/usr/bin/python

# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str
   return;

# Now you can call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

When the above code is executed, it produces the following result –

```
I'm first call to user defined function!
Again second call to the same function
```

Functions-learnpython

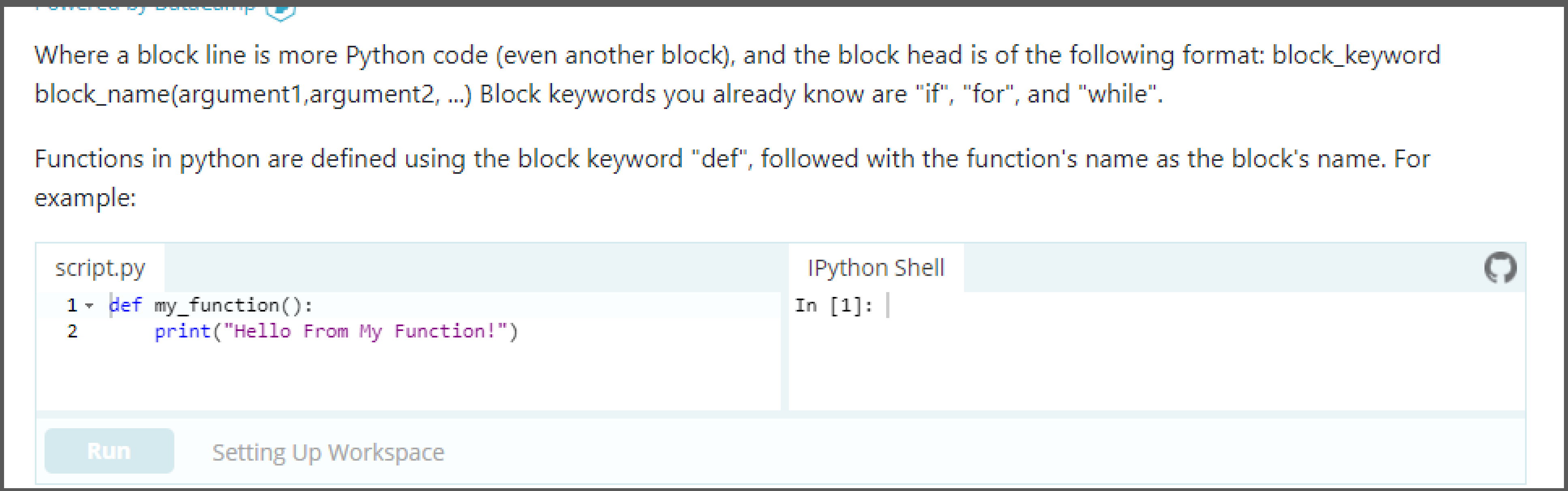


Functions

What are Functions?

Functions are a convenient way to divide your code into useful blocks, allowing us to order our code, make it more readable, reuse it and save some time. Also functions are a key way to define interfaces so programmers can share their code.

