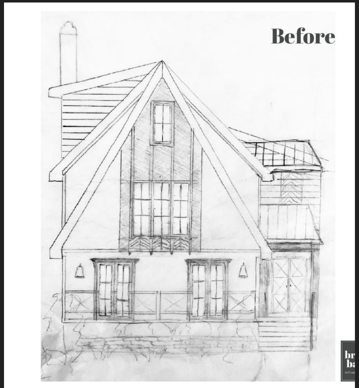
COMP 110

OOP Part 2: Classes and Methods

But first, a review of **classes** and **objects**

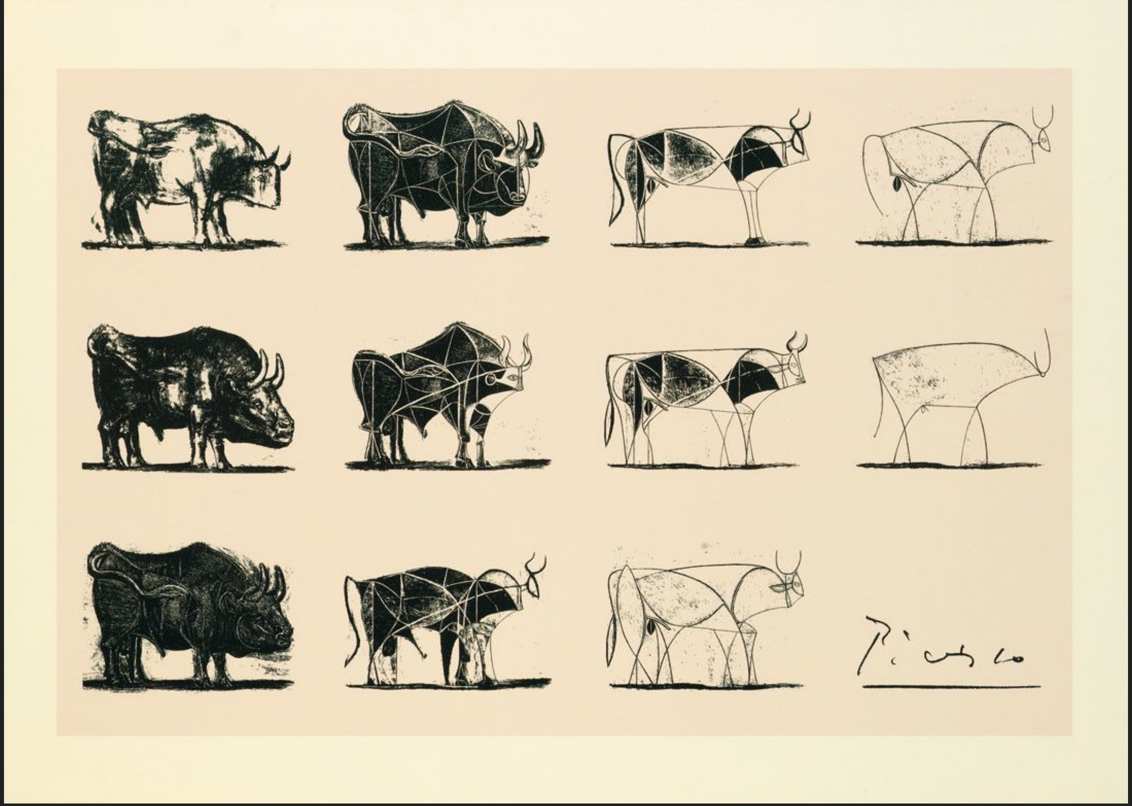
- Think of a **class** as a blueprint/ template
    - Defines attributes and behaviors its objects will have
- An **object** is an *instance* of a class
    - E.g., if the class is the blueprint, the object is the house!
    - Has all the specified attributes and behaviors
    - Different objects share these attributes and behaviors, but are distinct!



After
(Virtual Image
385

After
(Virtual Image
385

What does Picasso’s “Bull” progression show?



Pablo Picasso. Bull (1945). A Lithographic Progression.

Abstraction: whittling down to the essentials

Real-world example: Flights

What information do you need when you're preparing for (or actively on) a flight?

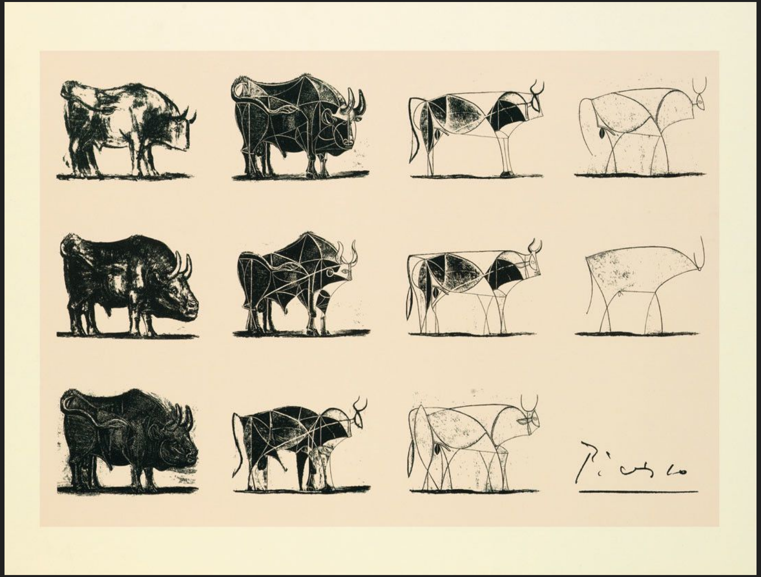
❏ ALL of the flight details?

