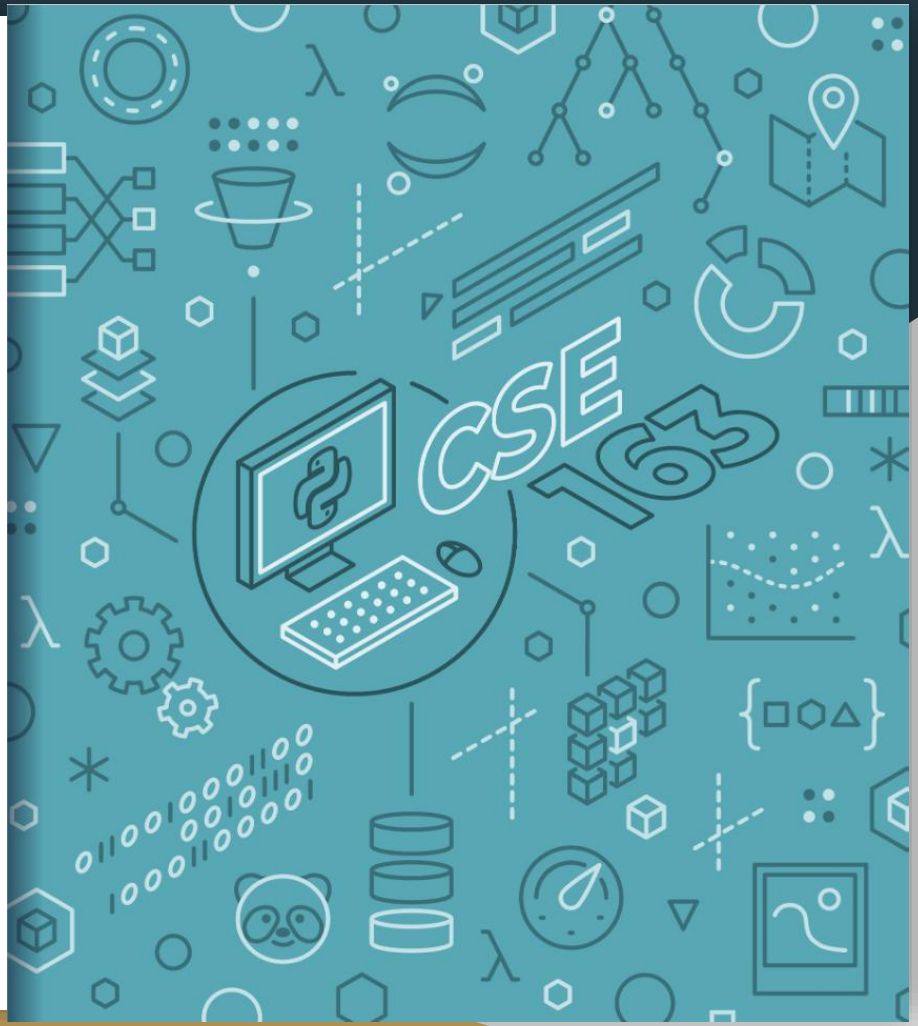
CSE 163
More Objects

Adrian Salguero
Spring 2026

💭 Icebreaker (discuss with neighbors): ‘
Which fictional world would you like to live in?
Add to our Slido!

slido.com
#cse163



Announcements

- **Take Home Assessment 2: Networks** due Thursday, April 23rd at 11:59pm!
- **Lesson 6 Canvas Quiz** due tonight at 11:59pm!
- **Reading Assignment 2** due tonight at 11:59pm!
- **THA 1 Peer Reviews** due on Gradescope Wednesday, April 15th at 11:59pm!
  - **Instructions**
  - “I Like”, “I Wish”, and “What If”
  - You are only ***assessed by the review you give***
  - If you are assigned a ***blank notebook***, please write that it was blank on your response

Class

- A ***class*** lets you define a new object by specifying what state and behaviors it has
- A class is a ***blueprint*** that we use to construct ***instances*** of the object

```
class SpiderFolk:

  def __init__(self, name: str) -> None:
    self.name: str = name

  def thwip(self) -> None:
    print(self.name + ': Thwip!')
```

A class definition

An initializer that sets fields **(state)**

A method **(behavior)**

Private in Python

- Python has no way to actually enforce this, but by convention people don’t access things that start with “--”

```
class Dog:
  def __init__(self, name: str) -> None:
    self._name: str = name

  def bark(self) -> None:
    print(self._name + ': Woof')
```

Private fields/attributes and methods

- Both fields (or attributes) and methods can be private if you don't want them access or changed by the user

```
class SpiderFolk:
  def __init__(self, name: str) -> None:
    self._name: str = name

  def _secret_identity(self) -> str:
    if 'spider' not in self._name.lower():
      return "Spider-Man!"
    else:
      return "who?"

  def get_me_pictures(self, count: int) -> None:
    print(f'{count} pictures of {self._secret_identity()}')
```

“Setter” and “Getter” Methods

- “Getter” methods return values, like fields or quick manipulations
- “Setter” methods update fields or create new values

```
class SpiderFolk:
  def __init__(self, name: str) -> None:
    self._name: str = name

  def get_name(self) -> str:
    return self._name                    # a getter method

  def rename(self, new_name: str) -> None:
    self._name = new_name + “(renamed)”    # a setter method
```

Default Parameters

- You can use default parameters like you would before
- Be careful when using objects as default values, can have negative side-effects

```
def append_to(element: int,
              to: list[int] = []) -> list[int]:
    to.append(element)
    return to

my_list = append_to(12)
print(my_list)
my_other_list = append_to(42)
print(my_other_list)
```

Python Tutor

Default Parameters Done Correctly!

- Do not use an object as a default parameter, instead use None

```
# Option 1
def append_to(element: int,
              to: list[int] | None  = None) -> list[int]:
    if to is None:
        to = []
    to.append(element)
    return to

# Option 2
def append_to(element, to=None):
    to = [] if to is None else to
    to.append(element)
    return to
```

Group Work: Best Practices

- When working with a new group for the first time:
  - Introduce yourself!
  - If possible, angle one of your screens so that everyone can see and discuss together
  - Be respectful of each other and allow everyone to speak

- Tips:
  - Start with making sure that everyone agrees to work on the same problem
  - Allow everyone to contribute or a chance to ask questions.
  - Ask if everyone agrees and periodically ask each other questions.
  - Call TAs or Adrian over if you need any help.
  - Don’t sit in silence.