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

```python
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

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

greetings()
```

```
Hello and good day!
```

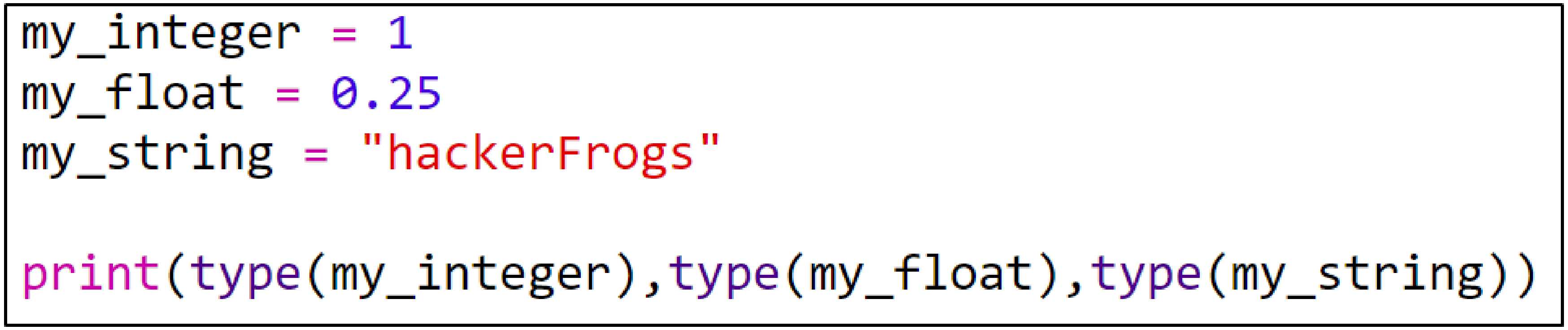
Executing a function is usually referred to as “calling a function”, or “a function call”.

This Workshop's Topics

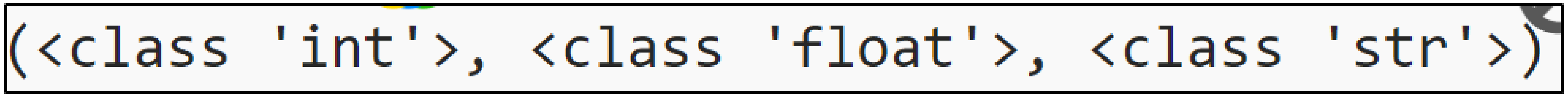
New topics for this session:

- Classes and Objects

Part 3: Objects and Classes



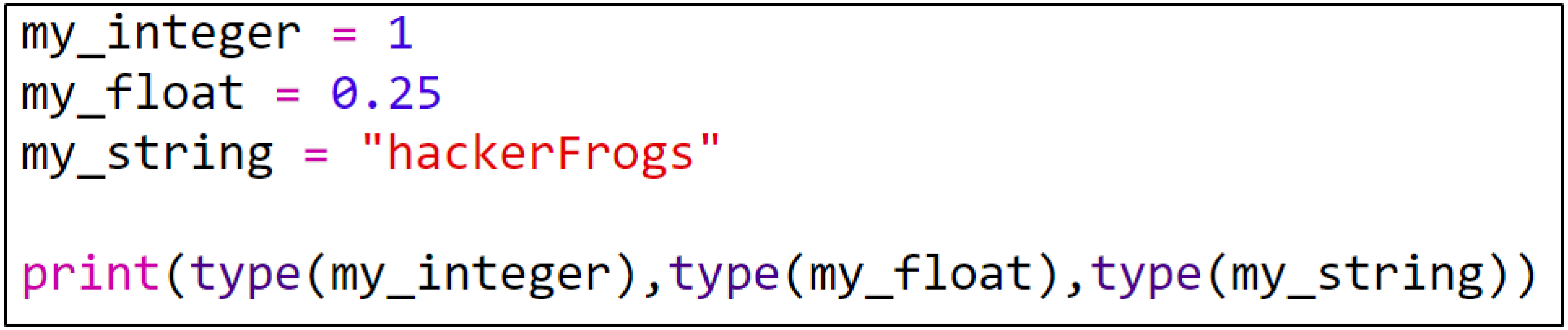
```python
my_integer = 1
my_float = 0.25
my_string = "hackerFrogs"

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



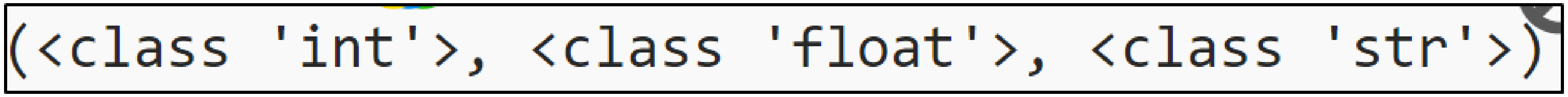
```
(<class 'int'>, <class 'float'>, <class 'str'>)
```

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

Classes



```python
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.

