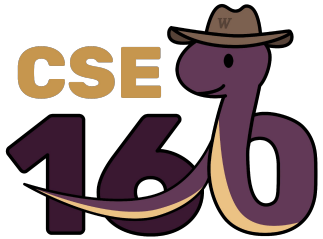


Classes and Objects

CSE 160 Summer 2026

Questions?

Announcements

- **Homework Assignment 4 due tonight, August 3 at 11:59 PM**
- **Programming Practice 6 due Sunday, August 9 at 11:59 PM**
- **Resubmission Cycle 3 due Sunday, August 9th at 11:59pm**
  - Homeworks 1-3 eligible
  - Fill out the Google Form!

Today's Roadmap

1. Classes Recap
2. Warmup
3. Attributes
4. Examples

Jupyter Hub

Classes

Classes** provide the blueprint for creating an **object

Class Blueprint

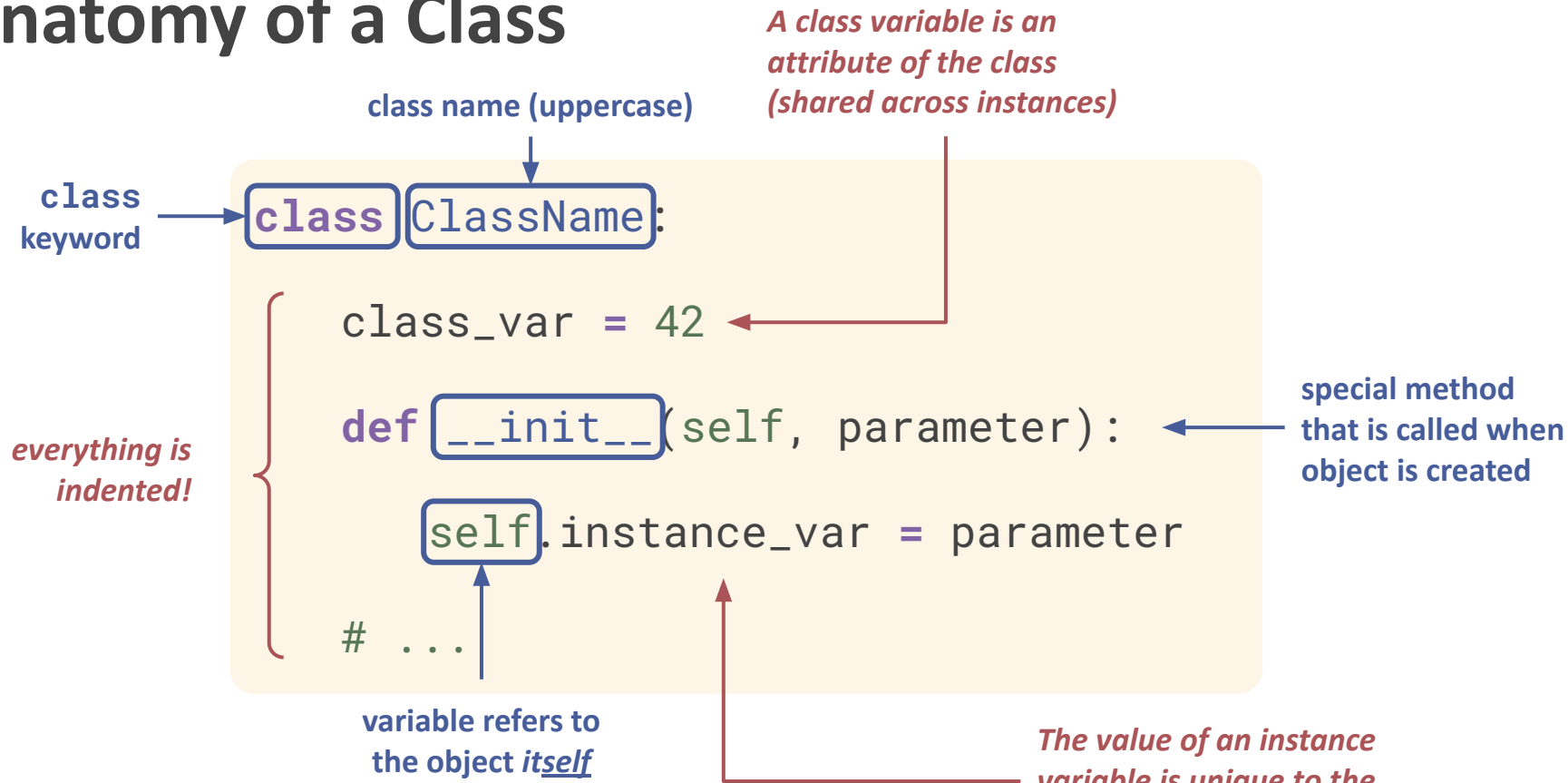
Specific Object

.evolve()

```
class Pokemon:
    ...
```

```
charmander =
Pokemon("Charmander", "Fire")
```

Anatomy of a Class



Think Pair Share

Consider the **Pokemon** class:

```
class Pokemon:

    hp = 20

    def __init__(self, name):
        self.name = name

    def evolve(self):
        self.hp *= 2

    def evolve_all(self):
        Pokemon.hp *= 3
```

Sli.do

#cse160

```
eevee = Pokemon("Eevee")
bulbasaur = Pokemon("Bulbasaur")

eevee.evolve()
eevee.evolve_all()
```

What is the value of **`bulbasaur.hp`** ***after the above code is executed?***

A	20	C	60
B	40	D	80

What's Going On Here??

```
class Pokemon:

    hp = 20

    def __init__(self, name):
        self.name = name

    def evolve(self):
        self.hp *= 2

    def evolve_all(self):
        Pokemon.hp *= 3
```

```
eevee = Pokemon("Eevee")
bulbasaur = Pokemon("Bulbasaur")

eevee.evolve()
eevee.evolve_all()
```

Python Tutor

Attributes

Attributes are (just) variables!

