Classes

Recall: Classes define a type. Objects are a specific type

Analogy: Cat is a class of animal; My cat, Jersey, is an instance of cat.

Recall: Classes consist of data and methods

Advantages of classes

Modularity: break up application into objects (similar to TDD)

Encapsulation: Data is *encapsulated* inside classes; Use interface (e.g. methods) to access data. => The class implementation can change without users being aware of it, e.g. this is how classes support **abstraction**

Class syntax

```
class <className>:

    def __init__(self, param1, param2, …..):

        # initialize member variables

        # members are what we call the data in a class

        self.member1 = <initial value depends on type: int, str, etc>

    ….

    <other methods here: all should have self as first parameter!>
```

Class syntax

```
class <className>:

    def __init__(self, param1, param2, …..):

        # initialize member variables

        # members are what we call the data in a class

        self.member1 = <initial value depends on type: int, str, etc>

    ….

    <other methods here: all should have self as first parameter!>
```

__init__ is the constructor method. This method is called when you create an object, e.g.

```
point = Point(x,y)
```

calls the __init__ function inside class Point

Class syntax

self is a special parameter that represents the object that this method “runs on”

```
class <className>:

    def __init__(self, param1, param2, …..):

        # initialize member variables

        # members are what we call the data in a class

        self.member1 = <initial value depends on type: int, str, etc>

    ….

    <other methods here: all should have self as first parameter!>
```

Class syntax

any other parameters (possibly none) go after **self**

```
class <className>:

    def __init__(self, param1, param2, …..):

        # initialize member variables

        # members are what we call the data in a class

        self.member1 = <initial value depends on type: int, str, etc>

    ….

    <other methods here: all should have self as first parameter!>
```

Class syntax

We use **self** again to refer to the object's own data!!!

```
class <className>:

    def __init__(self, param1, param2, …..):

        # initialize member variables

        # members are what we call the data in a class

        self.member1 = <initial value depends on type: int, str, etc>

    ….

    <other methods here: all should have self as first parameter!>
```