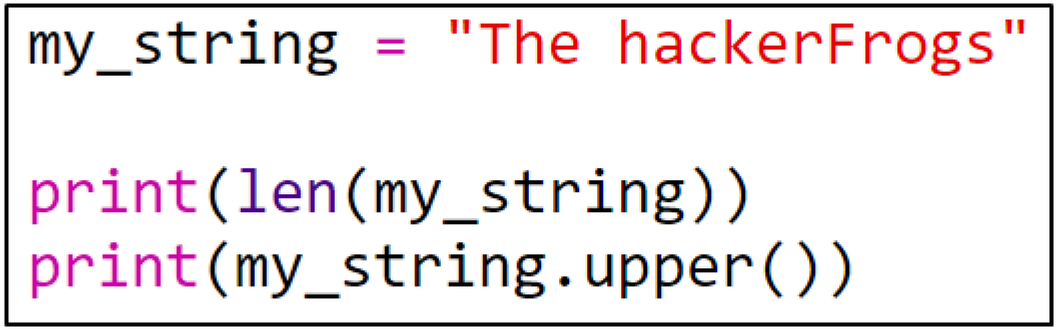
Classes

```python
my_string = "The hackerFrogs"

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

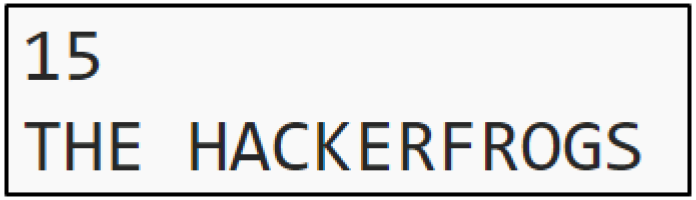
```
15
THE HACKERFROGS
```

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

Classes



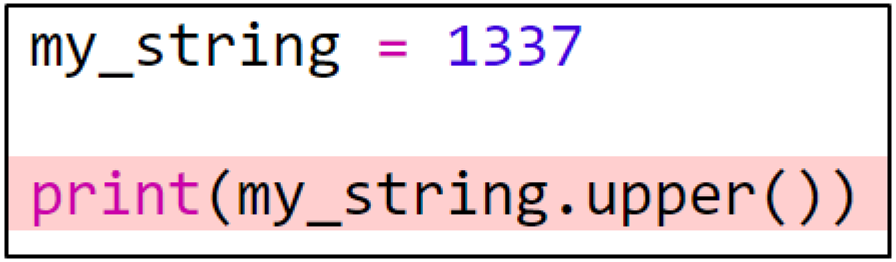
```
my_string = "The hackerFrogs"

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



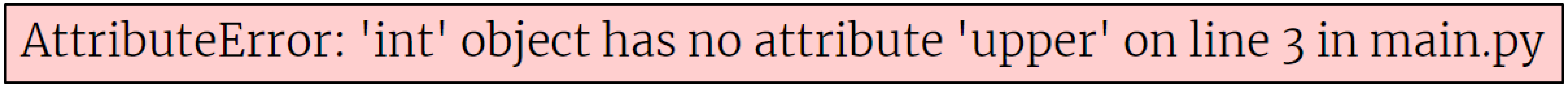
```
15
THE HACKERFROGS
```

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



```python
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.

Creating Classes