Functions-learnpython



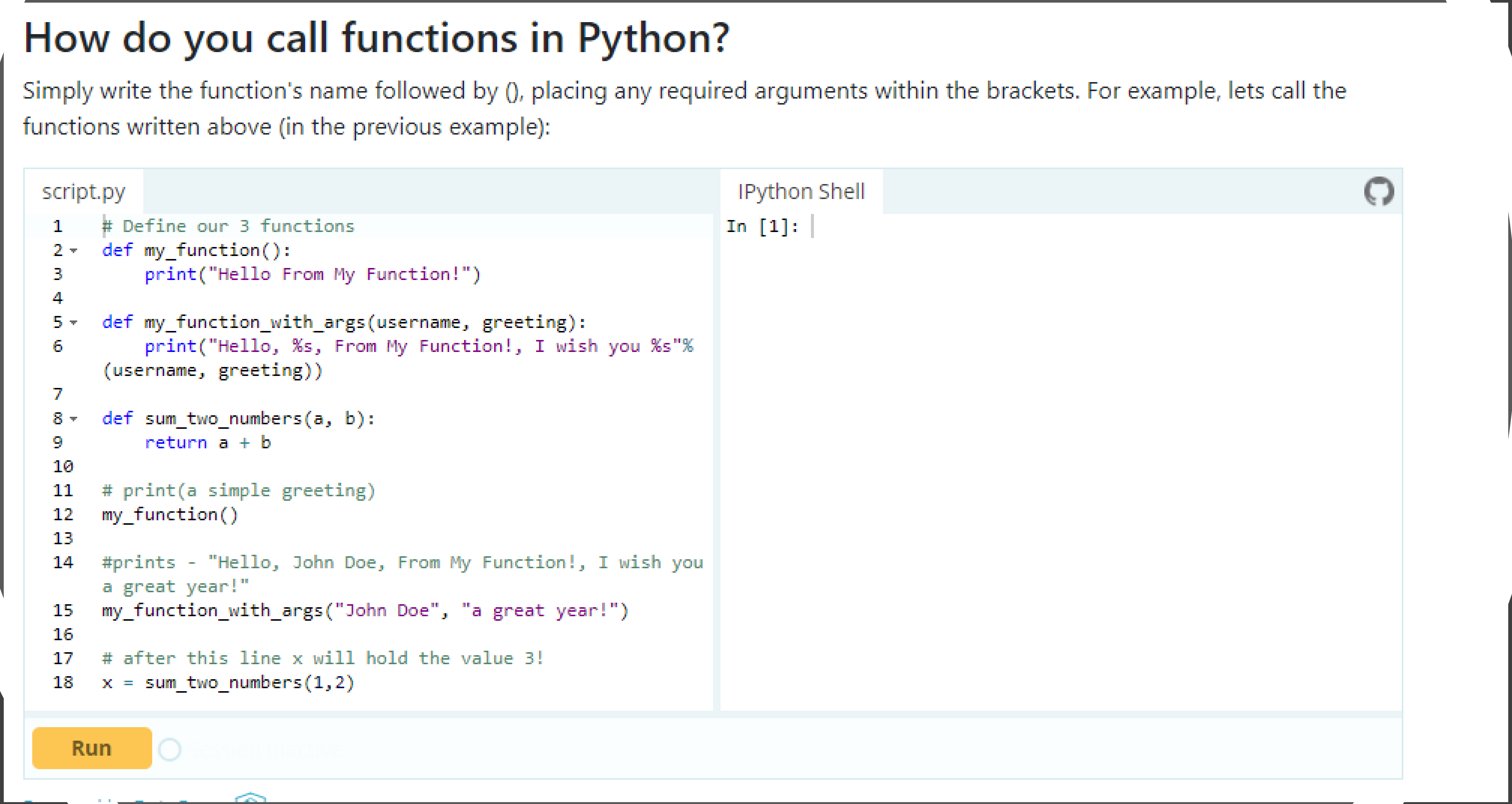
Where a block line is more Python code (even another block), and the block head is of the following format: block_keyword block_name(argument1,argument2, ...) Block keywords you already know are "if", "for", and "while".

Functions in python are defined using the block keyword "def", followed with the function's name as the block's name. For example:

```python
def my_function():
    print("Hello From My Function!")
```

```
In [1]:
```

Functions-learnpython



How do you call functions in Python?

Simply write the function's name followed by (), placing any required arguments within the brackets. For example, lets call the functions written above (in the previous example):

