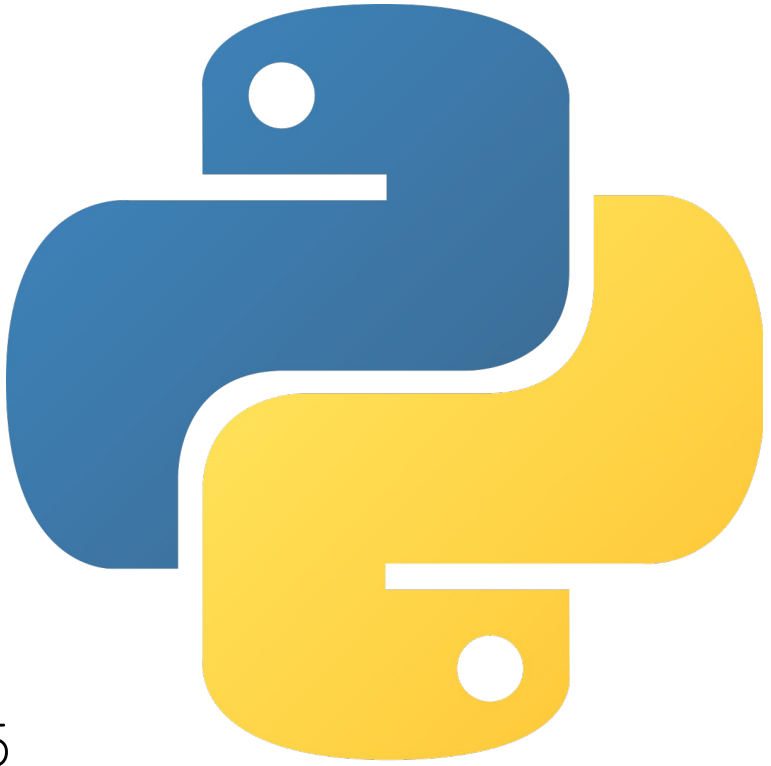
HackerFrogs Afterschool
Python Programming Basics: Part 6

Class:
Programming (Python)

Workshop Number:
AS-PRO-PY-06

Document Version:
1.75

Special Requirements:
Completion of AS-PRO-PY-05



What We Learned Before

This workshop is the sixth class for intro to Python programming.

During our last workshop, we learned about a few programming concepts through Python, including the following:

Programming Loops

```
for i in range(3):
    print(i)
```

```
0
1
2
```

Loops are tools that programmers use to run the same instructions multiple times.

Programming Loops

```
for i in range(3):
    print(i)
```

```
0
1
2
```

Loops can either run a preset number of times before terminating, or iterate through all items in a list or string, in the case of **for loops**...

Programming Loops

```
count = 0

while count < 3:
    print(count)
    count += 1
```

```
0
1
2
```

Or run continuously until a predetermined trigger state is achieved, in the case of **while loops**.

This Workshop's Topics

New topics for this session:

- Functions

Functions

```
>>> print("Print() is a built-in function in Python.")
Print() is a built-in function in Python.
```

Programming functions are defined instructions that can be used multiple times in the same code.

Functions

```
>>> print("Print() is a built-in function in Python.")
Print() is a built-in function in Python.
```

If a specific set of instructions needs to be run multiple times in the same code, it is more efficient to create a function out of those instructions rather than input the same instructions again.

Functions

```
>>> print("Print() is a built-in function in Python.")
Print() is a built-in function in Python.
```

In fact, we've been making use of the most common Python function this whole time: the **print** function.

Functions

```
>>> print("Print() is a built-in function in Python.")
Print() is a built-in function in Python.
```

Any arguments that the function requires to execute properly (if any) are provided inside the parentheses when calling (running) the function.

Functions

```
def my_greeting(name):
    print('Greetings, %s!' % name)
```

To create a function, start with def, then the name of the function, then a pair of parentheses, with any arguments required inside the parentheses, then a colon.