```
class Student:
    def __init__(self, name, fave_subject):
        self.name = name
        self.fave_subject = fave_subject
```

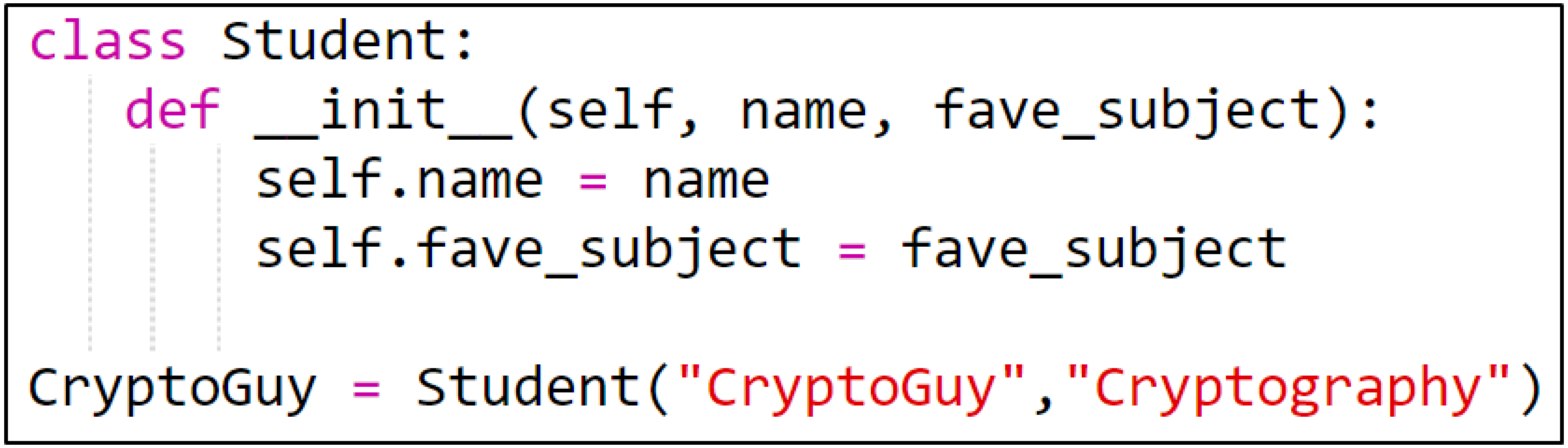
To create a class, we begin with class, then the name of the class, then a colon. The first letter of the class name is capitalized.

Creating Classes

```
class Student:
    def __init__(self, name, fave_subject):
        self.name = name
        self.fave_subject = fave_subject
```

On the next line, indent (4 spaces), then def, then __init__, a pair of parentheses, then inside the parentheses, self, then any other attributes the class requires, then a colon.

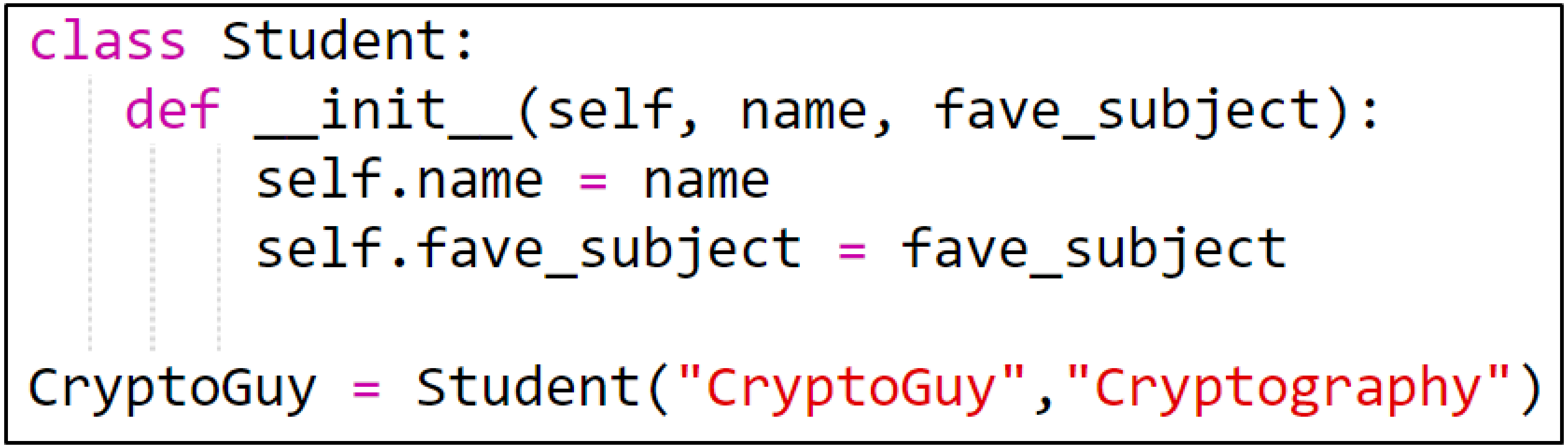
The __init__ Function



