Discussion 08

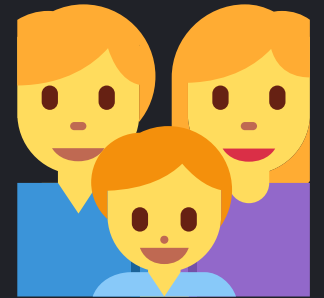**OOP, Inheritance, String Representation**

Aditya Balasubramanian
`aditbala [at] berkeley [dot] edu`

Slides available at `teaching.aditbala.com`

Announcements 📣

- Homework 6 is due Thursday 3/16.
- Ants project is due Friday 3/24.
  - Checkpoint 1 due Friday 3/17.
  - Checkpoint 2 due Tuesday 3/21.
  - Early submission bonus point 3/23.

Inheritance 👪

```python
class Dog():
    def __init__(self, name, owner):
        self.name = name
        self.owner = owner
    def eat(self, thing):
        print(self.name + " ate a " + str(thing) + "!")
    def talk(self):
        print(self.name + " says woof!")

class Cat():
    def __init__(self, name, owner lives=9):
        self.name = name
        self.owner = owner
        self.lives = lives
    def eat(self, thing):
        print(self.name + " ate a " + str(thing) + "!")
    def talk(self):
        print(self.name + " says meow!")
```

What's the problem?

- Too much repeated code
- How to avoid this problem?
- Inheritance!

```python
class Dog(Pet): # Dog inherits the Pet class - as in, all Dogs are Pets
    ...
```

Now with Inheritance!

```python
class Pet():
    def __init__(self, name, owner):
        self.name = name
        self.owner = owner
    def eat(self, thing):
        print(self.name + " ate a " + str(thing) + "!")
    def talk(self):
        print(self.name)

class Dog(Pet): # Inherits all methods/variables from the Pet class
    def talk(self):
        print(self.name + ' says woof!')
```

Inheritance - super()

- super() will refer to methods in the parent class

```python
class Cat(Pet): # Inherits all methods/variables from the Pet class
    def __init__(self, name, owner, lives = 9):
        super().__init__(name, owner)
        # same as calling Pet.__init__(self, name, owner) from here
        self.lives = 9
    def talk(self):
        print(self.name + ' says meow!')
```

Worksheet

Representation

- `__str__`
  ○ return's a human readable form of object
- `__repr__`
  ○ return's a human readable form of object

Representation Demo

```python
class Rational:

    def __init__(self, numerator, denominator):
        self.numerator = numerator
        self.denominator = denominator

    def __str__(self):
        return f'{self.numerator}/{self.denominator}'

    def __repr__(self):
        return f'Rational({self.numerator},{self.denominator})'

>>> a = Rational(1, 2)
>>> str(a)
'1/2'
>>> repr(a)
'Rational(1,2)'
>>> print(a)
1/2
>>> a
Rational(1,2)
```

Thank you!

Anon Feedback -> https://tinyurl.com/adit-anon