Spring 2022

Class 19 - Union Types, Default Parameters, OOP Practice

Announcements

- RD01 Posted Monday - Weapons of Math Destruction
  - Due Sunday 4/10
- EX08 Posted Yesterday - Analysis for Continuous Improvement
  - Due Tuesday 3/29
  - Requires completion of EX07 - Finish this up ASAP and hand-in late!
- QZ03 moved back by one week
  - This unit's focus on Object-oriented Programming needed more time.
  - New Date: Thursday, April 7th
  - Taking with ARS? Reschedule your quiz reservation *TODAY!*

Hack 110

- Optional 2-day Event Aimed at Prospective Majors
  - Work in teams of up to 2 people
  - Need a teammate? Meet peers in the workshop!
  - For street credit only, not for course credit. Great for resume building!

- Friday, April 1st from 6pm to 8pm - Required Workshops!
  - Web Development - How to make an interactive web page with Flask, HTML, CSS
  - Game Development - How to build a game with Pygame

- Friday, April 8th from 7pm to 7am - Hackathon
  - Additional workshops
  - Fun events
  - Food and more!

- RSVP for HACK110 Workshop + Hackathon required by end-of-day TODAY
  - https://bit.ly/hack11022s

```python
class Dog:
    name: str

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

    def speak(self) -> str:
        return f"{self.name}: woof"


class Cat:
    name: str

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

    def speak(self) -> str:
        return f"{self.name}: meow"


fido: Cat = Cat("Cleo")
leo: Dog = Dog("Loki")

print(fido.speak())
print(leo.speak())
```

Diagram 1

Produce an environment diagram of the code listing left.

```python
class Dog:
    name: str

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

    def speak(self) -> str:
        return f"{self.name}: woof"


class Cat:
    name: str

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

    def speak(self) -> str:
        return f"{self.name}: meow"


fido: Cat = Cat("Cleo")
leo: Dog = Dog("Loki")

print(fido.speak())
print(leo.speak())
```

```python
from __future__ import annotations


class Point:
    """Model a 2D Point."""

    x: float
    y: float

    def __init__(self, x: float, y: float):
        """Initialize a Point with its x, y components."""
        self.x = x
        self.y = y

    def scale_by(self, factor: float) -> None:
        """Mutates: multiplies components by factor."""
        self.x *= factor
        self.y *= factor

    def scale(self, factor: float) -> Point:
        """Pure method that does not mutate the Point."""
        scaled: Point = Point(self.x * factor, self.y * factor)
        return scaled


p0: Point = Point(1.0, 2.0)
p0.scale_by(2.0)
p1: Point = p0.scale(2.0)
print(f"p0: (({p0.x}, {p0.y})) - p1: ({p1.x}, {p1.y})")
```

Diagram 2

Produce an environment diagram of the code listing left.

```python
from __future__ import annotations


class Point:
    """Model a 2D Point."""

    x: float
    y: float

    def __init__(self, x: float, y: float):
        """Initialize a Point with its x, y components."""
        self.x = x
        self.y = y

    def scale_by(self, factor: float) -> None:
        """Mutates: multiplies components by factor."""
        self.x *= factor
        self.y *= factor

    def scale(self, factor: float) -> Point:
        """Pure method that does not mutate the Point."""
        scaled: Point = Point(self.x * factor, self.y * factor)
        return scaled


p0: Point = Point(1.0, 2.0)
p0.scale_by(2.0)
p1: Point = p0.scale(2.0)
print(f"p0: (({p0.x}, {p0.y})) - p1: ({p1.x}, {p1.y})")
```