```python
class Student:
    def __init__(self, name, fave_subject):
        self.name = name
        self.fave_subject = fave_subject

CryptoGuy = Student("CryptoGuy","Cryptography")
```

One important feature of class creation is defining the **__init__** function, which specifies the attributes of the class.

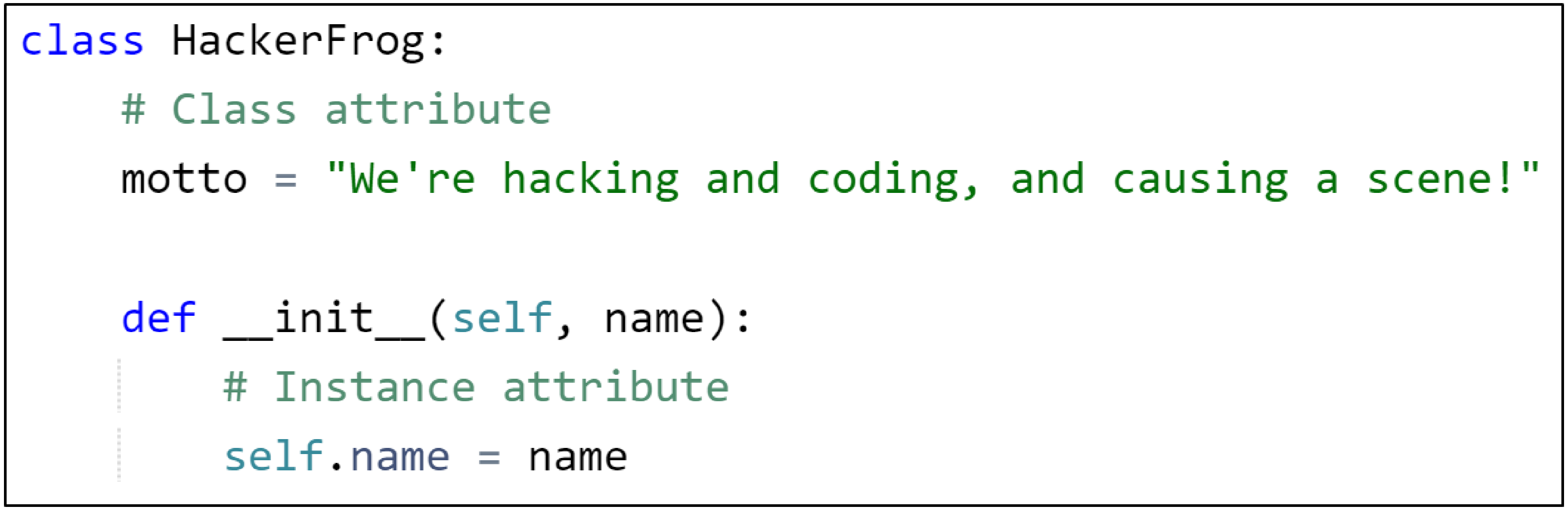
The Class Self Attribute



```python
class Student:
    def __init__(self, name, fave_subject):
        self.name = name
        self.fave_subject = fave_subject

