Object-Oriented Programming: Takeaways

Syntax

- Creating a class:

```
class Class:
def __init__(self, team_name):
    self.team_name = team_name
```

- Creating an instance of a class:

```
spurs = Team("San Antonio Spurs")
```

- Defining a class method:

```
@classmethod
def older_team(self, team1, team2):
    return "Not yet implemented"
```

Concepts

- In object-oriented programming, everything is an object. Classes and instances are known as objects and they're a fundamental part of object-oriented programming.
- The special `__init__` function runs whenever a class is instantiated. The `__init__` function can take in parameters, but self is always the first one. Self is just a reference to the instance of the class and is automatically passed in when you instantiate an instance of the class.
- Inheritance enables you you to organize classes in a tree-like hierarchy. Inheriting from a class means that the new class can exhibit behavior of the inherited class but also define its own additional behavior.
- Class methods act on an entire class rather than a particular instance of a class. We often use them as utility functions.
- Overloading is a technique used to modify a inherited class to ensure not all behavior is inherited. Overloading methods gives access to powerful functions without having to implement tedious logic.

Resources

- Object-oriented Programming
- Documentation for classmethod