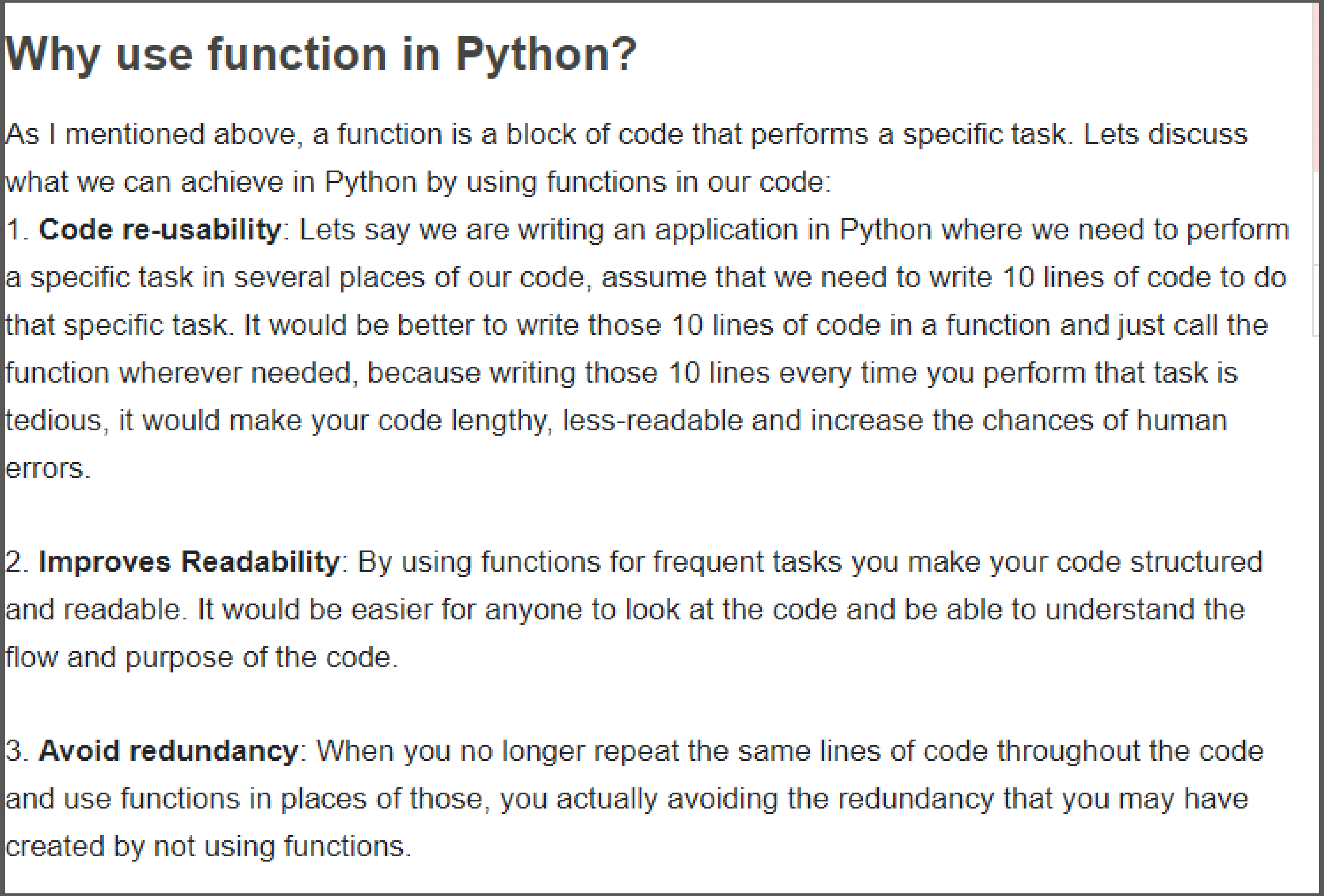
script.py

```python
# Define our 3 functions
def my_function():
    print("Hello From My Function!")

def my_function_with_args(username, greeting):
    print("Hello, %s, From My Function!, I wish you %s"%(username, greeting))

def sum_two_numbers(a, b):
    return a + b

# print(a simple greeting)
my_function()

#prints - "Hello, John Doe, From My Function!, I wish you a great year!"
my_function_with_args("John Doe", "a great year!")

# after this line x will hold the value 3!
x = sum_two_numbers(1,2)
```