CryptoGuy = Student("CryptoGuy","Cryptography")
```

One attribute which all classes share is the **self** attribute, which is used to access other attributes and methods of the class in the code.

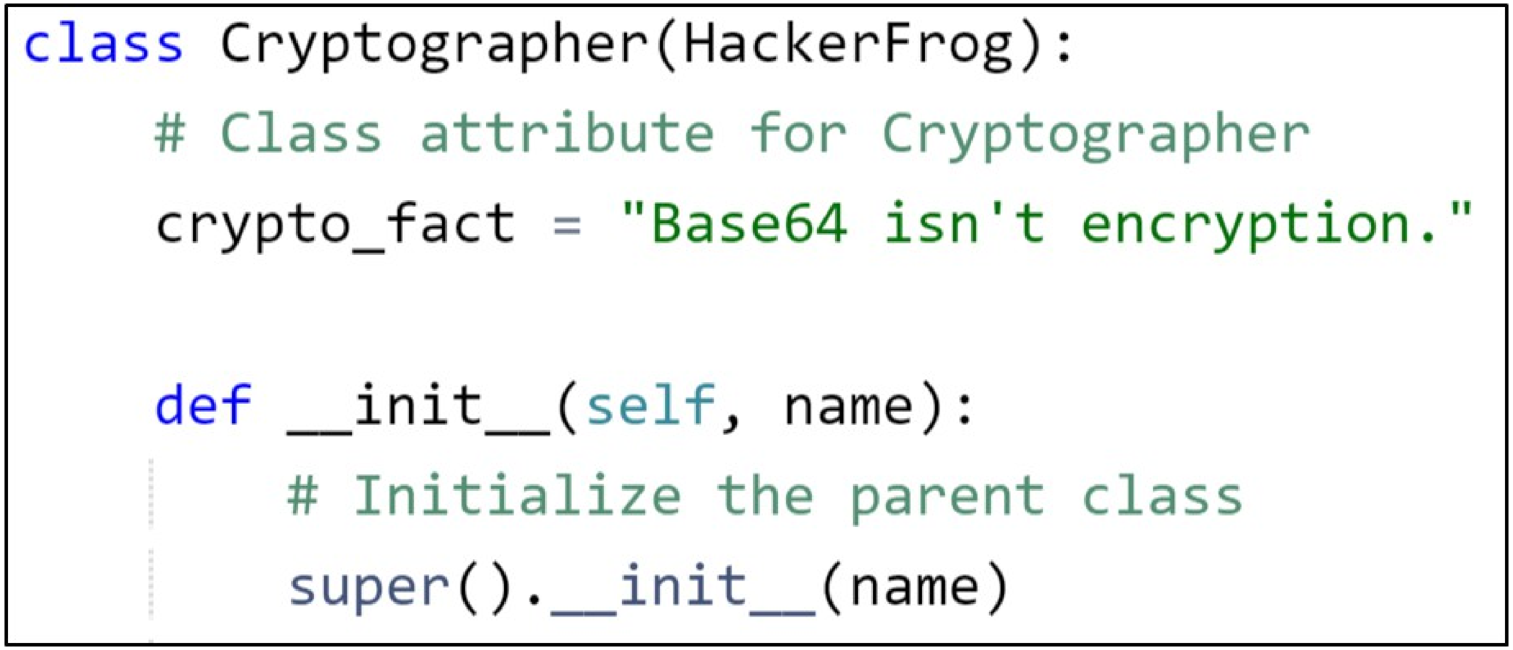
Derived Classes (Subclasses)



```python
class HackerFrog:
    # Class attribute
    motto = "We're hacking and coding, and causing a scene!"