  ❏ E.g., how the pilot flies the plane

or,

❏ Only the ones that are essential for you to know?

  ❏ Departure and arrival times/cities, your seat assignment, plans after landing



Pablo Picasso. Bull (1945).
A Lithographic Progression.

Abstraction: whittling down to the essentials

Monday's example: Instagram Profiles

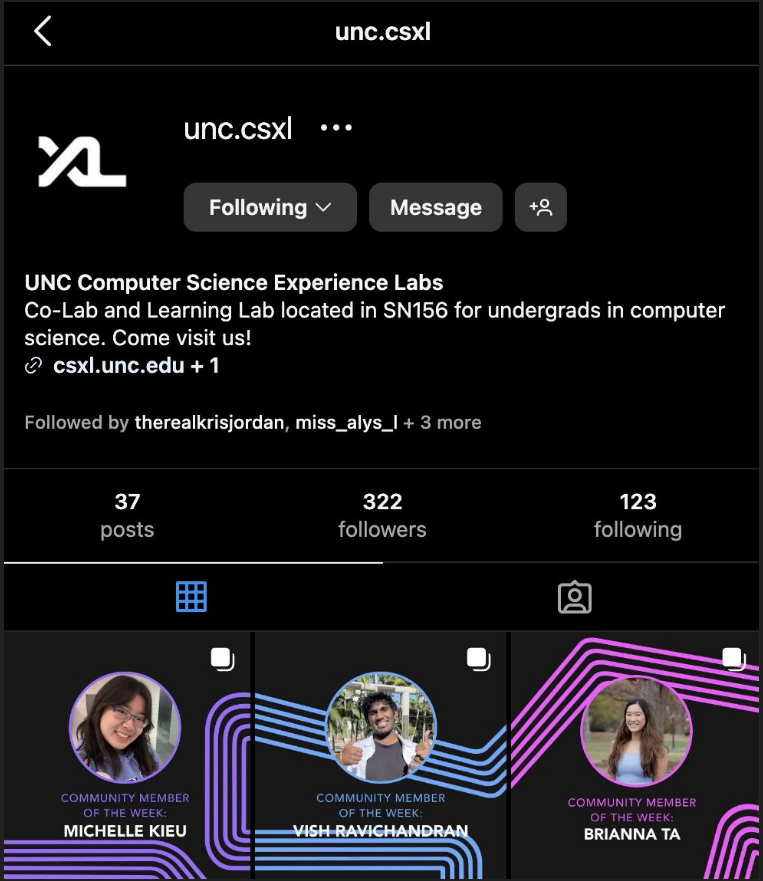
When you:

❑ Follow someone

❑ Add to your story

❑ Post a new photo

Do you think about what's happening behind the scenes (in Meta's code)?



unc.csxl
unc.csxl •••
Following
Message

UNC Computer Science Experience Labs
Co-Lab and Learning Lab located in SN156 for undergrads in computer science. Come visit us!
csxl.unc.edu + 1

