



# Object Oriented Programming



# Complete Python Bootcamp

- Object Oriented Programming (OOP) allows programmers to create their own objects that have methods and attributes.
- Recall that after defining a string, list, dictionary, or other objects, you were able to call methods off of them with the `.method_name()` syntax.



# Complete Python Bootcamp

These methods act as functions that use information about the object, as well as the object itself to return results, or change the current object.

For example this includes appending to a list, or counting the occurrences of an element in a tuple.



# Complete Python Bootcamp

OOP allows users to create their own objects.

The general format is often confusing when first encountered, and its usefulness may not be completely clear at first.

In general, OOP allows us to create code that is repeatable and organized.



# Complete Python Bootcamp

For much larger scripts of Python code, functions by themselves aren't enough for organization and repeatability.

Commonly repeated tasks and objects can be defined with OOP to create code that is more usable.

Let's check out the syntax.



# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self,param1,param2):
```

```
        self.param1 = param1
```

```
        self.param2 = param2
```

```
    def some_method(self):
```

```
        # perform some action
```

```
        print(self.param1)
```



# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self, param1, param2):
```

```
        self.param1 = param1
```

```
        self.param2 = param2
```

```
    def some_method(self):
```

```
        # perform some action
```

```
        print(self.param1)
```



# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self, param1, param2):
```

```
        self.param1 = param1
```

```
        self.param2 = param2
```

```
    def some_method(self):
```

```
        # perform some action
```

```
        print(self.param1)
```





# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self, param1, param2):
```

```
        self.param1 = param1
```

```
        self.param2 = param2
```

```
    def some_method(self):
```

```
        # perform some action
```

```
        print(self.param1)
```



# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self, param1, param2):
```

```
        self.param1 = param1
```

```
        self.param2 = param2
```

```
    def some_method(self):
```

```
        # perform some action
```

```
        print(self.param1)
```



# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self,param1,param2):
```

```
        self.param1 = param1
```

```
        self.param2 = param2
```

```
    def some_method(self):
```

```
        # perform some action
```

```
        print(self.param1)
```



# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self, param1, param2):
```

```
        self.param1 = param1
```

```
        self.param2 = param2
```

```
    def some_method(self):
```

```
        # perform some action
```

```
        print(self.param1)
```



# Complete Python Bootcamp

```
class NameOfClass():
```

```
    def __init__(self, param1, param2):  
        self.param1 = param1  
        self.param2 = param2
```

```
    def some_method(self):  
        # perform some action  
        print(self.param1)
```



# Complete Python Bootcamp

Let's explore Object Oriented Programming  
in more detail with code!



# **Object Oriented Programming Challenge Solution**