
Python DeCal
Week 8

Announcements

- HW6 just due
- Final Projects
  - Communicating with your group members
  - Final Project Proposal due: **Next Wednesday**
    - **Submit a PDF**
    - **You don’t have to stick with it but it’s recommended that you do**
- Attendance: https://tinyurl.com/my-mosquito-friend

Recap and Discussion

- In `pandas`, what would you do if you want to convert a column in a dataframe into a numpy array?
- How useful have you found the documentations to be?

What is the tool we use for Scientific Writing?

- LaTeX
  - Typesetting system
  - Code-like, but definitely easier than Python
    - You don’t see the output while you are writing it

- Overleaf
  - Online LaTeX editor
  - Easy for collaborative work
  - www.overleaf.com

Useful Packages and links

- Packages
  - amsmath
  - amssymb
  - physics
  - enumitem
  - GOOGLE AS YOU NEED MORE!

Links

- https://www.overleaf.com/learn
  - Overleaf documentation
- http://detexify.kirelabs.org/classify.html
  - Searching for command by drawing the symbols
- www.google.com
  - We all know what this is for
- https://texnique.xyz/
  - If you are bored and want practice in LaTeX

FRIDAY

https://forms.gle/DeSMGtQKQqwRy3GY6

Announcements

- Start really getting ready to write your project proposal this weekend

- No Python HW this week, You’re done learning the basics!

- Project Proposals are due on Wednesday, this is your HW

Breakout Rooms

- Breakout rooms with your project partners if they are here. We want to discuss with you what you are settling on for your project?

Object Oriented Programming

- Arguably one of the most **powerful** features of python
- Allows us to construct **objects** with their own **traits** and **methods**
- We can create a **Class** of objects, once you make an object that object will have all the traits and methods of its' class.

Example

Class Human

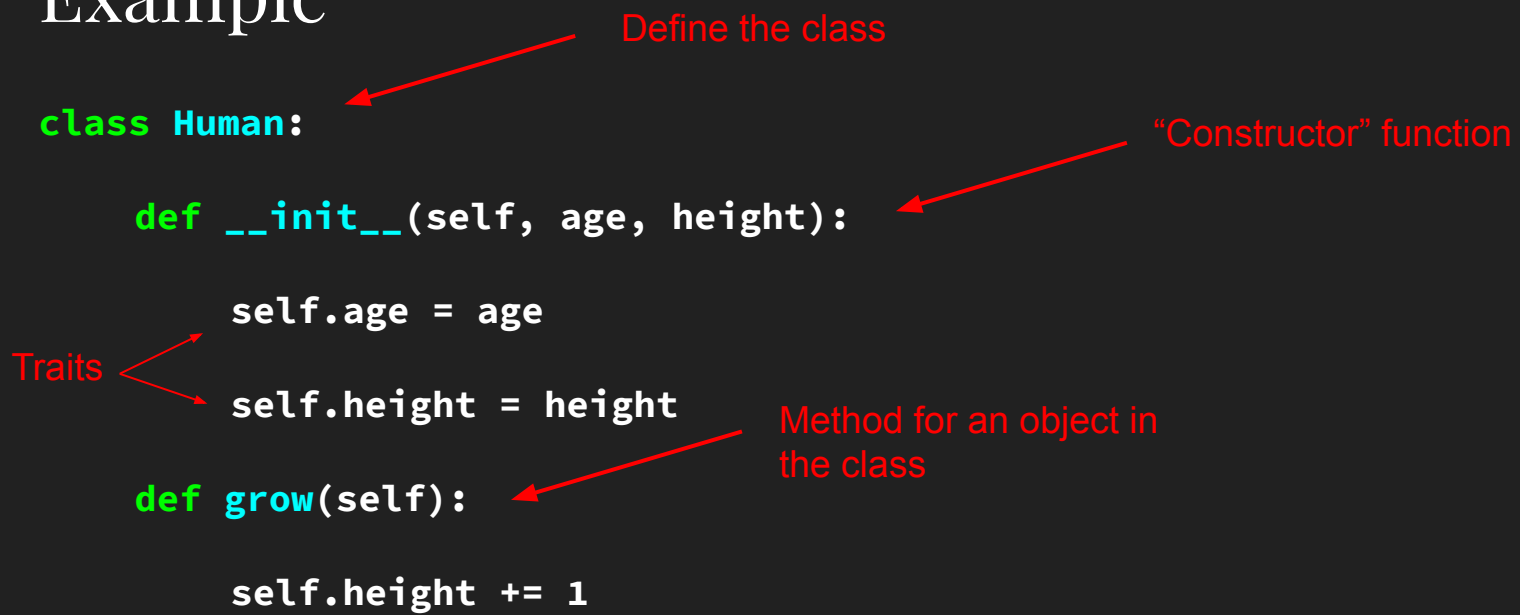
Traits:  Properties of the object

- eye color, hair color, skin color, height, weight, etc....

Methods:  Things the object can do

- Jump, run, clap, skip, push, talk, think, etc....