Followed by therealkrisjordan, miss_alys_l + 3 more

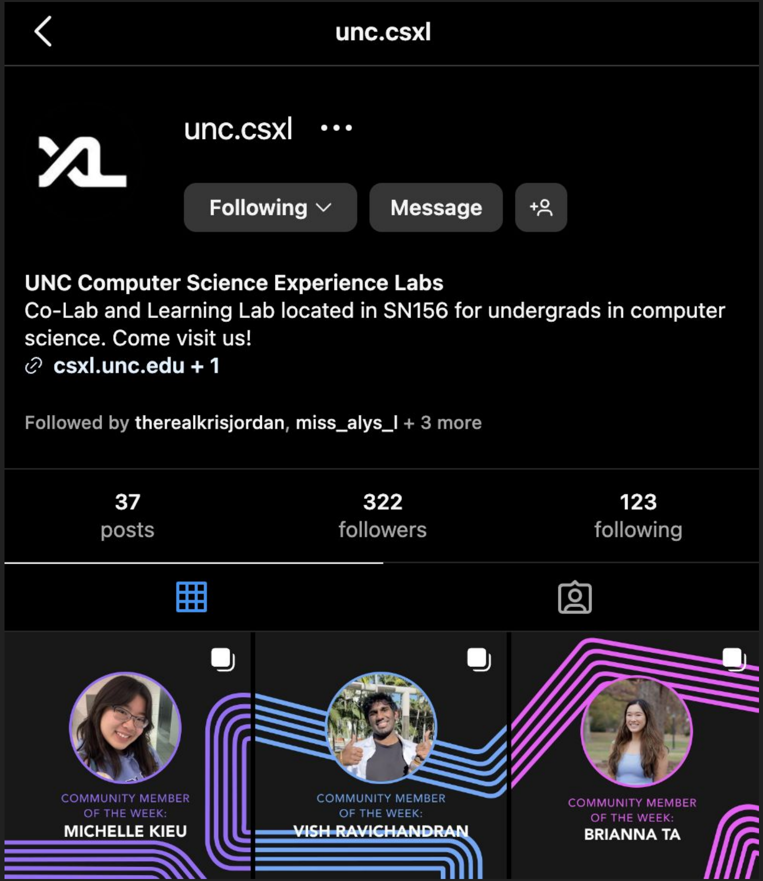
37	322	123
posts	followers	following

COMMUNITY MEMBER OF THE WEEK: MICHELLE KIEU
COMMUNITY MEMBER OF THE WEEK: VISH RAVICHANDRAN
COMMUNITY MEMBER OF THE WEEK: BRIANNA TA

Objects are a **data abstraction**

All objects have:

1. An **internal representation**
   a. Data attributes
2. An **interface** for interacting with the object
   a. Interface defines behaviors but *hides implementation* (the details!)
   b. **Methods**: Functions defined within a class
      i. `self` is the first parameter



unc.csxl

unc.csxl •••

UNC Computer Science Experience Labs
Co-Lab and Learning Lab located in SN156 for undergrads in computer science. Come visit us!
csxl.unc.edu + 1

Followed by therealkrisjordan, miss_alys_l + 3 more

37 posts

322 followers

123 following

COMMUNITY MEMBER OF THE WEEK: MICHELLE KIEU
COMMUNITY MEMBER OF THE WEEK: VISH RAVICHANDRAN
COMMUNITY MEMBER OF THE WEEK: BRIANNA TA

Methods: defined in the *class*, used on *objects*

```python
class Profile:
    username: str
    followers: list[str]
    following: list[str]

    def __init__(self, usr):
        self.username = usr
        self.followers = []
        self.following = []

    # Method definitions
    def follow(self, username: str) -> None:
        self.following.append(username)

    def get_following(self) -> list[str]:
        return self.following

my_prof: Profile = Profile("comp110fan")# Calls __init__()
print(my_prof.following)
my_prof.follow("unc.latinosintech")
print(my_prof.following)
```

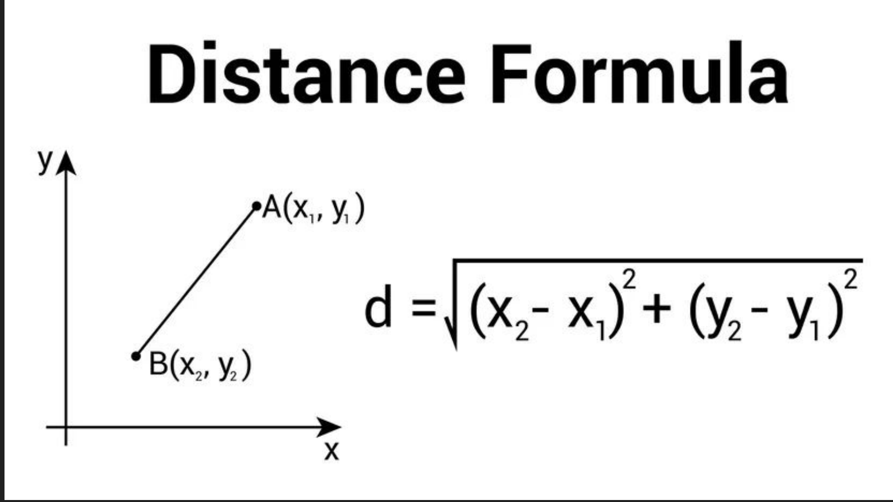
Method definitions
(first parameter is `self`)!

Method call
<object>.<method>(<non-`self` parameters>)

Memory diagram

```python
class Profile:
    username: str
    followers: list[str]
    following: list[str]

    def __init__(self, usr):
        self.username = usr
        self.followers = []
        self.following = []

    # Method definitions
    def follow(self, username: str) -> None:
        self.following.append(username)

    def get_following(self) -> list[str]:
        return self.following

my_prof: Profile = Profile("comp110fan")
print(my_prof.following)
my_prof.follow("unc.latinosintech")
print(my_prof.following)
```

Class and method writing



- Write a class called `Coordinate`
- It should have two attributes:
  - `x: float` and `y: float`
- Write a `constructor` that takes three parameters:
  - `self`, `x (float)` and `y (float)`
- Write a method called `get_dist` that takes as parameters `self` and `other` (another `Coordinate` object). The method should return the distance between the two `Coordinate` objects (use the equation above!).