Functions

```
def my_greeting(name):
    print('Greetings, %s!' % name)
```

Since a function is a code block, on the next line, indent (4 spaces), then input the function's instructions. If the instructions span multiple lines, each line must be indented. Any argument variables should be included in the instructions.

Functions

```
def my_greeting(name):
    print(f"Greetings, {name}!")


my_greeting("theshyhat")
```

```
Greetings, theshyhat!
```

Here, our example function is defined, then the function is called, with the corresponding output shown thereafter.

The Return Statement

```
def sales_tax_calc(amount):
    sales_tax = 1.15
    return amount * sales_tax

sales_tax_calc(55.00)
```

Return is an instruction that can be used in functions. When used, **return** stores the value specified, but doesn't print it out to the console.

The Return Statement

```
def sales_tax_calc(amount):
    sales_tax = 1.15
    return amount * sales_tax

print(sales_tax_calc(55.00))
```

```
63.25
```

So the **return** portion of a function is useful for passing values to other parts of the program, but won't output to the console unless we use the **print** function with it.

Setting Variables Inside of Functions

```
name = "John"
def name_set(new_name):
    name = new_name
    print(name)
name_set("Betty")
print(name)
```

By default, if a function uses a variable which exists outside of the function, any modification to the value of that variable will revert after the function finishes execution.

Setting Variables Inside of Functions

```
def pi_number():
    pi_num = 3.14
    return
pi_number()
print(pi_num)
```

In fact, any variables defined inside of functions are not accessible by the rest of the code. The above code results in an error:
NameError: name 'pi_num' is not defined

Functions with Parameters

```
def add2numbers(a, b):
    return a + b
x = add2numbers(1, 2)
print(x)
```

Functions can be written with parameters, which are used inside of the function's code block

Functions with Default Parameters

```
def greet_user(first_name='John', last_name='Doe'):
    print(f'Hello {first_name} {last_name}!')

greet_user(last_name='Smith')
greet_user()
```

Functions can also be written with default parameters, which are substituted in if the function is called without specifying that parameter.

Programming Functions Exercise

Let's practice using functions with Python at the following URL:

https://learnpython.org/en/Functions

Functions Quiz (1 of 3)

What will print to the console?

```
def add_num(a, b):
   return a + b

add_num(2, 3)
```

A) The number 5
B) There is a syntax error in the code
C) Nothing will be printed to the console

Functions Quiz (1 of 3)

What will print to the console?

```
def add_num(a, b):
   return a + b

add_num(2, 3)
```

A) The number 5
B) There is a syntax error in the code
C) Nothing will be printed to the console

Functions Quiz (2 of 3)

What will print to the console?

```
my_number = 7
def number_set(num):
   my_number = num
number_set(14)
print(my_number)
```

A) The number 7
B) The number 14

Functions Quiz (2 of 3)

What will print to the console?

```
my_number = 7
def number_set(num):
   my_number = num
number_set(14)
print(my_number)
```

A) The number 7
B) The number 14

Functions Quiz (3 of 3)

In Python, what does it mean to “call a function”? Choose two of the following four options.

A) To define a function and its parameters
B) To execute the code within a function
C) To communicate with a function by phone
D) To invoke a function by using its name followed by parentheses

Functions Quiz (3 of 3)

In Python, what does it mean to “call a function”? Choose two of the following four options.

A) To define a function and its parameters
B) To execute the code within a function
C) To communicate with a function by phone
D) To invoke a function by using its name followed by parentheses

Workshop Review Exercise

Before we finish, let's a write a program which features many of the concepts we learned in this workshop.

We can use the online-python.com website to write the program

Workshop Review Exercise

The program should have two lists.

One with four colors: blue, red, green, and yellow.

Another with five fruits: apple, orange, grape, banana, and watermelon.

Workshop Review Exercise

The program should have a function which loops through a list, printing out each item.

The function should be called twice, once for each list.