    def __init__(self, name):
        # Instance attribute
        self.name = name
```

Derived classes are classes that inherit properties and methods from another class, called the base class or superclass.

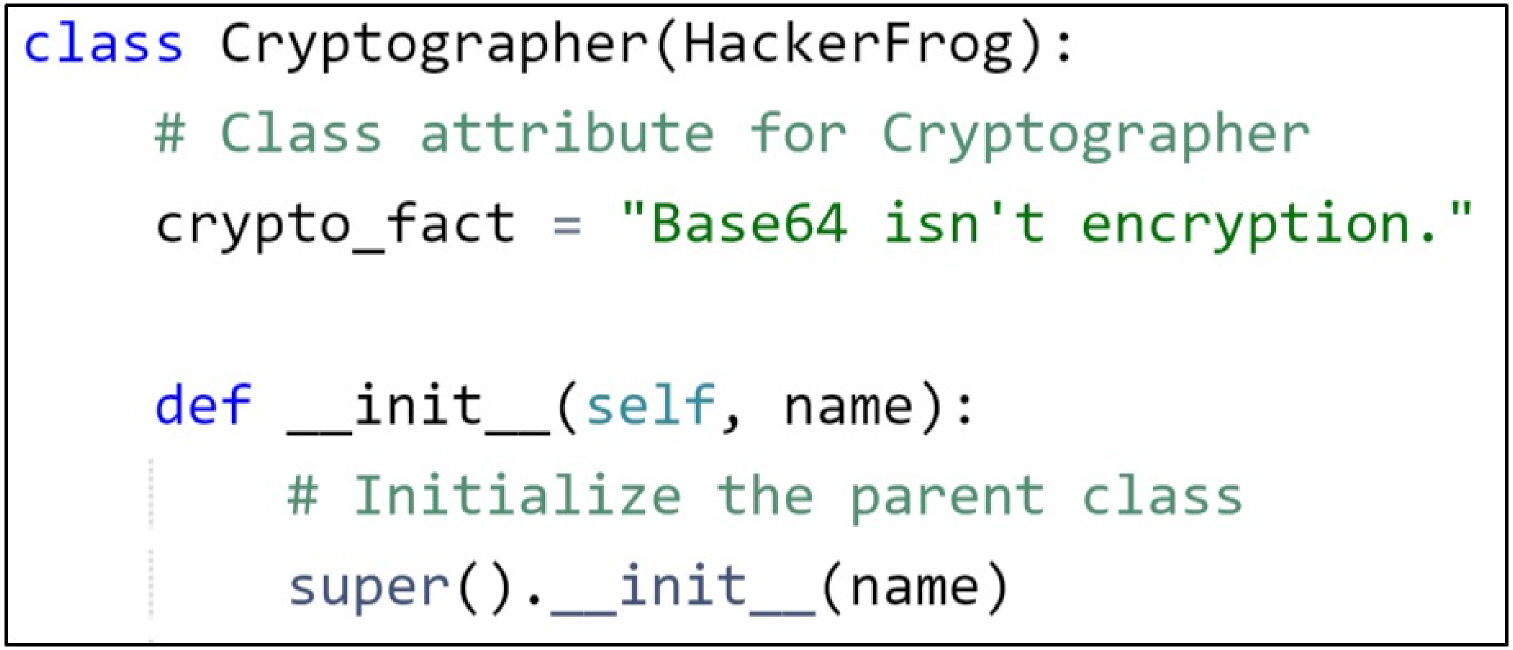
Derived Classes (Subclasses)



```python
class Cryptographer(HackerFrog):
    # Class attribute for Cryptographer
    crypto_fact = "Base64 isn't encryption."

    def __init__(self, name):
        # Initialize the parent class
        super().__init__(name)
```

Here the Cryptographer class is derived from the HackerFrog class, so it inherits the motto from its superclass.

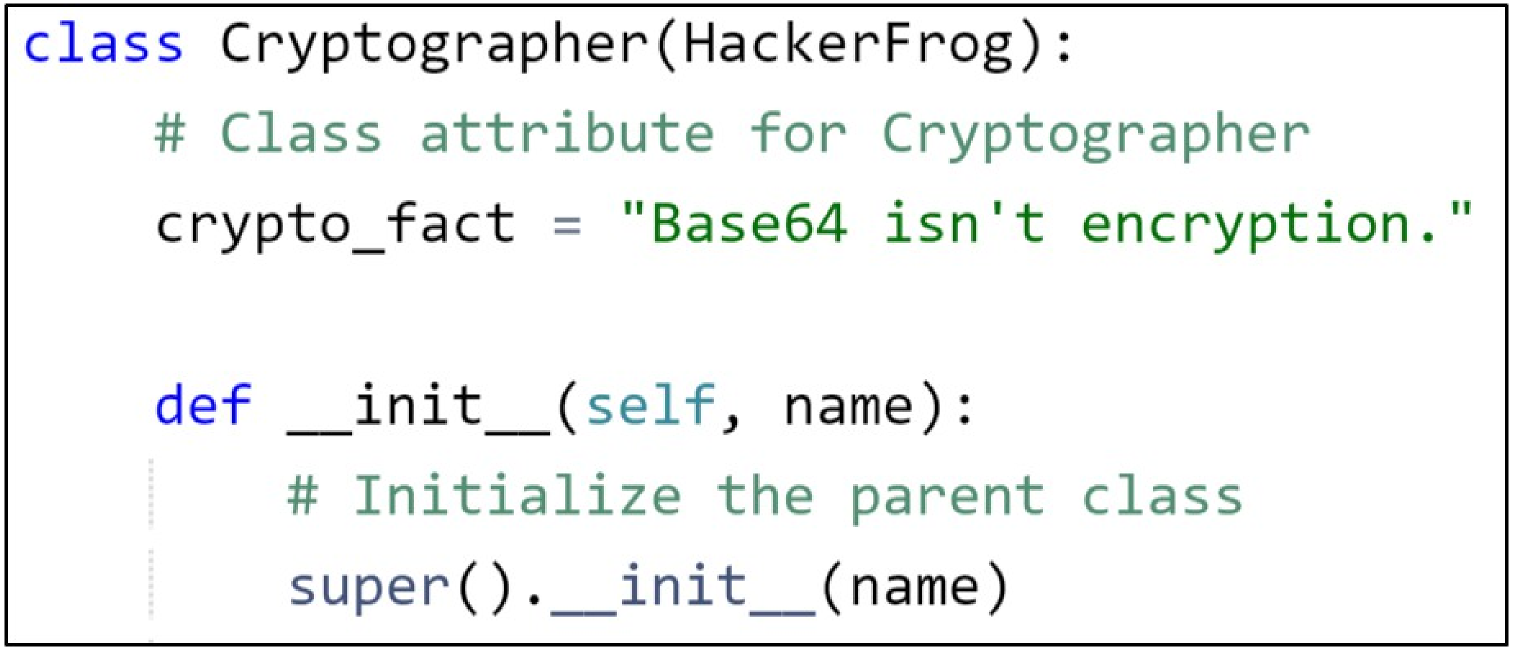
Derived Classes (Subclasses)



