Discussion 8

Inheritance and String Representation

Antonio Kam

anto [at] berkeley [dot] edu

All slides can be found on

`teaching.rouxl.es`

Announcements

- HW 06 Released!
  - OOP/Inheritance
- Ants Phase 1 due on Friday
  - Please finish it, the checkpoint is there to keep you on track and is super important

Notes from last section

- More of these *diagrams*; Honestly, no feedback other than that this discussion was great (😳). The *diagrams* were nice.
- mini-lecture was helpful but a little fast today (but mostly bc OOP is confusing)
  - makes sense, oop is confusing, but we'll get more familiar with it as time goes on
- If you had unlimited money to start a business, what would it be? (ie what type of business would you start)
  - i have no clue at all tbh, never thought of this
  - i literally sat here thinking for like 5 minutes and still couldn't come up with something
- What are your favorite places to travel?
  - Hong Kong: food; tbh a lot of asian countries (food)

Notes from last section

- a little bit more on OOP
  - 🤯 this discussion is also oop related!
- Mini-lecture was helpful + inheritance more clear (lab)
  - nice, we'll see a bit more
- would you rather have unlimited bacon but no games. Or games, unlimited games, but no games.
  - ...what?
  - I assume the other clause says unlimited games but no bacon
  - and i think that choice is easy - unlimited games but no bacon
- What is your fastest record for the rubik's cube? Could we see a demo 0.o
  - One Handed: 6.85 Single, 9.28 Average; Blindfolded: 17.47 Single

Mid-Semester Feedback

- Overall I think y'all like me (hopefully) :partying:
- Sometimes there's very contrasting feedback
  - Some people want discussions to go faster
  - Some people want discussions to go slower
    - 🤦‍♀️
  - I unfortunately cannot accommodate for all of this
  - Discussion review sections!
- I speak too fast
  - many people said this, and it's true 😭 i speak very fast, and it's a reoccuring issue for me.
  - "Maybe try to talk a little bit slower, but I also understand that you kind of have to talk fast to get through everything" - this is not true, i can talk slow and do this

Mid-Semester Feedback

- Move through the material slower, simplify the material
- Sometimes we spend too much time on the first topic when there are multiple topics to be covered.
  - this is still a pretty bad problem for me, but i think i've been doing a better job of selecting the more important topics
  - still room to improve though
- Many of you said I have a lot of energy 😳
  - This is something that I don't have normally (outside of teaching), but it's weird that this has consistently showed up for me teaching 🤔
- "Very fun. A lot of energy to the room even when the classroom doesn't give much back."
  - hi can u give energy back pls 👉 🥹 👈

Mid-Semester Feedback

- pacing, maybe more dispersed lecturing on content
  - i think in general this won't help too much with time management, just because the transitions tend to be what takes up more time, but i am still open to trying this
- Mayhaps more coding examples
  - true
- giving more guidance in lab / going over
  - we're all here during lab - it's a time for you to ask questions and even collaborate with the people near you!
- switch on the lights at the front

Temperature Check 🌡️

- Inheritance
- Representation (`repr`, `str`)

Inheritance

```
class Dog():
    def __init__(self, name, owner):
        self.is_alive = True
        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.is_alive = True
        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!")
```

Inheritance

Notice the redundancies in the code? One of the core foundations in this class is to not repeat yourself (DRY)

- Instead, you can use inheritance to solve this problem
- Syntax when creating a class is to put brackets around the class you want to inherit:

```
class Cat(Pet): # Cat inherits the Pet class - as in, all cats are pets (in this world)
    ...
```

Inheritance

```
class Pet():
    def __init__(self, name, owner):
        self.is_alive = True    # It's alive!!!
        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 Animal class
    def talk(self):
        print(self.name + ' says woof!')
```