Workshop Review Exercise

- One list with colors: blue, red, green, and yellow

- Another list with five fruits: apple, orange, grape, banana, and watermelon

- A function which iterates and prints out each item in a list

- Call the function once for each list

Workshop Review Exercise

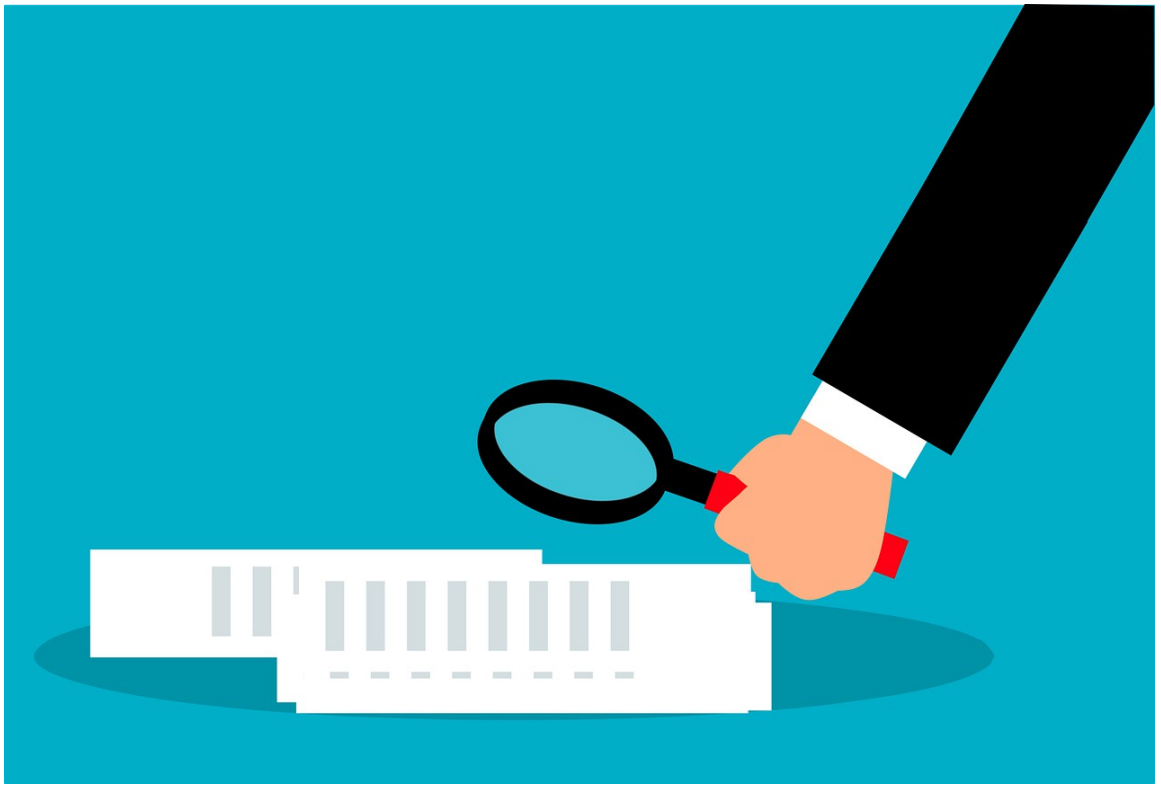
The following code will fulfill the exercise requirements:

```
colors = ['blue', 'red', 'green', 'yellow']
fruits = ['apple', 'orange', 'grape', 'banana', 'watermelon']

def list_out(target):
  for i in target:
    print(i)

list_out(colors)
list_out(fruits)
```

Summary



Let's review the programming concepts we learned in this workshop:

Functions

```
def greetings():
    print('Hello and good day!')

greetings()
```

```
Hello and good day!
```

Functions are defined blocks of code that can be executed multiple times in the same program.

Functions

```
def greetings():
    print('Hello and good day!')

greetings()
```

```
Hello and good day!
```

Executing a function is usually referred to as “calling a function”, or “a function call”.

What's Next?

In the next HackerFrogs Afterschool programming workshop, we'll conclude our intro to Python with the learnpython.org website.

What's Next?

Next workshop topics:

- Classes and Objects

Extra Credit

Looking for more study material on this workshop's topics?

See this video's description for links to supplemental documents and exercises!

Until Next Time, HackerFrogs!