COMP 110

Object Oriented Programming (OOP)

What are objects *in the real world*?

Things that can be perceived, used, or interacted with

They can be *physical*:

- Chair is a type of furniture
- Human is a type of mammal
- Fork is a type of utensil

or *abstract*:

- Happiness is a type of emotion
- Friendship is a type of relationship
- Learning is a type of experience

And they all serve distinct purposes!

What are objects *in Python*?

Many types of data in Python:

`23` `"hello world!"` `3.14159` `[24, 26, 25, 27]`

`{110.001: "Lytle and Jordan", 110.003: "Hinks"}` `True`

Every object has:

- A type
- An internal data representation
- A set of procedures for interaction with the object

An object is an instance of a type

- `23` is an instance of an `int`
- `"hello world!"` is an instance of a `str`

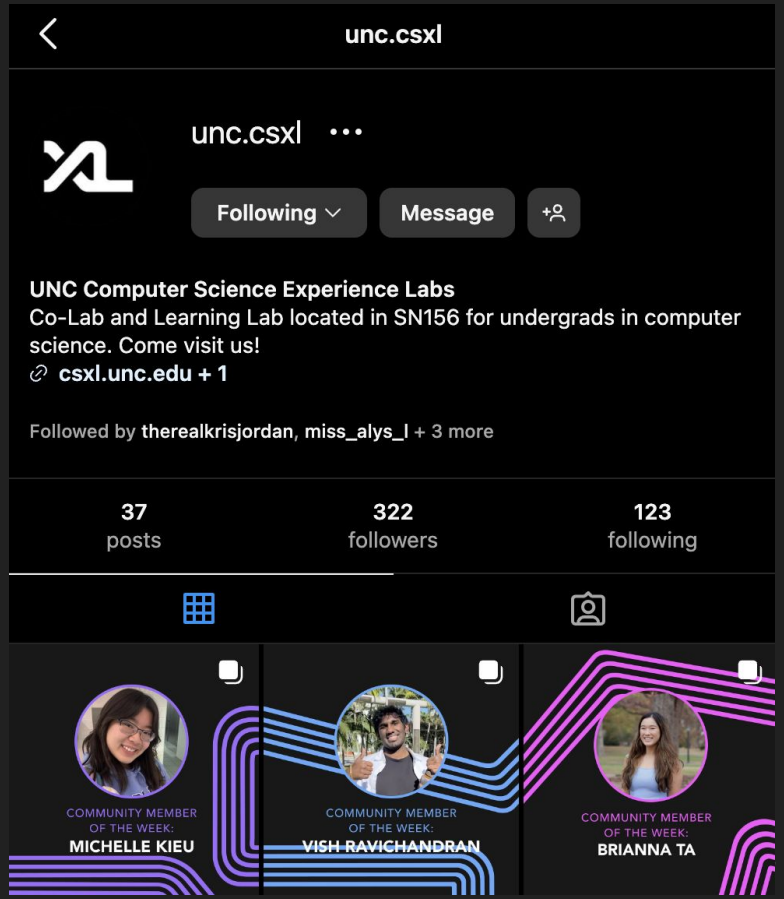
Modeling an Instagram profile with code

What data should we keep track of?

```
username: str = "unc.csxl"
bio: str = "UNC CS Experience Labs"
posts: int = 37
followers: int = 322
following: int = 123
private: bool = False
```

What functions would be useful?

- View # followers or following
- Write or update a bio
- (Un)follow an account
- Make an account private/public



Modeling an Instagram profile with code

What data should we keep track of?

```
username: str = "unc.csxl"
bio: str = "UNC CS Experience Labs"
posts: int = 37
followers: int = 322
following: int = 123
private: bool = False
```

What functions would be useful?

- View # followers or following
- Write or update a bio
- (Un)follow an account
- Make an account private/public

Instagram has over **2 billion** user profiles…

What challenges could we encounter?

It'd be nice to be able to bundle these attributes and functions into one object per profile…

Modeling an Instagram profile with a `class`

declaring a new data type!

```
class Profile:
```

Modeling an Instagram profile with a `class`

declaring a new data type!

```
class Profile:
    username: str
    bio: str
    followers: int
    following: int
    private: bool
```

declaring attributes
(every Instagram profile has these!)

Modeling an Instagram profile with a `class`

declaring a new data type!

```
class Profile:
    username: str
    bio: str
    followers: int
    following: int
    private: bool
```

declaring attributes
(every Instagram profile has these!)

```
    def __init__(self):
        self.username = "usr9"
        self.bio = ""
        self.followers = 0
        self.following = 0
        self.private = False
```

initializing attributes
(what are the default values?)

Modeling an Instagram profile with a `class`

declaring a new data type!

```
class Profile:
    username: str
    bio: str
    followers: int
    following: int
    private: bool
```

declaring attributes
(every Instagram profile has these!)

```
    def __init__(self):
        self.username = "usr9"
        self.bio = ""
        self.followers = 0
        self.following = 0
        self.private = False
```

initializing attributes
(what are the default values?)

```
my_prof: Profile = Profile()
```

Construct a new profile!

