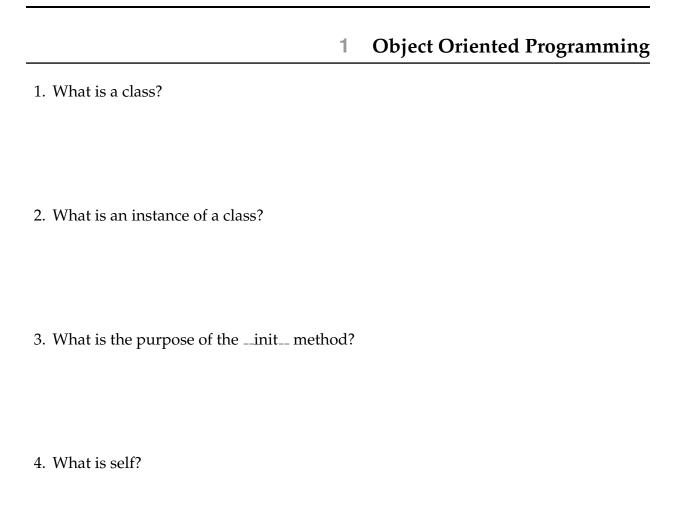
A Whale-ful amount of OOP AND INHERITANCE

COMPUTER SCIENCE MENTORS CS 88

March 29th to April 2nd

Founded January 10, 1967



5. What would Python display? Write the result of executing the following code and prompts. If nothing would happen, write "Nothing". If an error occurs, write "Error".

```
class UPECheck:
    status = "candidate"
    tasks done = 0
    def ___init___(self, name):
        self.name = name
    def task(self, other):
        other.tasks_done += 5
    def __repr__(self):
        return "UPE Check for " + self.name
>>> vanshaj = UPECheck("Vanshaj")
>>> vanshaj.status, vanshaj.tasks done
>>> vanshaj.status = "officer"
>>> vanshaj.status
>>> UPECheck.status
>>> rahul = UPECheck("Rahul")
>>> rahul.officer_chat = vanshaj
>>> rahul.officer_chat
>>> vanshaj.tasks_done += rahul.tasks_done
>>> vanshaj.tasks done, rahul.tasks done
>>> vanshaj.task(rahul)
>>> vanshaj.tasks_done, rahul.tasks_done
```

6. We now want to write three different classes, Postman, Client, and Email to simulate UPE email services. Fill in the definitions below to finish the implementation!
>>> postman = Postman() #Create a new Postman
>>> john = Client(postman, "John") #Create client named John>>>
rohan = Client(postman, "Rohan") #Create client named
Rohan
>>> john.compose("POG", "Rohan") #John sends an email to Rohan
>>> rohan.compose("CHAMP", "John") #Rohan sends an email to John
>>> rohan.inbox[0].msg #Rohan's inbox
"POG"
>>> john.inbox[0].msg #John's inbox
"CHAMP"

```
class Email:
    """Every email object has 3 instance attributes: the
      message,
    the sender (their name), and the addressee (the
       destination's
    name).
    def __init__(self, msg, sender, addressee):
class Postman:
    """Each Postman has an instance attribute clients, which
       is a
    dictionary that associates client names with client
       objects.
    11 11 11
    def ___init___(self):
        self.clients = {}
    def send(self, email):
        """Take an email and put it in the inbox of the client
        is addressed to."""
    def register_client(self, client, client_name):
        """Takes a client object and client_name and adds it
           to the
        clients instance attribute.
```

CSM 88: OOP AND INHERITANCE class Client: """Every Client has instance attributes name (which is used for addressing emails to the client), mailman (which is used to send emails out to other clients), and inbox (a list of all emails the client has received). def __init__(self, mailman, name): self.inbox = []def compose(self, msg, recipient): """Send an email with the given message msg to the given recipient.""" def receive(self, email):

"""Take an email and add it to the inbox of this

client.

11 11 11

7. Fill in the classes Emotion, Joy, and Sadness below so that you get the following output from the Python interpreter.

```
>>> Emotion.num
()
>>> joy = Joy()
>>> sadness = Sadness()
>>> emotion = Emotion()
>>> Emotion.num # number of Emotion instances created
3
>>> joy.power
>>> joy.catchphrase() # Print Joy's catchphrase
Think positive thoughts
>>> sadness.catchphrase() #Print Sad's catchphrase
I'm positive you will get lost
>>> sadness.power
5
>>> emotion.catchphrase()
I'm just an emotion.
>>> joy.feeling(sadness) # print "Together" if same power
Together
>>> sadness.feeling(joy)
Together
>>> joy.power = 7
>>> joy.feeling(sadness) # Print the catchphrase of the more
  powerful feeling before the less powerful feeling
Think positive thoughts
I'm positive you will get lost
>>> sadness.feeling(joy)
Think positive thoughts
I'm positive you will get lost
```

class Emotion

```
def __init__(self):

def feeling(self, other):

def catchphrase(self):
```

def catchphrase(self):

class Sadness

def catchphrase(self):