```python
class Cryptographer(HackerFrog):
    # Class attribute for Cryptographer
    crypto_fact = "Base64 isn't encryption."

    def __init__(self, name):
        # Initialize the parent class
        super().__init__(name)
```

Here the Cryptographer class is derived from the HackerFrog class, so it inherits the motto attribute from its superclass.

Derived Classes (Subclasses)



```python
class Cryptographer(HackerFrog):
    # Class attribute for Cryptographer
    crypto_fact = "Base64 isn't encryption."

    def __init__(self, name):
        # Initialize the parent class
        super().__init__(name)
```

Here the Cryptographer class is derived from the HackerFrog class, so it inherits the motto attribute from its superclass.

Classes and Objects Exercise

Let's practice using classes and objects with Python at the following URL:

https://learnpython.org/en/Classes_and_Objects

Classes and Objects Quiz (1 of 2)

Which of the following is statements regarding the relationship between a class and an instance is true?

A) A class is a blueprint for creating objects, an instance is an object belonging to a class

B) Instances of a class have all of the attributes and methods of their class

C) The class variables of instances and their associated class can be different

D) All of the above

Classes and Objects Quiz (1 of 2)

Which of the following is statements regarding the relationship between a class and an instance is true?

A) A class is a blueprint for creating objects, an instance is an object belonging to a class

B) Instances of a class have all of the attributes and methods of their class

C) The class variables of instances and their associated class can be different

D) All of the above

Classes and Objects Quiz (2 of 2)

Which of the following is not considered objects in the Python programming language?

A) Operators, like +, -, /, *, etc, etc

B) Variables, such as integers, lists, strings, etc, etc

C) Keywords, like **if**, **else**, **for**, **while**, etc, etc

D) A and C

Classes and Objects Quiz (2 of 2)

Which of the following is not considered objects in the Python programming language?

A) Operators, like +, -, /, *, etc, etc

B) Variables, such as integers, lists, strings, etc, etc

C) Keywords, like **if**, **else**, **for**, **while**, etc, etc

D) A and C

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 a window in the learnpython.org website to write the program

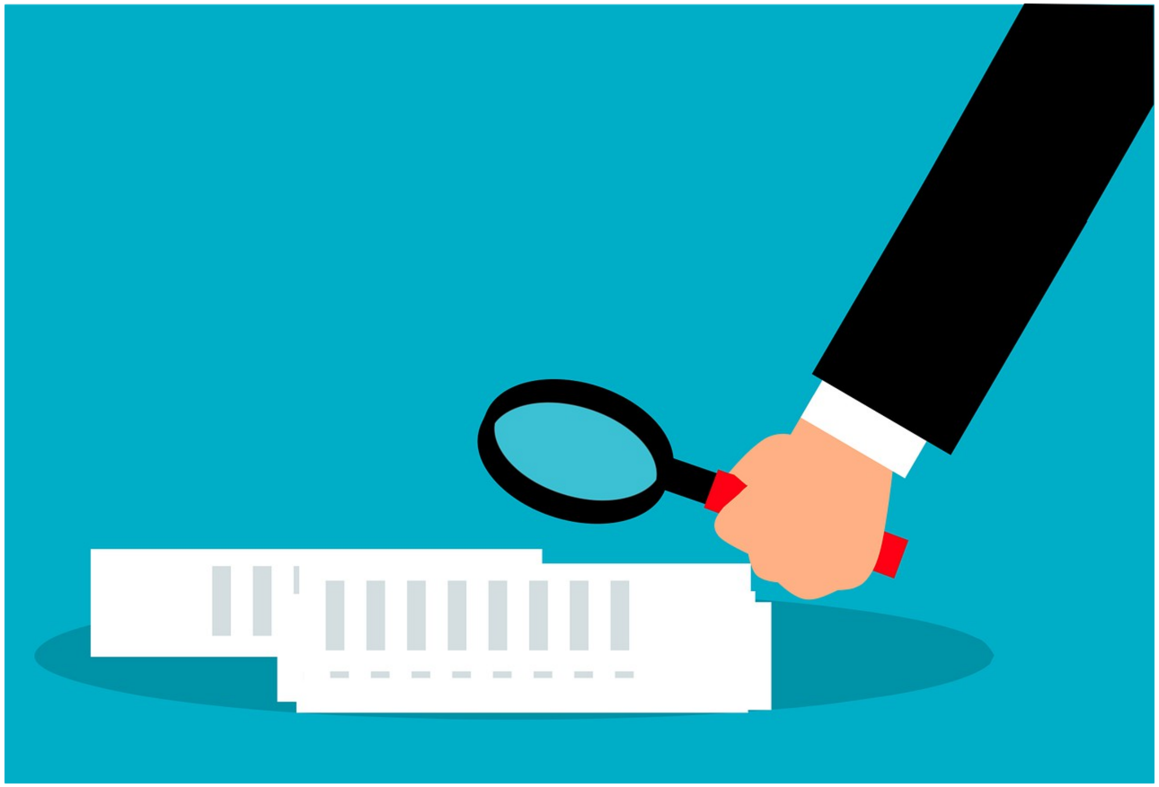
Workshop Review Exercise

The program should create a new class, **person**, with the following attributes: **first_name**, **last_name**, **age**, **birthday**, and **fave_color**. Lastly, we'll create a function with the class, self_intro, which prints out all of the class' attributes in a string.

Workshop Review Exercise

