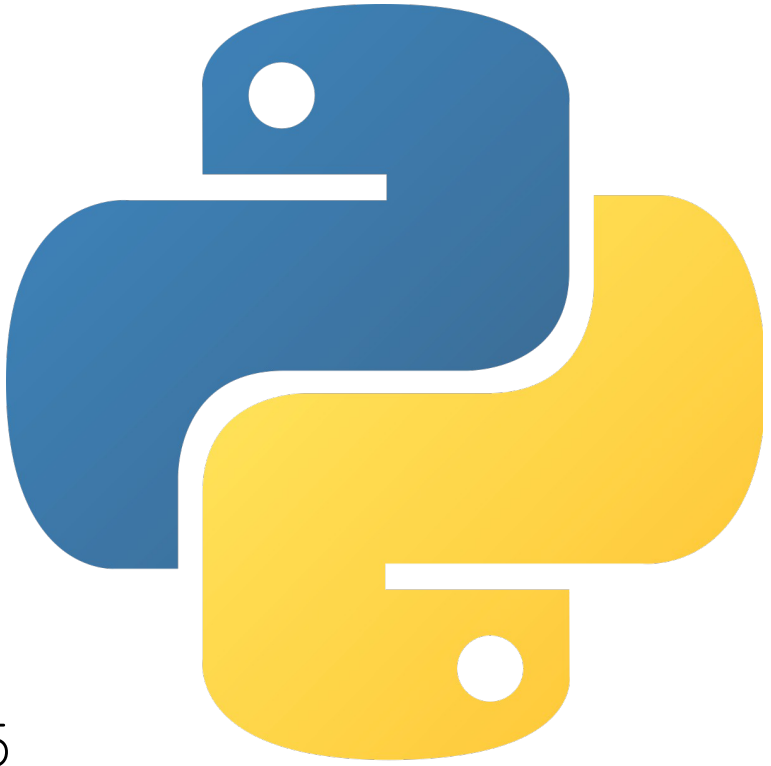
HackerFrogs Afterschool
Python Programming Basics: Part 6

Class:
Programming (Python)

Workshop Number:
AS-PRO-PY-06

Document Version:
1.2

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:

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”.

This Workshop's Topics

New topics for this session:

- Classes and Objects

- Dictionaries

Part 3: Objects and Classes

```
my_integer = 1
my_float = 0.25
my_string = "hackerFrogs"

print(type(my_integer),type(my_float),type(my_string))
```

```
(<class 'int'>, <class 'float'>, <class 'str'>)
```

Python is an object-oriented programming (OOP) language, and as such, all pieces of data in Python code are considered objects.

Classes

```
my_integer = 1
my_float = 0.25
my_string = "hackerFrogs"

print(type(my_integer),type(my_float),type(my_string))
```

```
(<class 'int'>, <class 'float'>, <class 'str'>)
```

In addition, all objects in Python are separated into different classes. We see that the three variables here are in the **integer**, **float**, and **string** classes, respectively.

Classes

```
my_string = "The hackerFrogs"

print(len(my_string))
print(my_string.upper())
```

```
15
THE HACKERFROGS
```

An object's class indicates which built-in functions, methods and variables it has.

Classes

```
my_string = "The hackerFrogs"

print(len(my_string))
print(my_string.upper())
```

```
15
THE HACKERFROGS
```

Here we've executed a function and method on the **my_string** variable. Both the **len** function and **upper** method are unique to the string class.

Classes

```
my_string = 1337

print(my_string.upper())
```

```
AttributeError: 'int' object has no attribute 'upper' on line 3 in main.py
```

If we changed the **my_string** value to an integer, executing either the **len** function or **upper** method would result in an error message.

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

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

Dictionaries can be identified by a pair of curly braces, within which are pairs of keys / values, each pair separated by commas. The key and value are themselves separated by a colon.

Dictionaries

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

Dictionaries can also be initialized in the format above, which is easier-to-read.

