HackerFrogs Afterschool
Python Programming Basics: Part 5

Class:
Programming

Workshop Number:
AS-PRO-PY-05

Document Version:
1.0

Special Requirements:
Completion of AS-PRO-PY-04

What We Learned Before

This workshop is the fourth and final class for intro Python programming.

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

Dictionaries

```
country_capitals = {"England":"London", "Canada":"Ottawa"}
```

Dictionaries in Python are similar to other container objects, such as lists, except that data stored in dictionaries are pairs of keys / values, as opposed to individual items.

Dictionaries

```
john_vital_stats = {
   "height": 170,
   "weight": 90.2,
   "age": 25,
   "gender": "male"
}
```

Dictionaries are useful for recording different key / value pairs that all pertain to one subject, or one common value across a number of subjects.

This Workshop's Topics

New topics for this session:

Modules and Packages

Part 2: Modules

A module is a piece of software which serves a specific purpose. Modules typically contain a number of functions, classes, variables, and / or methods which can be imported into a Python **namespace** to make use of extended functionality.

What are Namespaces?

A namespace is a collection of defined symbolic names and their corresponding values. Every Python program exists within a namespace, and importing modules into a namespace adds objects from the module into the current namespace.

Modules

```
import random
num_list = [1,2,3,4,5,6,7]
random.shuffle(num_list)
print(num_list)
```

```
[6, 4, 3, 5, 1, 2, 7]
```

How can modules enhance our Python programs? As an example, the Python **random** module contains a few different methods for generating random numbers or rearranging pre-existing data into random configurations.

Importing Modules

```
import random
num_list = [1,2,3,4,5,6,7]
random.shuffle(num_list)
print(num_list)
```

```
[6, 4, 3, 5, 1, 2, 7]
```

In order to use module functionality, the module must be imported into the namespace. This can be done in one of two ways, either by importing the whole module into the namespace, as shown above.

Importing Modules

```
from random import shuffle
num_list = [1,2,3,4,5,6,7]
shuffle(num_list)
print(num_list)
```

```
[5, 7, 2, 6, 1, 3, 4]
```

Or we can specify which methods or functions to import from a specific module, like the example shown here.

Importing Modules

```
from random import shuffle
num_list = [1,2,3,4,5,6,7]
shuffle(num_list)
print(num_list)
```

```
[5, 7, 2, 6, 1, 3, 4]
```

Note that when we import specific functions from a module, we can access those functions without specifying the module name, like we did in the first example (random.shuffle).

Custom Name Module Import

```
import random as rng_functions
num_list = [1,2,3,4,5,6,7]
rng_functions.shuffle(num_list)
print(num_list)
```

```
[5, 2, 3, 1, 4, 7, 6]
```

When importing modules into Python code, we can setup a custom name for the module within the code, either to shorten the module name, change it so that it doesn't overlap with another module in our namespace, or any other reason.

Creating a Module

```
def default_greeting():
    print('Hello, friend! How are you?')
```

Writing a module requires the creation of a **.py** file with the same name as the name of the module we want to write. Here we've created a file named **greetings.py** with the code shown above.

Creating a Module

```
>>> import greetings
>>> greetings.default_greeting()
Hello, friend! How are you?
```

Provided we run a Python interactive console from the same directory containing the **greetings.py** file, we can import the file as a module and use the function defined inside that file, just like any other module.

Module Initialization

```
import random
print(id(random.shuffle))
import random
print(id(random.shuffle))
```

```
140299445739456
140299445739456
```

When a module is imported, all objects within the module are initialized once, but if the same objects are imported a second time (for whatever reason), they will not be initialized again.

The Dir Function

```
print(dir(random))
```

```
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCO
'__loader__', '__name__', '__package__', '__spec__', '_accumulate',
'_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', '
'normalvariate', 'paretovariate', 'randint', 'random', 'randrange',
```

The **dir** function, when used with a module name, returns all name of attributes, methods, and variables in the module.

The Help Function

```
help(random.randint)
```

```
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

The help function can be used with an object to return the help text associated with that object.

Modules and Packages Exercise

Let's practice importing modules with Python at the following URL:

https://bellard.org/jslinux/

Modules and Packages Quiz Q1

When importing the **randint** function from the **random** module, which method adds the most objects to the namespace?

A) `import random`
B) `from random import *`
C) `from random import randint`

Modules and Packages Quiz Q1

When importing the **randint** function from the **random** module, which method adds the most objects to the namespace?

A) `import random`
B) `from random import *`
C) `from random import randint`

Modules and Packages Quiz Q2

```
import random
```

How do we access the randint function in code?

A) `random(randint())`
B) `random.randint()`
C) `randint().random`

Modules and Packages Quiz Q2