Evaluation

When looking for an attribute, Python **starts with *instance* variables**.

(in the case of `bulbasaur`*, there is no instance variable named* **hp***)*

If no instance variable exists, look for a ***class* variable**.

```
class Pokemon:

    hp = 20

    def __init__(self, name):
        self.name = name

    def evolve(self):
        self.hp *= 2

    def evolve_all(self):
        Pokemon.hp *= 3
```

Assignment

`self.attribute =`

Creates an instance variable (if it doesn't already exist)

`Class.attribute =`

Modifies the class variable (for everyone!)

Think Pair Share

The **`University`** class has the following **attributes**:

- **`name`**, a string
- **`mascot`**, a string
- **`enrollment`**, an int
- **`depts`**, a list of strings

Implement the following methods:

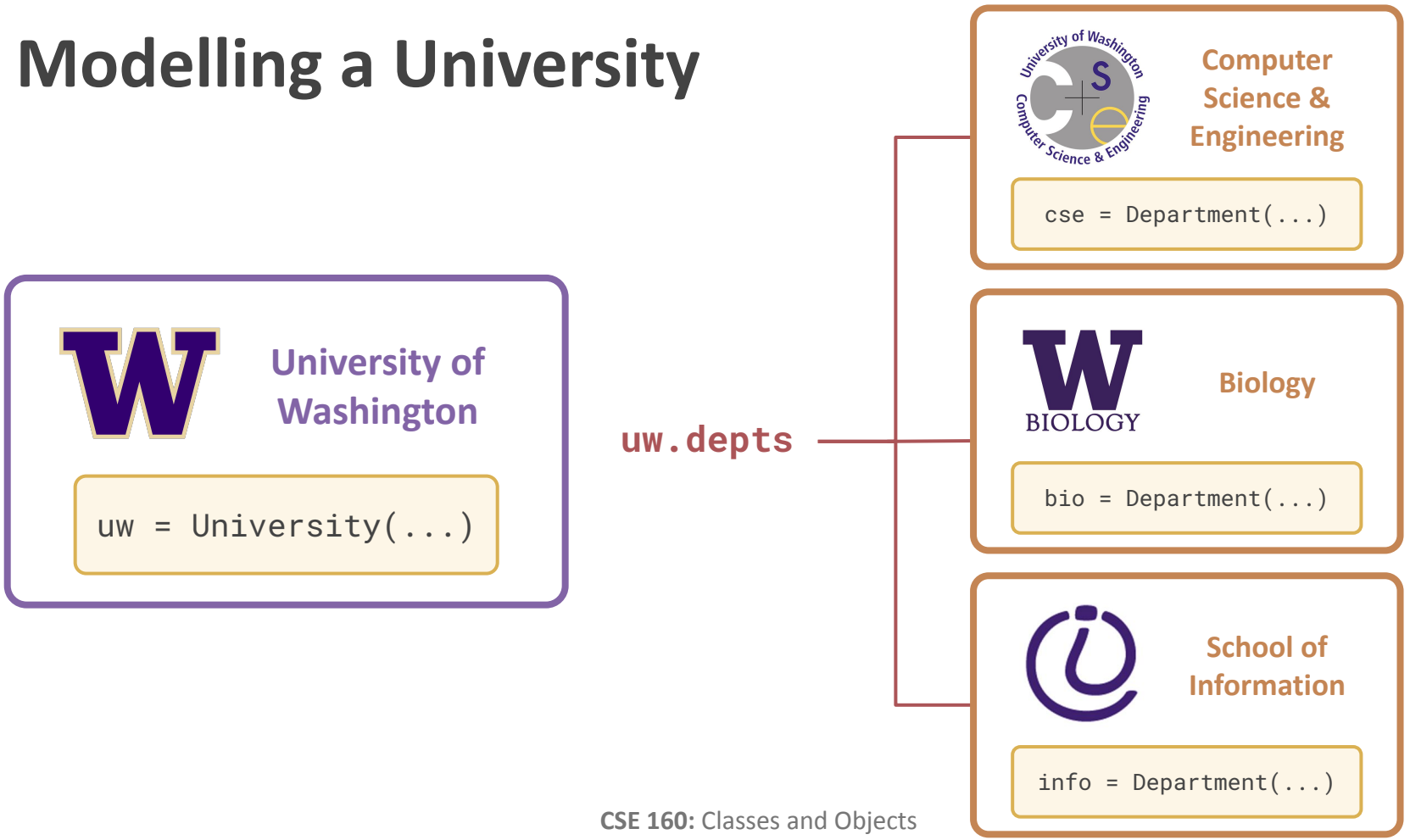
- **`welcome()`**, that prints:
  "[mascot] welcomes you to [name]"
- **`add_dept(new_dept)`**, that adds new_dept to the depts list

```
class University:

    def __init__(self, name, mascot,
                       enrollment):

        self.name = name
        self.mascot = mascot
        self.enrollment = enrollment
        self.depts = []
```

University of Washington

```
uw = University(
      "University of Washington",
      "Dubs",
      63736)
```

Modelling a University



Objects containing objects?

Why not!

Dunder Methods

Double underscore (*d-under*) methods serve special purposes in Python:

`__init__()`	Called when an instance of a class (an object) is created. Used to **initialize** the instance.
`__str__()`	Called by the `str()` and `print()` functions. Used to provide a ***readable* string** representing the object.
`__repr__()`	Helpful for debugging. Used to provide the **official string representation** of the object.