Modeling an Instagram profile with a `class`

declaring a new data type!

```
class Profile:
    username: str
    bio: str
    followers: int
    following: int
    private: bool
```

declaring attributes
(every Instagram profile has these!)

```
    def __init__(self):
        self.username = "usr9"
        self.bio = ""
        self.followers = 0
        self.following = 0
        self.private = False
```

initializing attributes
(what are the default values?)

```
my_prof: Profile = Profile()
my_prof.username = "comp110fan"
print(my_prof.private)
```

Memory diagram

```python
class Profile:
    username: str
    bio: str
    followers: int
    following: int
    private: bool

    def __init__(self):
        self.username = "usr9"
        self.bio = ""
        self.followers = 0
        self.following = 0
        self.private = False


my_prof: Profile = Profile()
my_prof.username = "comp110fan"
print(my_prof.private)
```

What if we wanted to keep track of usernames of followers/accounts we're following?

```python
class Profile:
    username: str
    bio: str
    followers: int
    following: int
    private: bool

    def __init__(self):
        self.username = "usr9"
        self.bio = ""
        self.followers = 0
        self.following = 0
        self.private = False


my_prof: Profile = Profile()
my_prof.username = "comp110fan"
print(my_prof.private)
```

How could we change our attributes to do this?

What if we wanted to keep track of usernames of followers/accounts we're following?

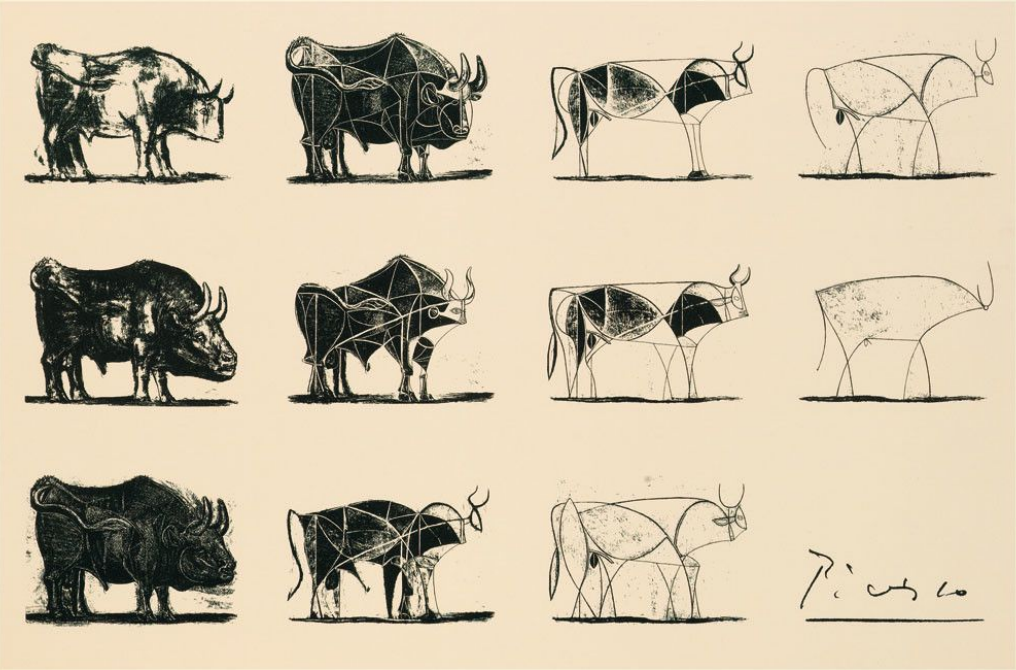
```
1  class Profile:
2      username: str
3      bio: str
4      followers: list[str]
5      following: list[str]
6      private: bool
7
8      def __init__(self):
9          self.username = "user2342300397"
10         self.bio = ""
11         self.followers = []
12         self.following = []
13         self.private = False
14
15
16 my_prof: Profile = Profile()
17 my_prof.username = "comp110fan"
18 my_prof.following.append("unc.latinosintech")
19 print(my_prof.following)
```

Use a list[str]!

Initially empty…

Until we follow an account!

Abstraction: simplifying down to the base idea

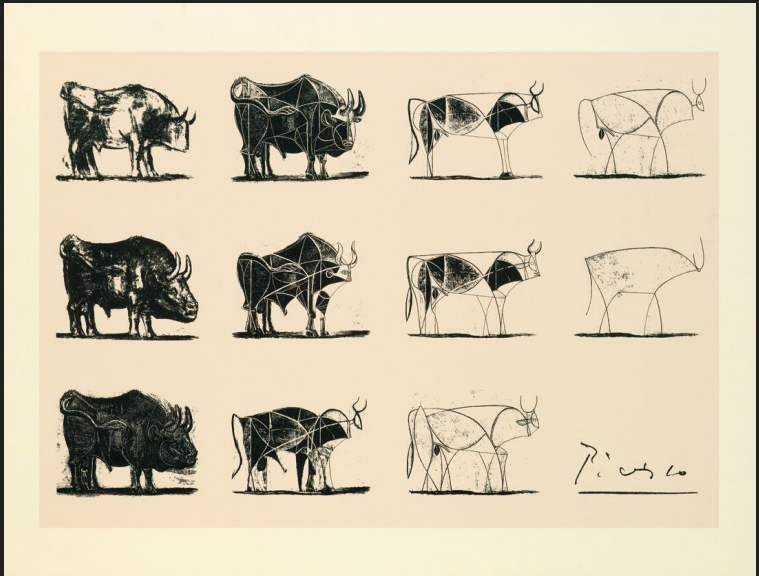


Pablo Picasso. Bull (1945). A Lithographic Progression.

