CMPSC 100 SPRING 2021

Objects

COURSE INFORMATION

- Grading update
  - You know what it is
- Quiz will be posted later this morning
- Update: the lab this week is optional (also known as “Extra Credit”)
  - The material we’re covering today isn’t *quite* as “required”
  - It is still significant

A class == the code creating
a blueprint for functionality

An object == class code *initialized*

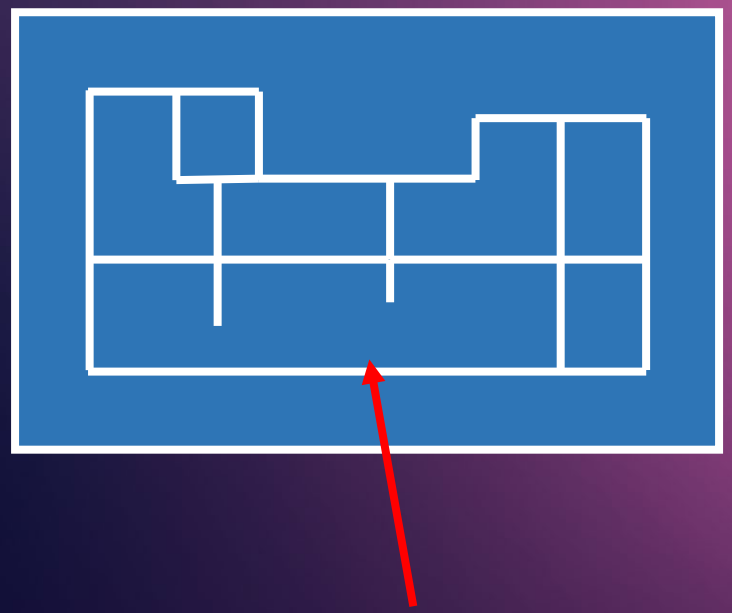
```
e.g. ulysses = Cat(
                name = “Ulysses”,
                fur_color = “brown”,
                is_tabby = True
            )
```

Will G. Wiz ever get a hat?

¯_(ツ)_/¯

LOLZ


CANNOT
PREDICT
NOW



Like a “blueprint” for functionality

For example, what does a Magic 8 Ball actually do?

- Shakes
- Answers cryptically
  - Can be positive, negative, or neutral
- Usually gets it wrong

Class declaration

```
# Because I’m probably violating some law calling it a “Magic 8 Ball”
class MagicBall:

    def __init__(self):
        self.predictions = self.load("data/responses.json")
```

Class “constructor”

An implicit parameter

```
def __init__(self):
    self.predictions = self.load("data/responses.json")
```

“self” -> “this copy”

```
def __init__(self):
        .
        .
        .
```

Constructor method

Called *immediately* when an object is **initialized**

Requests/requires the *minimum* amount of data required to create the object. In this case, all we need is a **numerator** (numer) and a **denominator** denom.

The living, rampaging, OBJECTZILLA

Identifier

Class name

```
magic_ball = MagicBall.MagicBall()
```

Arguments matching
Constructor

FUNCTIONS, METHODS, AND PROPERTIES

```
def shake(self):
        result = random.choice(self.predictions)
        self.message = result["message"]
        self.message_type = result["type"]
```

As with a normal function, we'd still need to call magic_ball.shake() <- no explicit parameters

Message and message_type are properties

We can call these from a file which imports MagicBall:

```
magic_ball.message
magic_ball.message_type
```