OBJECT ORIENTED PROGRAMMING IN PYTHON

Inheritance: Is-a Versus Has-A

Vicki Boykis
Senior Data Scientist

Extending our DataShells

DataShell

“I only process CSV datasets”

Inheritance - A class that takes on attributes from another, "parent" class and adds some more of its own functionality.

Two Dinosaurs

Class

```
class Dinosaur():
    pass
```

Objects

```
TRex = Dinosaur()
```

```
TBob = Dinosaur()
```

```
TSue = Dinosaur()
```

Is a and Has a Relationship

- A Pterodactyl **is-a** Dinosaur
- A Tyrannosaurus **is-a** Dinosaur
- Is a Pterodactyl a dinosaur? Yes,pterodactyl inherits from dinosaur.
- Is a Tyranosaurus a pterodactyl? No, but they're both dinosaurs.
- Is a dinosaur a pterodactyl? No, so it doesn't work the other way, either.

Inheriting a DataShell

```
# the class we want to create
StDevDataShell

class StDevDataShell(DataShell):
    pass
```

OBJECT ORIENTED PROGRAMMING IN PYTHON

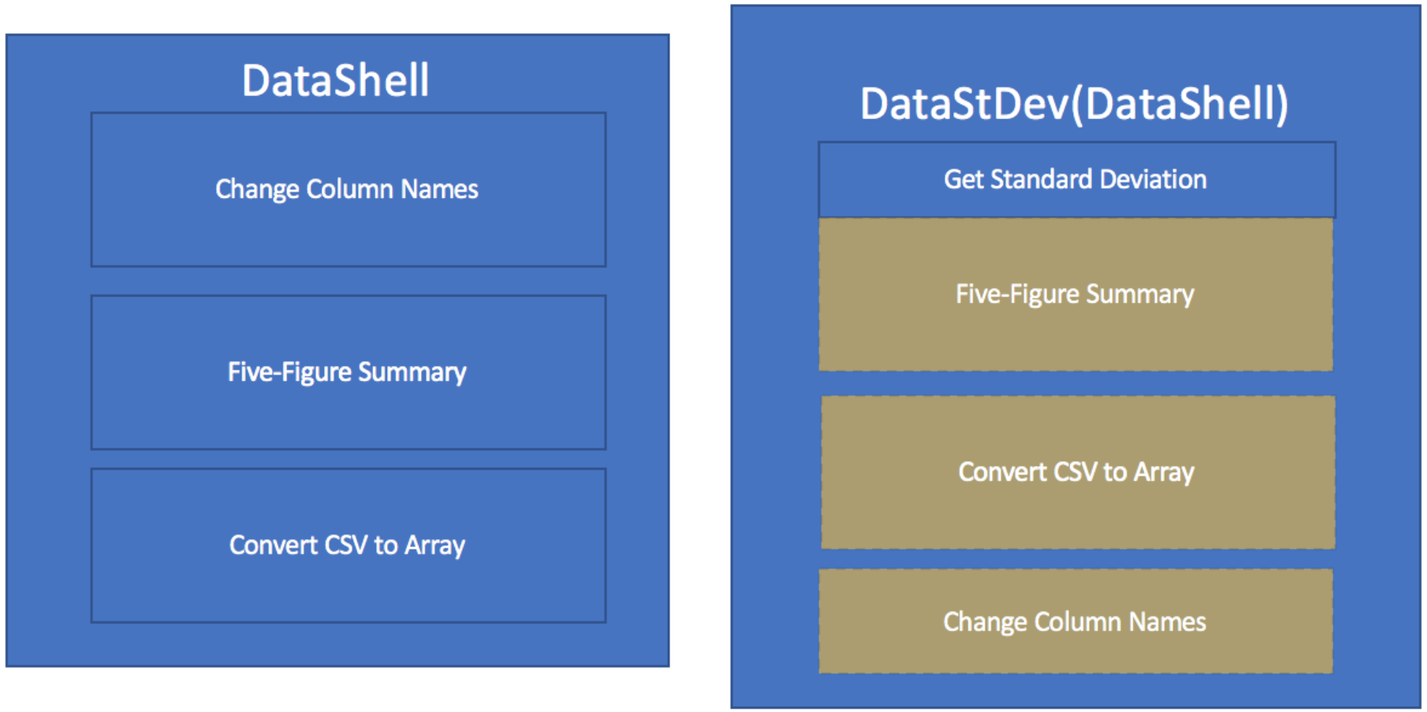
Let's practice!