```
class Person:
    def __init__(self, first_name, last_name, age, birthday, fave_color):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.birthday = birthday
        self.fave_color = fave_color

    def self_intro(self):
        print(f"Hello, my name is {self.first_name} {self.last_name}. I am 
{self.age} years old. My birthday is on {self.birthday}. My favorite color is 
{self.fave_color}.")

shyhat = Person("shy", "hat", 1337, "February 29th", "green")
shyhat.self_intro()
```

Summary



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

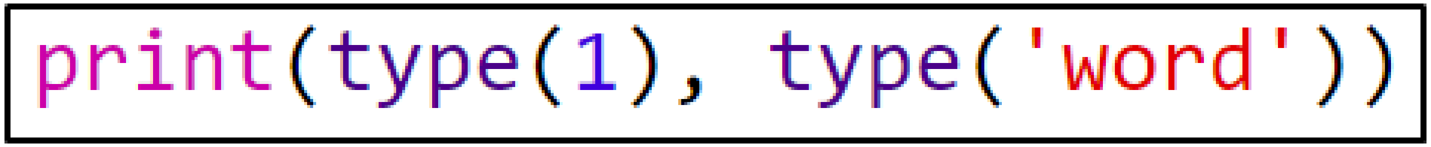
Classes & Objects

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

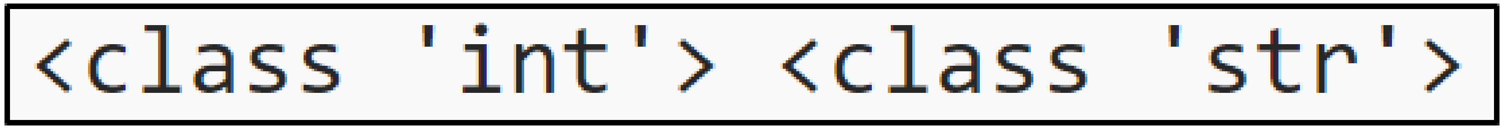
```
<class 'int'> <class 'str'>
```

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

Classes & Objects



```python
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.

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:

- Dictionaries

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!