Intro to OOP

Olivia Lynn
LSST-DA Data Science Fellowship Program Session 21
June 3, 2024

Intro to
Intro to OOP

Olivia Lynn
(she/any)

Software Dev at
LINCC Frameworks

MONDAY

Talk: 3:30 - 4:30 PM

Work: 4:30 - 5:30 PM

WEDNESDAY

Work: 9 - 10 AM | break | 10:30 AM - 12 PM

Optional deep dives throughout

→ UML, SOLID Principles, Composition vs Inheritance, etc

THURSDAY

Talk: 9 - 10 AM

Work: 10:30 AM - 12 PM

→ Unit Testing w/ GitHub Actions, PEP 8, Docstrings, ReadTheDocs, etc

Objects

What is OOP?

- A programming paradigm based on objects
  - Popular paradigms: procedural, functional, object-oriented
  - Popular languages w/ OOP capabilities: Python, Java, C++, C#

What is an object?

- Instantiation of a class, which is a set of blueprints for making objects
- Objects have <u>attributes</u> and <u>methods</u>

Terminology can vary

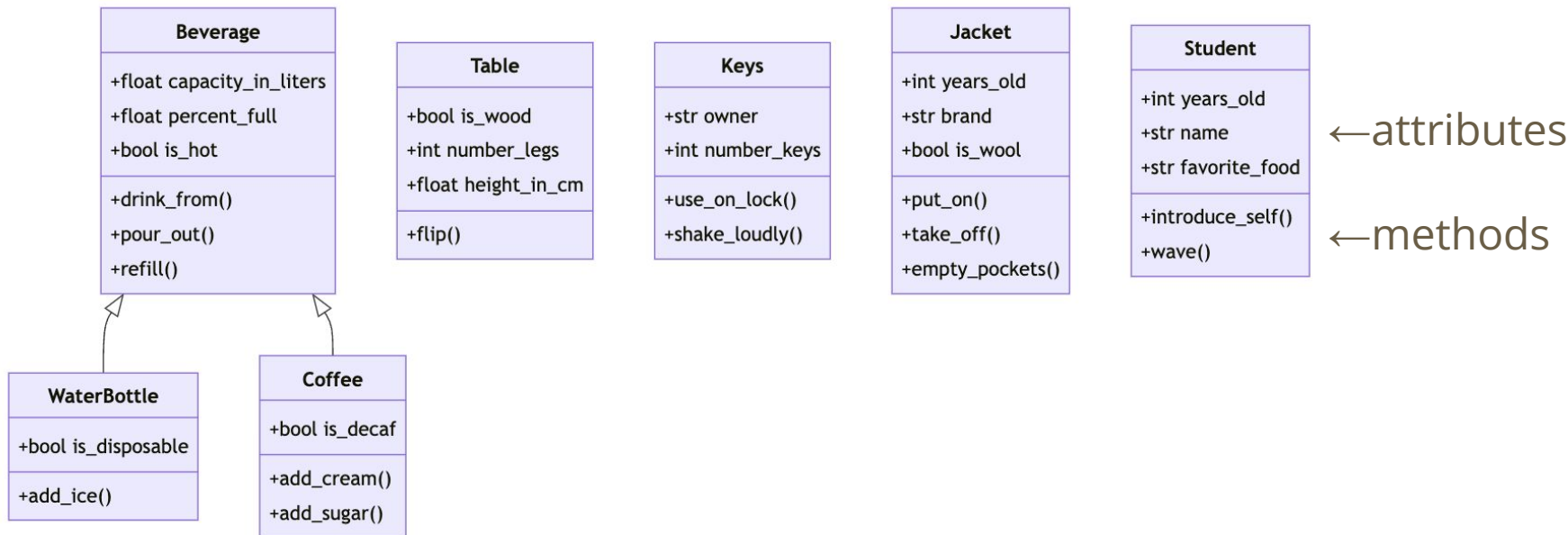| Concept	Python	Java	C++	C#
Data within an object	Attribute	Field	Member	Field
Functions in an object	Method	Method	Method	Method
Class-level variables	Class Attribute	Static Field	Static Member	Static Field
Class-level functions	Class Method	Static Method	Static Method	Static Method
Creating an object	Instantiation	Instantiation	Instantiation	Instantiation
Class blueprint	Class	Class	Class	Class

Laptop

Attributes

- How heavy it is
- How old it is
- Who it belongs to
- All of its specs would be attributes, too

Methods

- Turning on and off
- Displaying information
- Making noise
- Short circuiting, if you pour your drink on it
- Any behavior!

More Objects

Beverage
+float capacity_in_liters
+float percent_full
+bool is_hot

+drink_from()
+pour_out()
+refill()

Table
+bool is_wood
+int number_legs
+float height_in_cm

+flip()

Keys
+str owner
+int number_keys

+use_on_lock()
+shake_loudly()

Jacket
+int years_old
+str brand
+bool is_wool

+put_on()
+take_off()
+empty_pockets()

Student
+int years_old
+str name
+str favorite_food

+introduce_self()
+wave()

←attributes

←methods

WaterBottle
+bool is_disposable

+add_ice()

Coffee
+bool is_decaf

+add_cream()
+add_sugar()

Getting Conceptual

Objects can also represent more abstract or conceptual entities, such as roles, responsibilities, or parts of a system's architecture

→eg: managers, controllers, views, and services

```python
class EmailService:
    def send_email(self, recipient, subject, message):
        print(f"Sending email to {recipient}\nSubject: {subject}\nMessage: {message}")

# Usage
email_service = EmailService()
email_service.send_email("user@example.com", "Welcome!", "Thank you for signing up.")
# Output: Sending email to user@example.com
#         Subject: Welcome!
#         Message: Thank you for signing up.
```

The Pillars of OOP: Live Demo

https://github.com/LSSTC-DSFP/Session-21/blob/main/Day1/OOP_Demo.ipynb