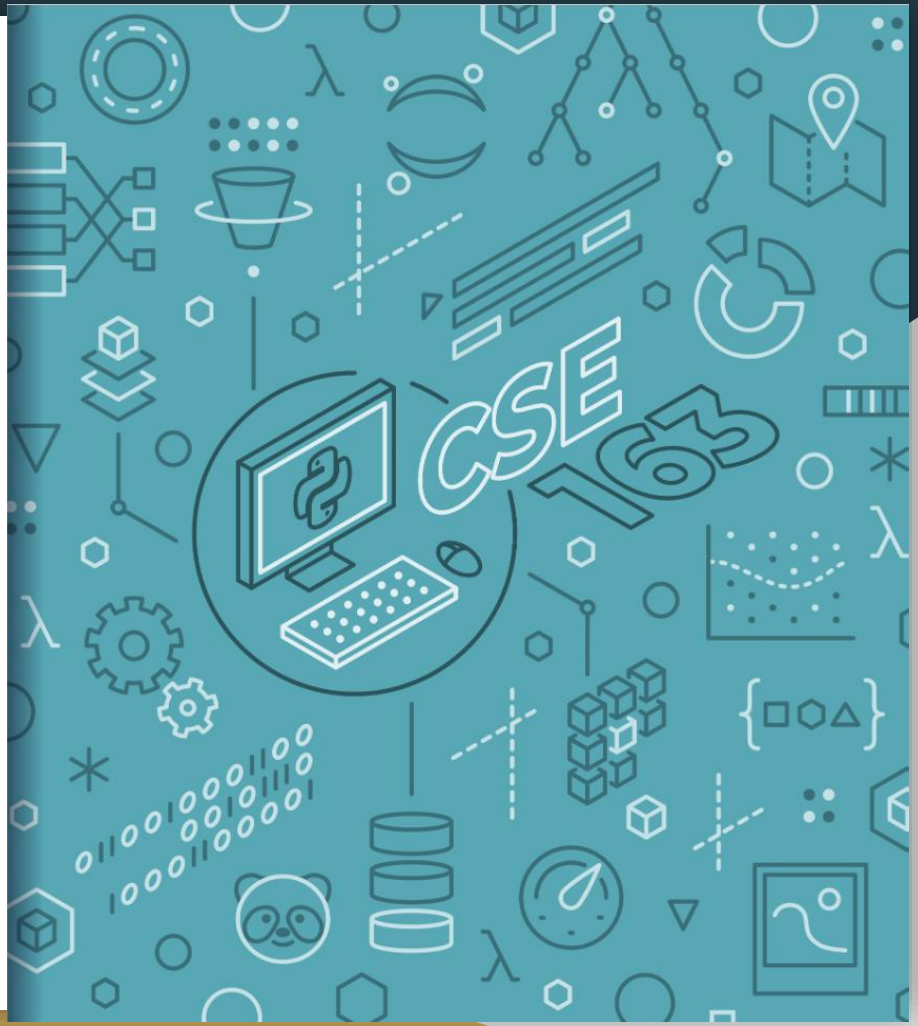
CSE 163
Object-Oriented Programming I

Adrian Salguero
Spring 2026

💭 Icebreaker (discuss with neighbors): ‘
Favorite outdoor area in Seattle or Washington?
Add to our Slido!

slido.com
#cse163



Announcements

- **Take Home Assessment 2: Networks will be released later today and will be due on Gradescope Thursday, April 23rd at 11:59pm!**
- **THA1 Peer Review assignments will be sent out tonight and due Wednesday, April 15th at 11:59pm!**
- **Lesson 5 Canvas Quiz due tonight at 11:59pm!**
- **Reading Assignment 2 due Monday April 13th at 11:59pm**
- **Section Assignment 2 is due tonight at 11:59pm!**
- **Resubmissions for THA 1 now open!**

Announcements (cont.)

- Can (re)submit THA 1
  - Open from April 10th - April 14th
  - Only technical component will be graded
  - Refer to Ed posts for resubmission guidelines and form
  - **MUST COMPLETE AND SUBMIT FORM FOR RESUB TO BE COUNTED!**

- THA 1 Peer Reviews will be released later today!
  - Gradescope submission!
  - 8 / 10 points are for completing the review
  - 2 / 10 points are for being thoughtful and thorough

Peer Reviews

- To find your assigned reviews
  - Navigate to your section
  - Find your name
  - You will see a row with two ID numbers and two links
  - **Enter the ID in Gradescope and review the notebook in the link!**

- Giving feedback
  - ***I Like***: Praise and positive comments
  - ***I Wish***: Constructive feedback
  - ***What If***: Suggestions (can be wacky, but be respectful)

Enter the Spider-verse!