Abstraction: simplifying down to the base idea

When you follow someone on Instagram, do you think about what's happening behind the scenes?

What information would we want to get or set in our Instagram profiles?



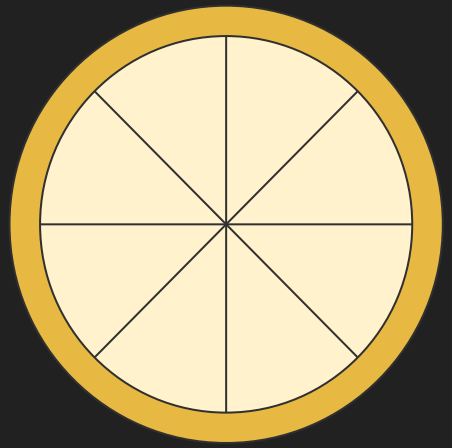
Pablo Picasso. Bull (1945).
A Lithographic Progression.

Example: Pizza

size: small

toppings: 0

gluten free: no

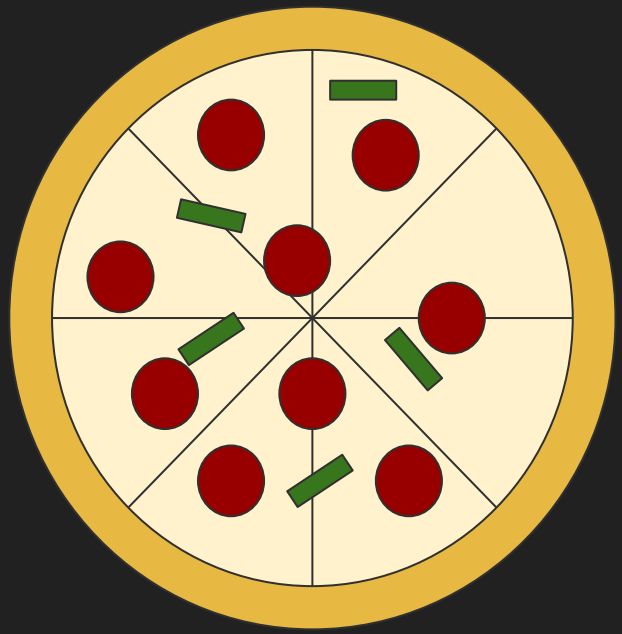


Example: Pizza

size: large

toppings: 2

gluten free: yes



Object Oriented Programming

Lets you create new objects in your program.

“Type” ~> “Class”

“Data/Variables” ~> “Attributes”

“Functions” ~> “Methods”

Creating a class

```
class <class name>:

    <class body>
```

Let's try it!
Create a file called pizza_orders.py.
Create a class called Pizza with an empty body.

Attributes

- first part of class body
- variables that belong to each instantiation of the object
- Syntax:

```
<attribute name> : <type>

gluten_free : bool
```

Let's try it!

Give the Pizza class the following attributes:

- **gluten_free:** boolean of whether or not pizza is GF
- **size:** string storing the size of the pizza
- **num_toppings:** number of toppings on the pizza

Constructor

- Method that defines what happens when new object is created
- Signature Syntax:

```
def __init__(self, <other parameters>):
```

*Essentially returns self

Let's try it!

Write a constructor that takes the following inputs and uses it to initialize the corresponding parameters

- **gf_input: bool**
- **size_input: str**
- **num_toppings_input: int**

Constructor

- Method that defines what happens when new object is created
- Signature Syntax:

```
def __init__(self, <other parameters>):
```

- Instantiation:

```
<class name>(<arguments>)
```

*Essentially returns self

Let's try it!

Create a Pizza object with the following arguments:

- **gf_input=False**
- **size_input="large"**
- **num_toppings_input:2**

Methods

- Functions that belong to an object
- Defining a method:

```
def <method_name>(self, <other params>) -> <ret type>:
```

```
def price(self) -> float:
```

- Calling a method:

```
my_pizza.price()
```

Methods

- Functions that belong to an object
- Defining a method:

```
def <method_name>(self, <other params>) -> <ret type>:

def price(self) -> float:
```

- Calling a method:

```
my_pizza.price()
```

* as opposed to `price(my_pizza)`

Let's try it!

Write a method called price with the following behavior:

- Size "small" costs $5.25, other sizes cost $7.50
- Each topping is $.25
- If gluten free, add $1

And call it!

Functions that use class objects

- You can also define a function outside the class that takes a class objects as input!

Let's try it!

Write a function called num_orders that takes as input a list[Pizza] and returns the number of of Pizza objects in the list.