ClassesObjects

Monday, April 20, 2020 9:33 AM

Defining Classes

```
class <ClassName>:
    def __init__(self, <param1>, <param2>, ...., <paramN>):
        self.<data1> = <param1>
        self.<data2> = <param2>
        ....
        self.<dataN> = <paramN>

    def <method1>(self, ...):
        <body1>

    def <method2>(self,...):
        <body2>
    ...
    def <methodN>(self,...):
        <bodyN>
```

self keyword needed to access internals in a class (e.g. data + methods)

constructor - initializes the obj - called first - create & init object's data in this method

member variables → the "data" in an object → these variables exist as long as the object does

additional methods

special method that lets us print an object

```
def __str__(self):
    return "String representation of this object"
```

↑ built-in to Python → we replace the default version with our own here

From <https://www.cs.swarthmore.edu/courses/CS21/S20/alinen/week06.html>

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def scale(self, factor):
        self.x = factor * self.x
        self.y = factor * self.y

    def setX(self, x):
        self.x = x

    def setY(self, y):
        self.y = y

    def getX(self):
        return self.x

    def getY(self):
        return self.y

    def __str__(self):
        return "%f,%f"%(self.getX(), self.getY())
```

Constructor for Point: point has 2 member variables: x & y. Each object will get its own x, y variables

method which scales the member variables in the object by factor

Mutator / Setter methods → modifies data in an object (best practice is to keep member variables internal to the class, A user shouldn't need to know the internals to use the class!)

Acessors / Getters: Returns data inside an object

returns a string we can use for printing!

Using Classes

Syntax:

<object> = <ClassName>(...params) # creates object

<object>.<member> # accesses method/data in object

NOTE! self is used when we define the class but not when we use it!

```
1. p = Point(5, -3)
2. p.scale(10)
3. print("The point is", p)

4. p.setX(-4)
5. print("The point is", p)
```

1. calls __init__(self, x, y) → 5, -3
2. calls scale(self, factor) → 10
3. calls __str__(self)