IPython Shell

```
In [1]:
```

Why Do we use functions



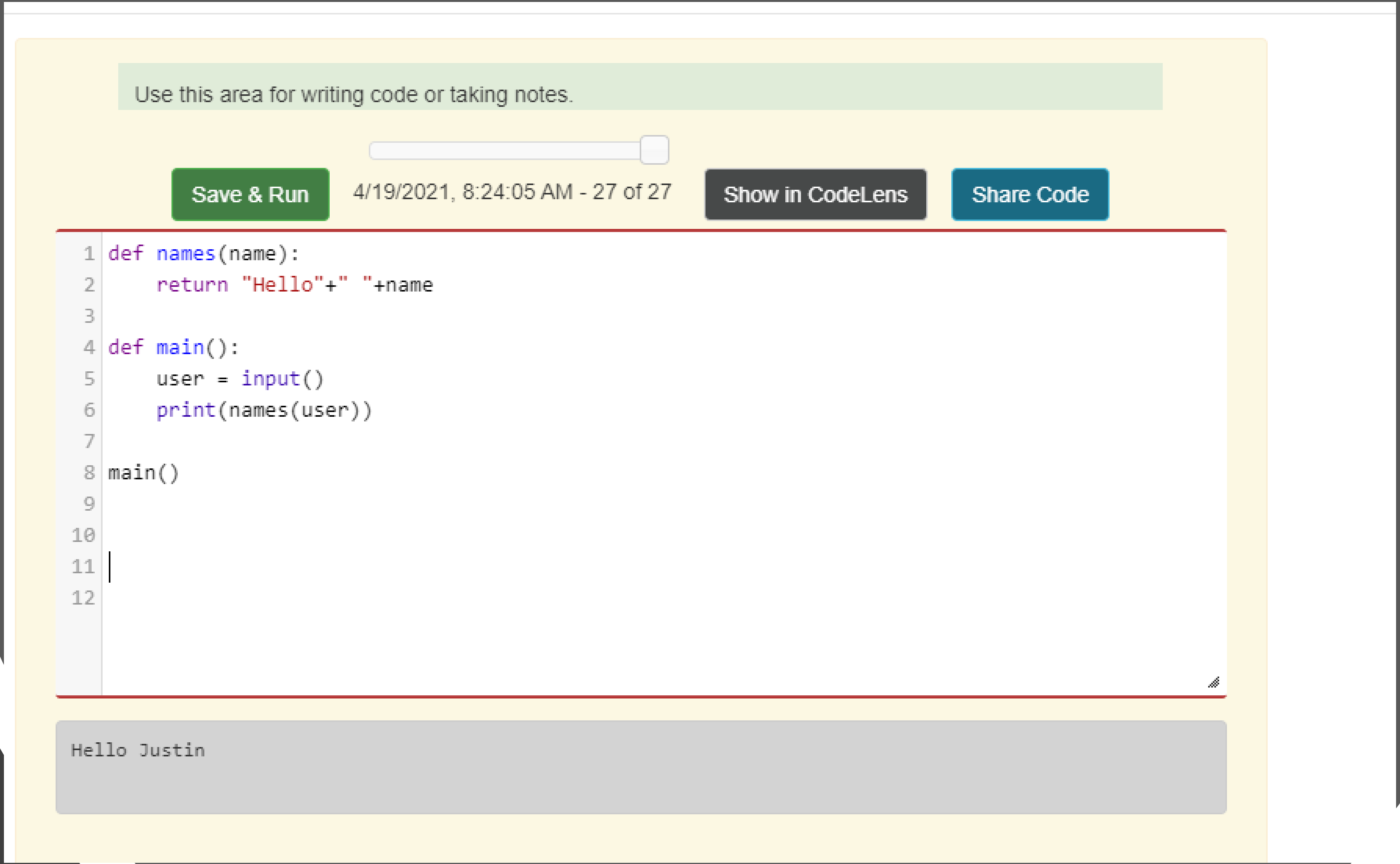
Why use function in Python?

As I mentioned above, a function is a block of code that performs a specific task. Lets discuss what we can achieve in Python by using functions in our code:
1. **Code re-usability**: Lets say we are writing an application in Python where we need to perform a specific task in several places of our code, assume that we need to write 10 lines of code to do that specific task. It would be better to write those 10 lines of code in a function and just call the function wherever needed, because writing those 10 lines every time you perform that task is tedious, it would make your code lengthy, less-readable and increase the chances of human errors.

2. **Improves Readability**: By using functions for frequent tasks you make your code structured and readable. It would be easier for anyone to look at the code and be able to understand the flow and purpose of the code.

3. **Avoid redundancy**: When you no longer repeat the same lines of code throughout the code and use functions in places of those, you actually avoiding the redundancy that you may have created by not using functions.

Main Functions


Use this area for writing code or taking notes.
Save & Run
4/19/2021, 8:24:05 AM - 27 of 27
Show in CodeLens
Share Code

```python
def names(name):
    return "Hello"+" "+name

def main():
    user = input()
    print(names(user))

main()
```

```
Hello Justin
```

Main Functions

6.8. Using a Main Function — How to Think like a Computer Scientist: Interactive Edition (runestone.academy)