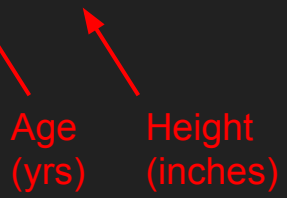
Example




```
class Human:

    def __init__(self, age, height):

        self.age = age

        self.height = height

    def grow(self):

        self.height += 1
```

Example

```
>>> James = Human(20, 70)
```



The line above creates an object from the human class with the required arguments of the constructor function. I call this object: James

New notation for calling a function!
Does it look familiar?

```
>>>James.grow()
```

The line above calls the grow method belonging to an object in the human class, which just adds an inch to my height so now

```
>>> print(James.height)

71
```

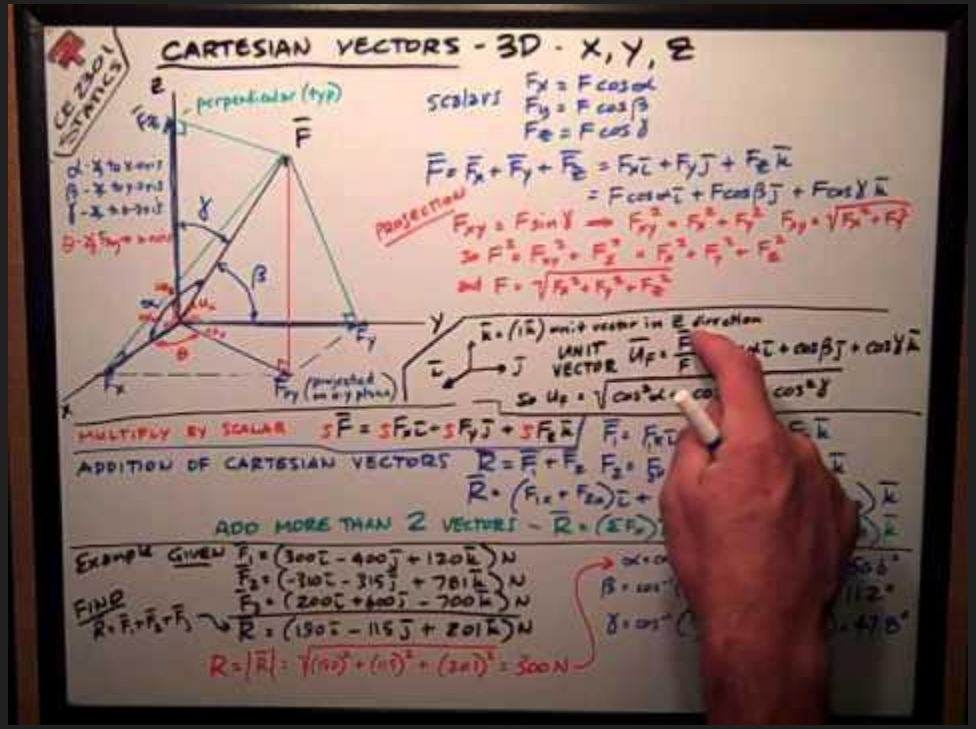
You've already used object oriented programming!

Anyone remember this?

```
ax.plot(x,y)

ax.set_title('Title')

ax.set_xlabel('x-axis')

ax.set_ylabel('y-axis')

ax.legend()
```

These are all just **methods** for the ax object!

Can be useful for physics!



Can be useful for physics!

Vector Class

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

Constructor Function + Traits

Vector Class

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    #all the properties of vectors and the tools we can use with them. We could use numpy, but this made more intuitive sense to make our own
    #class, and rewrite our own operators. It is nice to stay consistent with the class and object style from before with the balls.
    def len(self):
        return math.sqrt(self.x*self.x + self.y*self.y)
    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y)
    def __sub__(self, other):
        return Vector(self.x - other.x, self.y - other.y)
    def __mul__(self, other):
        return Vector(self.x * other, self.y * other)
    def __rmul__(self, other):
        return Vector(self.x * other, self.y * other)
    def __truediv__(self, other):
        return Vector(self.x / other, self.y / other)
```

Constructor Function + Traits

Methods

Vector Class

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    #all the properties of vectors and the tools we can use with them. We could use numpy, but this made more intuitive sense to make our own
    #class, and rewrite our own operators. It is nice to stay consistent with the class and object style from before with the balls.
    def len(self):
        return math.sqrt(self.x*self.x + self.y*self.y)
    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y)
    def __sub__(self, other):
        return Vector(self.x - other.x, self.y - other.y)
    def __mul__(self, other):
        return Vector(self.x * other, self.y * other)
    def __rmul__(self, other):
        return Vector(self.x * other, self.y * other)
    def __truediv__(self, other):
        return Vector(self.x / other, self.y / other)
    def angle(self):
        return math.atan2(self.y, self.x)
    def norm(self):
        if self.x == 0 and self.y == 0:
            return Vector(0, 0)
        return self / self.len()
    def dot(self, other):
        return self.x*other.x + self.y*other.y
```

Constructor Function + Traits

Methods

Demo