Computational Structures in Data Science

Object-Oriented Programming: Inheritance

UC Berkeley

Announcements

- Ants Project
  - New Written Q's
  - New Oral Check-Offs
- Midterm Regrade Requests due tonight

Computing(-ish) in the News

- Two CS-adjacent Articles
- "There's a Good Reason You Can't Concentrate" (Cal Newport, NYT Op-Ed)
- "Juries Take the Lead in the Push for Child Online Safety" (NYT

AI Examples

- Meant to share this before break…
- https://claude.ai/share/7b6b317b-bea4-4fa4-8463-d9265ad80a00
  - Some useful examples asking Claude to *study* and explain things
- https://claude.ai/share/21075111-d24b-49c2-a932-d2b2757834bd
  - Another example – Ask AI to give you more or less scaffolding to sole some problems.
  - (Claude seems to figure out from other sources I'm talking about C88C, but if you need to give it more background)

Computational Structures in Data Science

Object-Oriented Programming: Inheritance

UC Berkeley

Learning Objectives

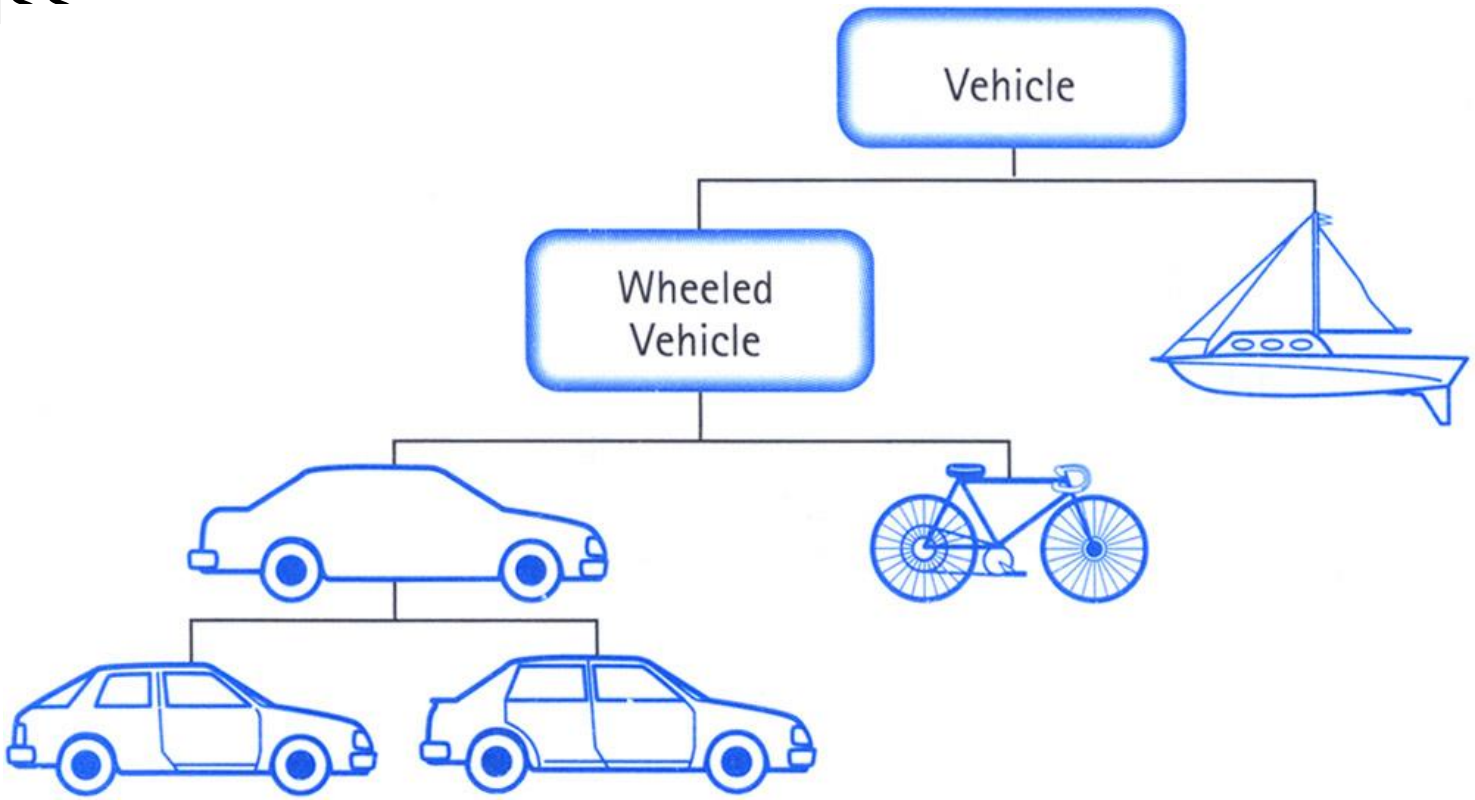
- Inheritance allows classes to reuse methods and attributes from a parent class.
- super() is a new method in Python
- Subclasses or child classes are distinct from on another, but share properties of the parent.