OBJECT ORIENTED PROGRAMMING IN PYTHON

Inheritance with DataShells

Vicki Boykis
Sr. Data Scientist

DataShell with Standard Deviation



Changing the DataShell

```
class DataShell:

    def __init__(self, filename):
        self.filename = filename

   def create_datashell(self):
        data_array = np.genfromtxt(self.filename, delimiter=',', dtype=None)
        self.array = data_array
        return self.array

    def show_shell(self):
        print(self.array)

    def rename_column(self, old_colname, new_colname):
        for index, value in enumerate(self.array[0]):
            if value == old_colname.encode('UTF-8'):
                self.array[0][index] = new_colname
        return self.array

    def five_figure_summary(self,col_position):
        statistics =  stats.describe(self.array[1:,col_position]
        .astype(np.float))
        return f"Five-figure stats of column {col_position}: {statistics}"
```

Allowing for a standard deviation

```
    def get_stdev(self,col_position):

        column = self.array[1:,col_position].astype(np.float)

        stdev = np.ndarray.std(column,axis=0)
        return f"Standard Deviation of column {col_position}: {stdev}"
```

Inheritance with DataShells

```
class DataStDev(DataShell):
```

```
    def __init__(self,filename):
        DataShell.filename = filename
```

```
    def get_stdev(self,col_position):
        column = self.array[1:,col_position].astype(np.float)
        stdev = np.ndarray.std(column,axis=0)
        return f"Standard Deviation of column {col_position}: {stdev}"
```

Calling our new DataShell

Code to call it:

```
carData = 'mtcars.csv'

myStDevShell = DataStDev(carData)
myStDevShell.create_datashell()
myStDevShell.get_stdev(1)
'Standard Deviation of column 1: 5.932029552301218'
```

OBJECT ORIENTED PROGRAMMING IN PYTHON

Let's practice!

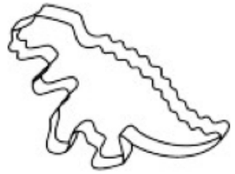
OBJECT ORIENTED PROGRAMMING IN PYTHON

Composition

Vicki Boykis
Senior Data Scientist

Inheritance versus Composition

```
class Dinosaur(object)
```



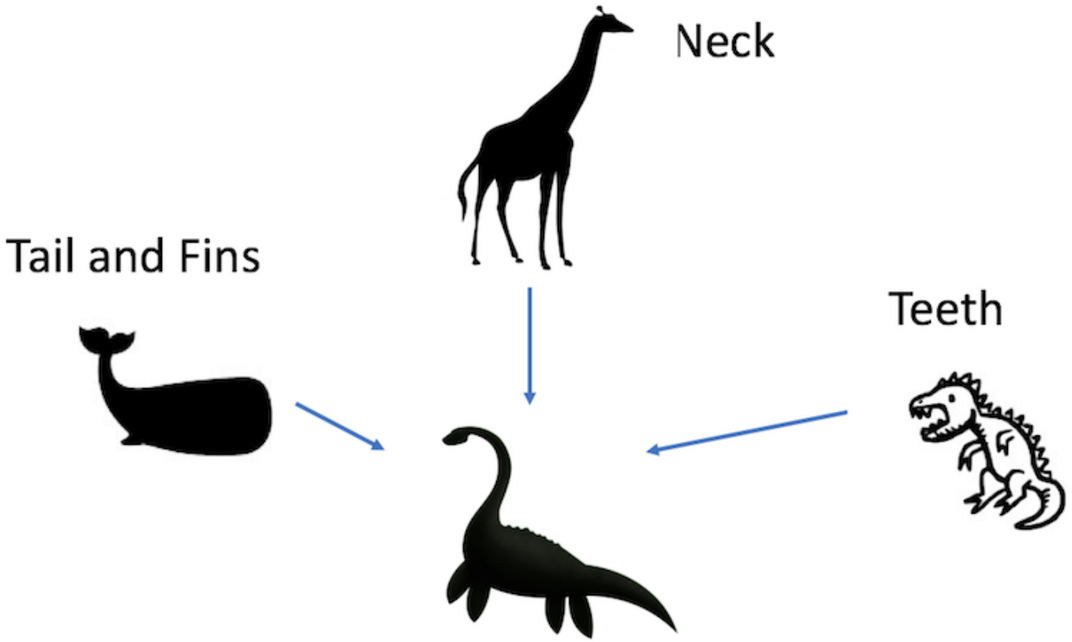
Pterodactyl Tom

```
Class Pterodactyl(Dinosaur)
```

Tyrannosaurus Sue

```
class Tyrannosaurus(Dinosaur)
```

Composing with Animals



Composition In a DataShell - 1

Five-Figure Summary Composition

```
def five_figure_summary(self,col_position):
    statistics =  stats.describe(self.array[1:,col_position].astype(np.float))
    return f"Five-figure stats of column {col_position}: {statistics}"
```

Composition In a DataShell - 2

Create DataShell Composition:

```
def create_datashell(self):
    data_array = np.genfromtxt(self.filename, delimiter=',', dtype=None)
    self.array = data_array
    return self.array
```

Composing with Pandas

Create DataShell Composition:

```
class DataShell:
    def __init__(self, filename):
        self.filename = filename

    def create_datashell(self):
        data_array = np.genfromtxt(self.filename, delimiter=',', dtype=None)
        self.array = data_array
        return self.array
```

```
class DataShellComposed:
    def __init__(self, filename):
        self.filename = filename

    def create_datashell(self):
        self.df = pandas.read_csv()
        return self.df
```