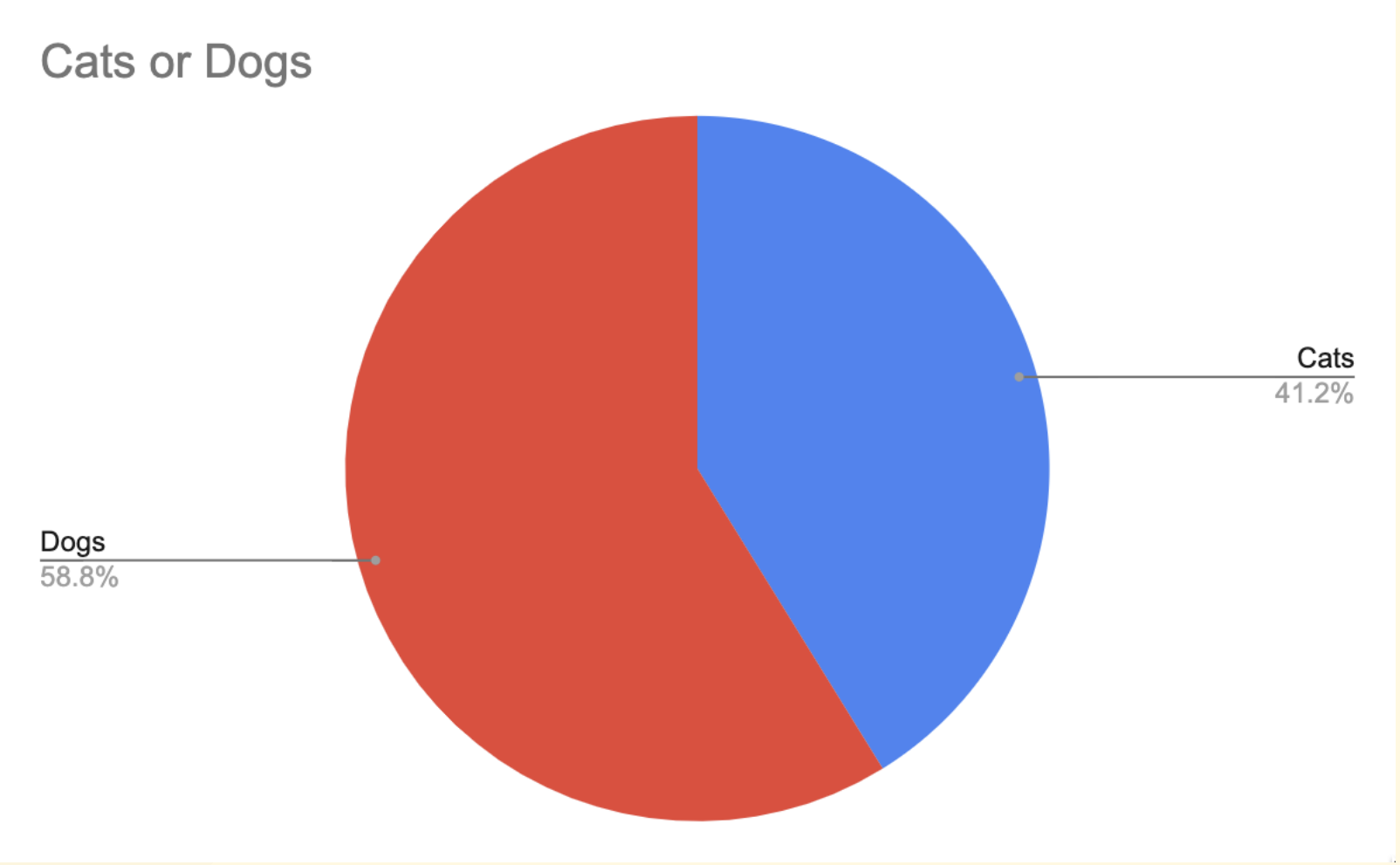
Inheritance - `super()`

- Calling `super()` will refer to the class's *superclass*
- You can use the parent's method and then add on to that.

```
class Cat(Pet): # Inherits all methods/variables from the Animal 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!

Results from last section (`links.rouxl.es/disc`)



Worksheet!

Representation

What is Representation?

- Python objects by default have really ugly names if you try to output them into the interpreter:
  - `<__main__.Cat object at 0x7fe611abff70>`
- Representation lets you define a better way to display this out in the console such that other people know what you're talking about

`repr` and `str`

- `repr`
  - Affects what is displayed when object is evaluated in terminal
  - 'Computer readable' (as in, you can use `eval` on something from a `__repr__` method, and it should not error if implemented correctly)
- `str`
  - Changes what is displayed when an object is printed
  - 'Human readable'
- If you directly output to a console, `__repr__` is used, but if you try to print, it will first try and find `__str__`, and if that doesn't exist, it will use `__repr__` instead.

`repr` and `str`

- In addition, the `print()` function calls the `__str__` method of the object and displays the returned string **with the quotations removed**, while simply calling the object in interactive mode in the interpreter calls the `__repr__` method and displays the returned string **with the quotations removed.**
- show worksheet for examples anto

Example

```
class Pet:
    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return "Good " + self.name

    def __str__(self):
        return self.name + " says hello!"
```

- `rex = Pet("Rex")`
- `rex` -> Good Rex (notice how this doesn't have quotes? Python behaviour!)
- `print(rex)` -> Rex says hello!

Worksheet!

Mental Health Resources

- CAPS:
  - If you need to talk to a professional, please call CAPS at 510-642-9494.
- After Hours Assistance
  - For any assistance after hours, details on what to do can be found at this link

Anonymous Feedback Form

`links.rouxl.es/feedback`

Thanks for coming! 🥳

Please give me feedback on what to improve!