Iterating Over Dictionaries

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

for i, j in country_capitals.items():
  print(i, j)
```

```
England London
Canada Ottawa
```

Items in a dictionary can be iterated over using a for loop, but we must use the **items** method, which renders the dictionary as a list with tuples as the content, which hold the key / value pairs.

Adding Keys / Values

```
friend_city = {'Carl': 'Toronto', 'Rachel': 'New York'}
friend_city['Bobby'] = 'Texas'
print(friend_city)
```

```
{'Carl': 'Toronto', 'Rachel':
'New York', 'Bobby': 'Texas'}
```

To add entries to an existing dictionary, simply define the dictionary's index with the name of the key, then supply the value on the other side of the equals sign, as shown above.

Removing Keys / Values

```
friend_city = {'Carl': 'Toronto', 'Rachel': 'New York'}
del friend_city['Carl']
print(friend_city)
```

```
{'Rachel': 'New York'}
```

To remove entries from a dictionary, there are two ways. The first way is to use the **del** keyword along with the dictionary key as the index name.

Removing Keys / Values

```
friend_city = {'Carl': 'Toronto', 'Rachel': 'New York'}
friend_city.pop('Rachel')
print(friend_city)
```

```
{'Carl': 'Toronto'}
```

The other way of removing entries from dictionaries is to the **pop** dictionary method, as illustrated above.

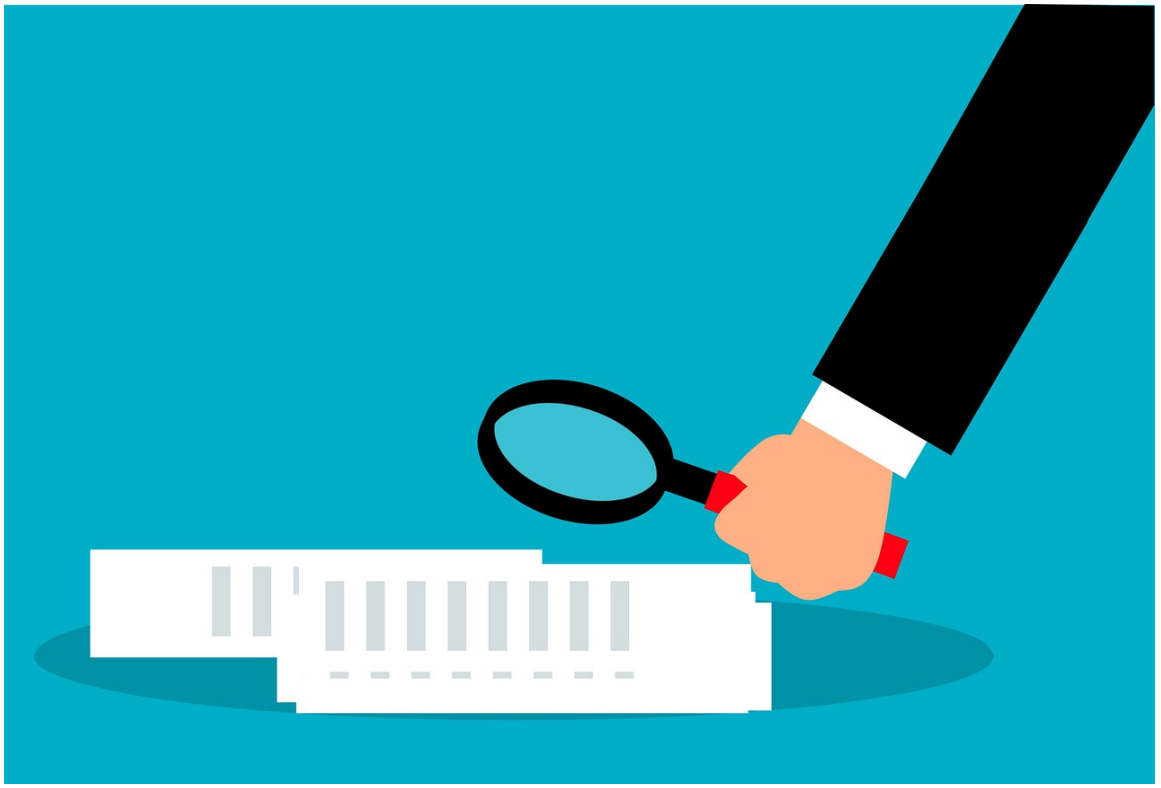
Dictionary Keys are Immutable

```
friend_city = {'Carl': 'Toronto', 'Rachel': 'New York'}
friend_city['Carl'] = 'Kelly'
print(friend_city)
```

```
{'Carl': 'Kelly', 'Rachel': 'New York'}
```

The names of keys cannot be changed after they've been created. One the key's value can be changed.

Summary



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

Classes & Objects

```
print(type(1), type('word'))
```

```
<class 'int'> <class 'str'>
```

Python is an object-oriented programming language, and as such all data in Python code are considered objects.

Classes & Objects

```
print(type(1), type('word'))
```

```
<class 'int'> <class 'str'>
```

All objects belong to a class, and an object's class indicates which built-in functions, methods, and variables it has.

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.

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:

- Program Writing Practice

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!