Inheritance

- Define a class as a specialization of an existing class
- Inherent its attributes, methods (behaviors)
- Add additional ones
- Redefine (specialize) existing ones
  - Ones in superclass still accessible in its namespace

Class Inheritance

- Classes can inherit methods and attributes from parent classes but extend into their own class



Python class statement

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

```
class ClassName ( inherits / parent-class ):
    <statement-1>
    .
    .
    .
    <statement-N>
```


Example

```
class BaseAccount:
    def __init__(self, name, initial_deposit):
        # Initialize the instance attributes
        self._name = name
        self._acct_no = Account._account_number_seed
        Account._account_number_seed += 1
        self._balance = initial_deposit

class CheckingAccount(BaseAccount):
    def __init__(self, name, initial_deposit):
        # Use superclass initializer
        BaseAccount.__init__(self, name, initial_deposit)
        # Alternatively:
        # super().__init__(name, initial_deposit)
        # Additional initialization
        self._type = "Checking"
```


Accessing the Parent Class

- `super()` *binds* methods in the parent or "superclass" to the current instance
  - Can be called anywhere in our class
  - Handles passing `self` to the method
  - Handles looking up an attribute on a parent class, too.
- We can directly call `ParentClass.method(self, …)`
  - This is not quite as flexible if our class structure changes.
- In general, prefer using **`super()`**!
- Outside of C88C, things can get complex…
  - https://docs.python.org/3/library/functions.html#super

Making A New Instance

- **We almost always want to call `__init__` on the parent class!**
- Python will not do this automatically!

How will our accounts evolve?

- A CheckingAccount, which is mostly a BaseAccout, with a new tyoe
- A SavingsAccount which allows us to accurse interest
- All savings accounts should have the same interest rate

Questions to consider:

- How do methods in child classes interact with methods in the parent class?
- How should we write our parent class (BaseAccount) to take advantage of features of our child classes?

Demo

- BaseAccount → CheckingAccount + SavingsAccount
- How do we make a new checking account?

Question

- What happens if we do not call __init__ on the parent class?

Discuss and make a guess!

A) Will the CheckingAccount be valid?
B) Will we get an immediate error?
C) Will some methods work but not others?

Question

- What happens if we try to withdraw() from a SavingsAccount?

Computational Structures in Data Science

Object-Oriented Programming: Evolving The Bank Model

UC Berkeley

Michael Ball | UC Berkeley | https://c88c.org |

Composing Classes Together

- Currently, our `BaseAccount` stores a lot of data in class attributes...
- This suggests we are trying to accomplish an entirely new kind of class, or object
  - A Bank!
- We should extract that these functions into their own class
- A bank can now manage:
  - making accounts
  - keeping track of account numbers
  - showing and listing accounts

Live Demo