Lesson Recap

- An ***object*** stores state and provides behaviors that operates on a state.
- Each object has its own state, but objects of the same class can have the same behaviors.

State:
`self.name`

Behavior:
`.thwip()`

Objects

```
spider_man = SpiderMan('Peter Parker')
amazing_sm = SpiderMan('Peter Parker')
```

State:
`self.name`

Behavior:
`.thwip()`

Objects

```
spider_man = SpiderMan('Peter Parker')
amazing_sm = SpiderMan('Peter Parker')
```

`spider_man`

State:
`'Peter Parker'`

Behavior:
`.thwip()`

`amazing_sm`

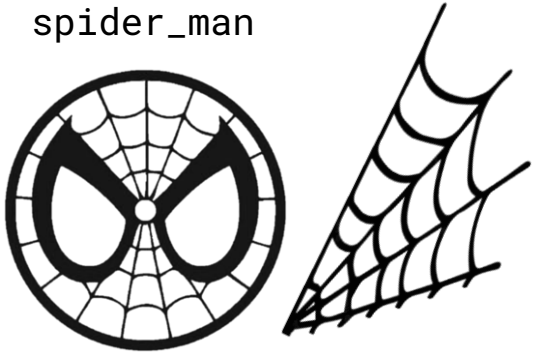
State:
`'Peter Parker'`

Behavior:
`.thwip()`

Objects

```
spider_man = SpiderMan('Peter Parker')
amazing_sm = SpiderMan('Peter Parker')
spider_man.thwip()
```

`spider_man`



State:
`'Peter Parker'`

Behavior:
`.thwip()`

`amazing_sm`

State:
`'Peter Parker'`

Behavior:
`.thwip()`

References

```
spider_man = SpiderMan('Peter Parker')
amazing_sm = spider_man
spider_man.thwip()
```

`spider_man`

State:
`'Peter Parker'`

Behavior:
`.thwip()`

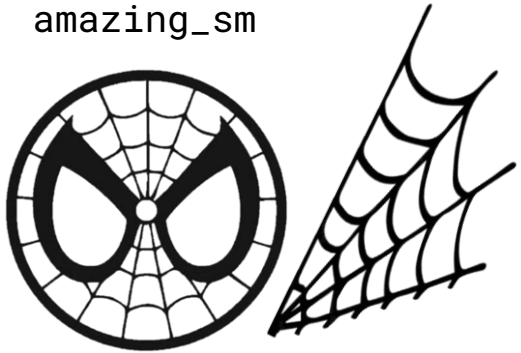
`amazing_sm`

State:
`'Peter Parker'`

Behavior:
`.thwip()`

```
spider_man = SpiderMan('Peter Parker')
amazing_sm = spider_man = SpiderMan('Peter Parker')
spider_man.thwip()
```

spider_man
amazing_sm



State:
'Peter Parker'

Behavior:
.thwip()

Multiple References!

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

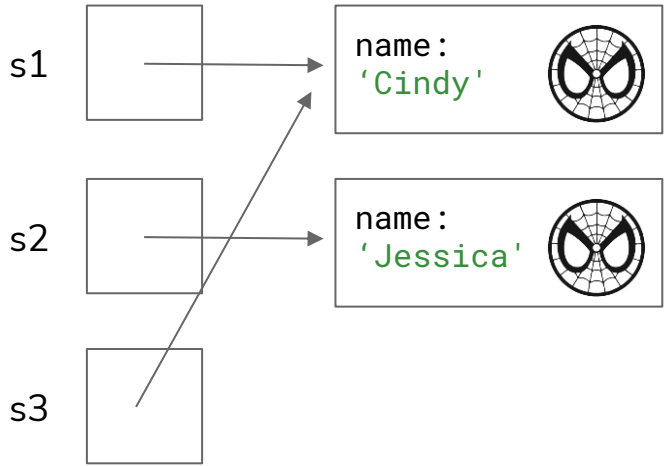
```
class SpiderFolk:

  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)**

```
s1 = SpiderFolk('Cindy')
s2 = SpiderFolk('Jessica')
s3 = s1
s1.thwip()
s2.thwip()
s3.thwip()
```



What is self?

- In the SpiderFolk example, every method (initializer or instance method) takes a parameter called ***“self”***
- This indicates which instance of the method is being called on

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

  def thwip(self):
    print(self.name + ': Thwip!')
```

```
s1 = SpiderFolk('Cindy')
s2 = SpiderFolk('Jessica')
s3 = s1

s1.thwip()
s2.thwip()
s3.thwip()
```

Who is ***self*** referring to for each call to ***thwip()?***

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.