What does our new class look like?

```
carData = 'mtcars.csv'
myDatashell = DataShellComposed(carData)
myDatashell.create_datashell()
print(type(myDatashell.df))
<class 'pandas.core.frame.DataFrame'>
```

OBJECT ORIENTED PROGRAMMING IN PYTHON

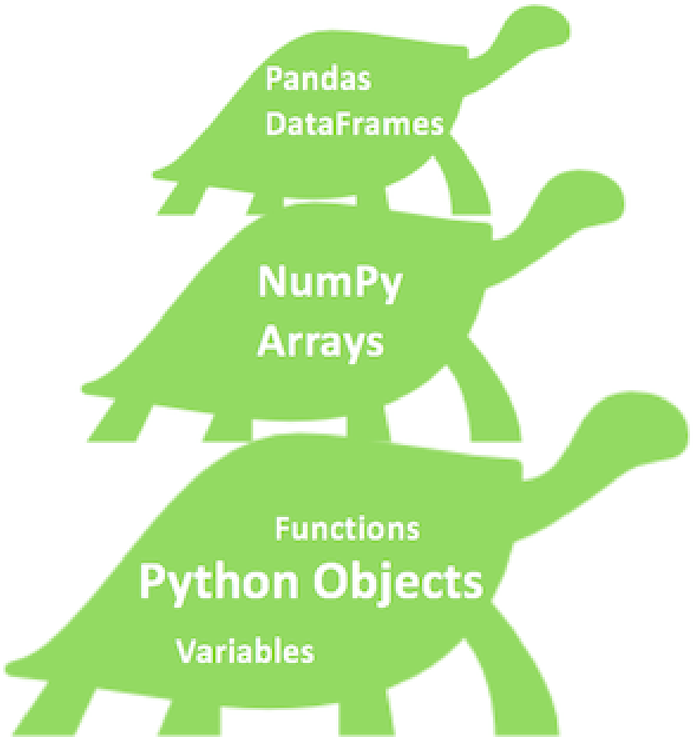
Let's practice!

OBJECT ORIENTED PROGRAMMING IN PYTHON

Wrapping Up OOP

Vicki Boykis
Senior Data Scientist

Understanding Objects



Understanding Classes and Instances of Classes.

Classes:

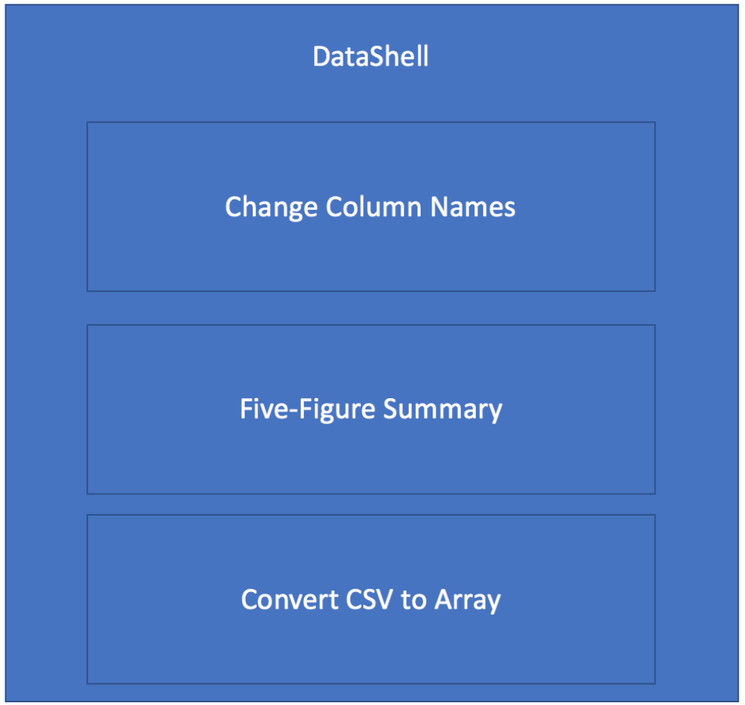
- Made up of methods and attributes
- Initialized with an **init** constructor method
- Has a self attribute that's referring to the class (or particular instance of that class)

The DataShell

DataFrame

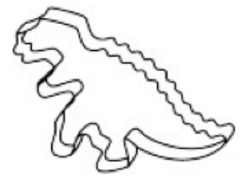
DataShell

The benefits of OOP



Inheritance and Composition

```
class Dinosaur(object)
```



Pterodactyl Tom

```
Class Pterodactyl(Dinosaur)
```

Tyrannosaurus Sue

```
class Tyrannosaurus(Dinosaur)
```

How to get better at OOP

1) Read well-documented codebases

2) Write your own classes

3) Ask for feedback

4) Don't get discouraged!

OBJECT ORIENTED PROGRAMMING IN PYTHON

Final Steps