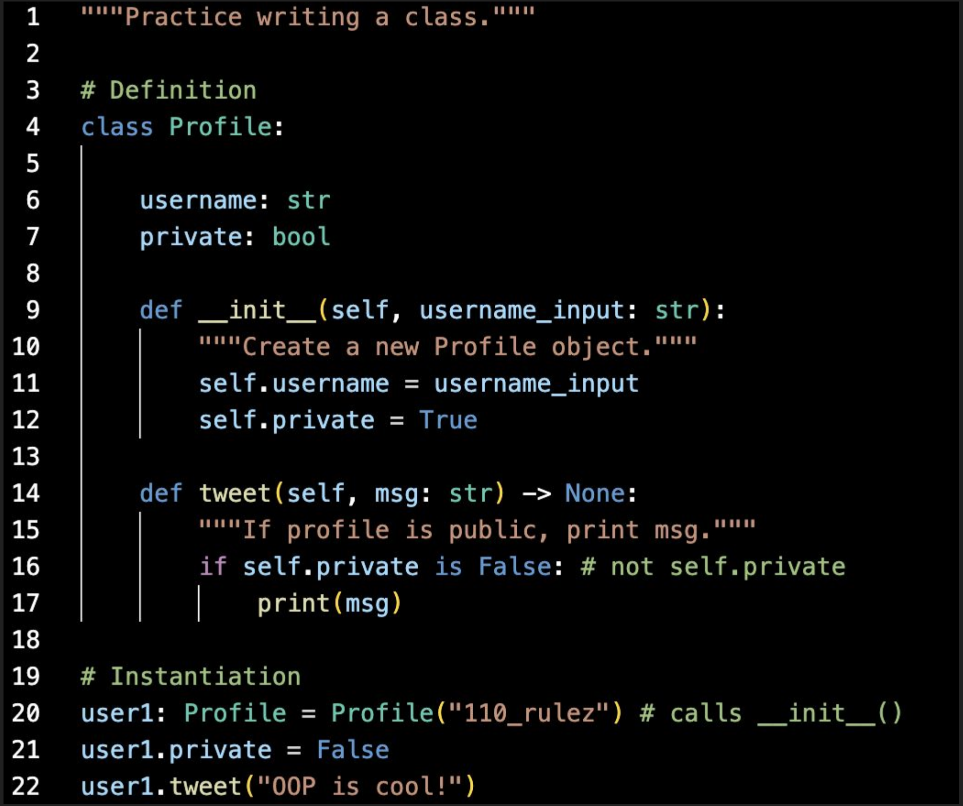
COMP 110

Magic Methods + Operator Overloads

Review

What are unique properties of the __init__ method? (What sets it apart from other methods?)

Review



```python
"""Practice writing a class."""

# Definition
class Profile:

    username: str
    private: bool

    def __init__(self, username_input: str):
        """Create a new Profile object."""
        self.username = username_input
        self.private = True

    def tweet(self, msg: str) -> None:
        """If profile is public, print msg."""
        if self.private is False: # not self.private
            print(msg)

# Instantiation
user1: Profile = Profile("110_rulez") # calls __init__()
user1.private = False
user1.tweet("OOP is cool!")
```

Magic Methods

- Methods with built in functionality!
- Not called *directly!*
- Names start and end with two underscores (__<method_name>__)

Question

When I call print(x), Python calls what magic method on x *before* printing?

Operator Overloads

- You can write magic methods to give operators meaning!
- Think about operators you use on numbers that you'd like to use on other objects, e.g. +, -, *, /, <, <=, etc…
- This is called operator overloading

Arithmetic Operator Overloads

+	__add__(self, other)
–	__sub__(self, other)
*	__mul__(self, other)
/	__truediv__(self, other)
**	__pow__(self, other)
%	__mod__(self, other)

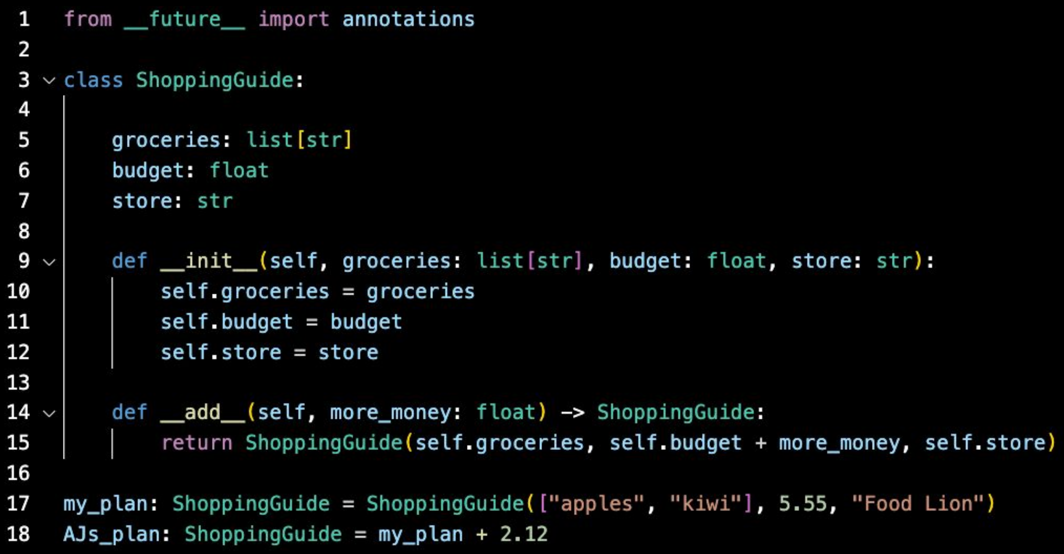
Comparison Operator Overloads

<	__lt__(self, other)
>	__gt__(self, other)
<=	__le__(self, other)
>=	__ge__(self, other)
==	__eq__(self, other)
!=	__ne__(self, other)

For each magic method call, what is self and (if applicable) what is other?

str(a)	__str__(self)
a + b	__add__(self, other)
a – b	__sub__(self, other)
a * b	__mul__(self, other)
a < b	__lt__(self, other)
a == b	__eq__(self, other)

Diagramming



```python
from __future__ import annotations

class ShoppingGuide:

    groceries: list[str]
    budget: float
    store: str

    def __init__(self, groceries: list[str], budget: float, store: str):
        self.groceries = groceries
        self.budget = budget
        self.store = store

    def __add__(self, more_money: float) -> ShoppingGuide:
        return ShoppingGuide(self.groceries, self.budget + more_money, self.store)

my_plan: ShoppingGuide = ShoppingGuide(["apples", "kiwi"], 5.55, "Food Lion")
AJs_plan: ShoppingGuide = my_plan + 2.12
```

Extra Challenge

- Write a __str__ magic method that gives me all the information of a ShoppingGuide object
- Change the __add__ magic method to add a list of more groceries instead of adding money to the budget. (Note that it still shouldn’t modify self!)

Challenge Question!

*You are going to use union types so review those!

Union Types

Now that I have:

```python
def add(x: int, y: int = 1) -> int:
    return x + y
```

Say I want this function to work for ints *or* floats…

I can express this using Union:

```python
def add(x: int | float, y: int | float = 1) -> int | float:
    return x + y
```