Objects

Like everyday objects!

descriptions, actions

Object

- constructors — def __init__(self, blah):
- variables — class vs. parameters
- methods

Instance attributes — vs. — Class attributes

puppy.name — Puppy.name

Waterbottle → cup

Repr functions

return "Boo!"

Inheritance

```
class Waterbottle:
    level=50
    material="plastic"
    def __init__(self, level, material):
        self.level = level
        self.material = material
    def open(self):
        self.level -= 10

    def being_green(self):
        self.material = "metal"
```

Waterbottle OG class

level: 40
material: "metal"
init()
open()
being_green()

wb (instance)

level: 60
material: aluminum
init
open
being green

wb = Waterbottle(60, "Aluminum")

wb.level = 50 → 40

wb.material = "Aluminum" → "metal"

wb.open() ←

wb.being_green()

Waterbottle.level = 50

Waterbottle.material = plastic

Waterbottle.open()

Waterbottle.level = 40

Waterbottle.being_green()

Waterbottle.material

```
class Email:
    """Every email object has 3 instance attributes: the
    message, the sender name, and the recipient name.
    """
    def __init__(self, msg, sender_name, recipient_name):
```

self.

self.msg =

se

```
class Server:
    """Each Server has an instance attribute clients, which
    is a dictionary that associates client names with
    client objects.
    """
    def __init__(self):
        self.clients = {}

    def send(self, email):
        """Take an email and put it in the inbox of the client
        it is addressed to.
        """
```

key

value

self.clients = { }

email object

key: name

value: object

email → recipient_name → client → recieve this email

rn = email.recipient_name

client_ob = self.clients[rn]

client_ob.receive(email).

```
    def register_client(self, client, client_name):
        """Takes a client object and client_name and adds them
        to the clients instance attribute.
        """
```

self.clients[client_name] = client

```
class Client:
    """Every Client has instance attributes name (which is
    used for addressing emails to the client), server
    (which is used to send emails out to other clients), and
    inbox (a list of all emails the client has received).
    """
    def __init__(self, server, name):
        self.inbox = []
```

self.

self.

self.sever.register_client(self , self.name)

```
    def compose(self, msg, recipient_name):
        """Send an email with the given message msg to the
        given recipient client.
        """
```

(message, sender, recipient)

email = Email(msg, self.name, rcv

```
    def receive(self, email):
        """Take an email and add it to the inbox of this
        client.
        """
```

self.inbox.append(email)