```
import random
```

How do we access the randint function in code?

A) `random(randint())`
B) `random.randint()`
C) `randint().random`

Modules and Packages Quiz Q3

In which directory does Python look for modules?

A) /usr/local/lib/python (Linux)
B) C:\Python\Lib (Windows)
C) the directory where Python program is run
D) any of the above

Modules and Packages Quiz Q3

In which directory does Python look for modules?

A) /usr/local/lib/python (Linux)
B) C:\Python\Lib (Windows)
C) the directory where Python program is run
D) any of the above

Workshop Review Exercise

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

A board game player wants to write a program which simulates rolling dice.

We can write the program on online-python.com

Workshop Review Exercise

The program should have a function which takes an integer as an argument which represents the number of dice to be rolled. A random number between 1 and 6 equal to the argument, adds them up, and returns the total.

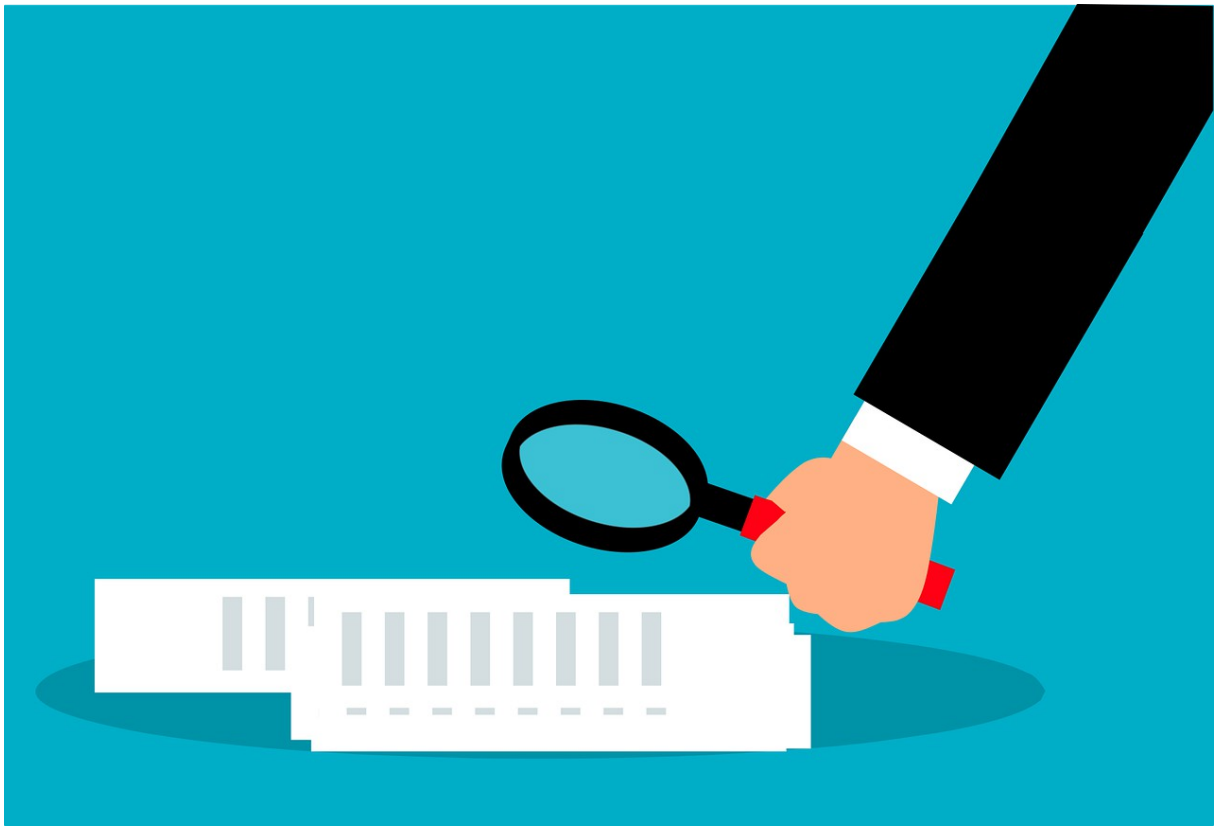
The program should use the **random** Python module

Workshop Review Exercise

The following code is a possible solution to the challenge:

```
import random
def roll_dice(num_rolls):
  total = 0
  for _ in range(num_rolls)
    result = random.randint(1, 6)
    total += result
  return total
```

Summary



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

Modules and Packages

```
>>> import greetings
>>> greetings.default_greeting()
Hello, friend! How are you?
```

Modules and packages are pre-written code that can be imported into Python files to add functionality.

What's Next?

This is the end of the HackerFrogs AfterSchool intro to Python programming classes, but now that we know the basics we can start learning a lot more about coding in this extremely popular programming language!

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!