Class 8 - Object Oriented Programming

MISE Summer Programming Camp 2023

Detour: Announcement

At the end of this class, **10 students** will be selected to participate in the MISE Summer Camp

Your performance will help decide who gets selected

So we are changing the rules on the class project and it is now a mandatory part of the class

Class Project

Class Project

Motivation: explore your creativity, problem-solving skills, and proficiency in Python

Goal: implement **1 Python program** that does anything (open ended)

Due date: June 24 (so you have one week, start early!)

Time expectation: it's up to you, it should take you at least 1 hour, but take as much as you want

Be creative! Do something you'll have fun doing!

Project Timeline

- **Do some research on possible ideas**
  - Look for inspiration in your hobbies and other interests
- **Decide what you want to work on and write a short description**
  - Make sure you clearly define the project early on
- **Implement your project**
  - Use your IDE and write code!
- **Submit your project to piazza (a thread will be created for that)**

Project Ideas

Data Analysis and Visualization

Analyze and visualize a dataset of interest, such as population, financial data, or sports statistics

Game Development

Create a simple game. It can be either a simple text-based thing or more interactive

Encryption and Decryption *

Create a program that encrypts and decrypts text using encryption algorithms

Application Development

Build some application, e.g. a todo list that helps users manage their tasks

Web Scraper *

Develop a program that extracts data from websites, to gather information for research

Machine Learning *

Train a machine learning model, such as email spam filtering or sentiment classification

See Worksheet 3 for more on libraries

Some Helpful Libraries *(optional)

Data Analysis and Visualization

Matplotlib: plotting
Pandas: data manipulation
NumPy: numerical computing

Game Development

Pygame: library for game development, with graphics, sound, and user input features

Encryption and Decryption

cryptography: A library that provides various cryptographic algorithms.

Application Development

Web Scraper

Beautiful Soup: easy extraction of data from HTML
Requests: making HTTP requests

Machine Learning

Scikit-learn: A comprehensive library for machine learning

Tips on how to succeed

- **Simple is better**
  - A complete simple project is better than an incomplete advanced project
- **Use the internet**
  - Don’t be afraid to use Google to find ideas and help
- **Be modular**
  - Don’t write everything at once, break the project into smaller, manageable tasks.
- **Test often**
  - Test your code at every point of the process, don’t wait until the end to test
- **Accept that not everything will work**
  - Some of your ideas won’t work, that’s ok! It doesn’t have to be perfect, move on
- **Ask for help**
  - Post on Piazza if you need help!

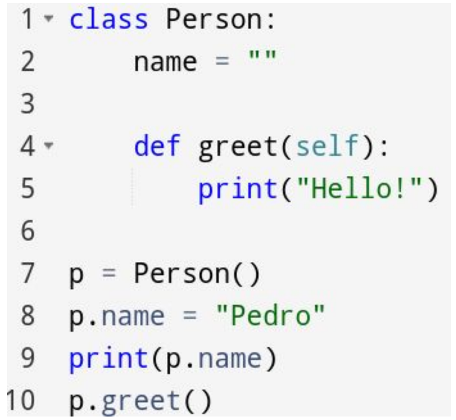
Object Oriented Programming

Objects and Classes

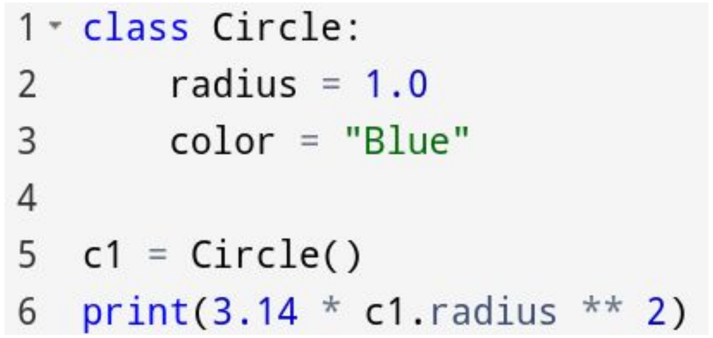
Variables in Python are all *Objects* (so an *Object* is the most generic data type)

In practice they are collections of **attributes and methods**

A **class** is a blueprint or template that defines the structure and behavior of objects



```python
class Person:
    name = ""

    def greet(self):
        print("Hello!")

p = Person()
p.name = "Pedro"
print(p.name)
p.greet()
```



```python
class Circle:
    radius = 1.0
    color = "Blue"

c1 = Circle()
print(3.14 * c1.radius ** 2)
```

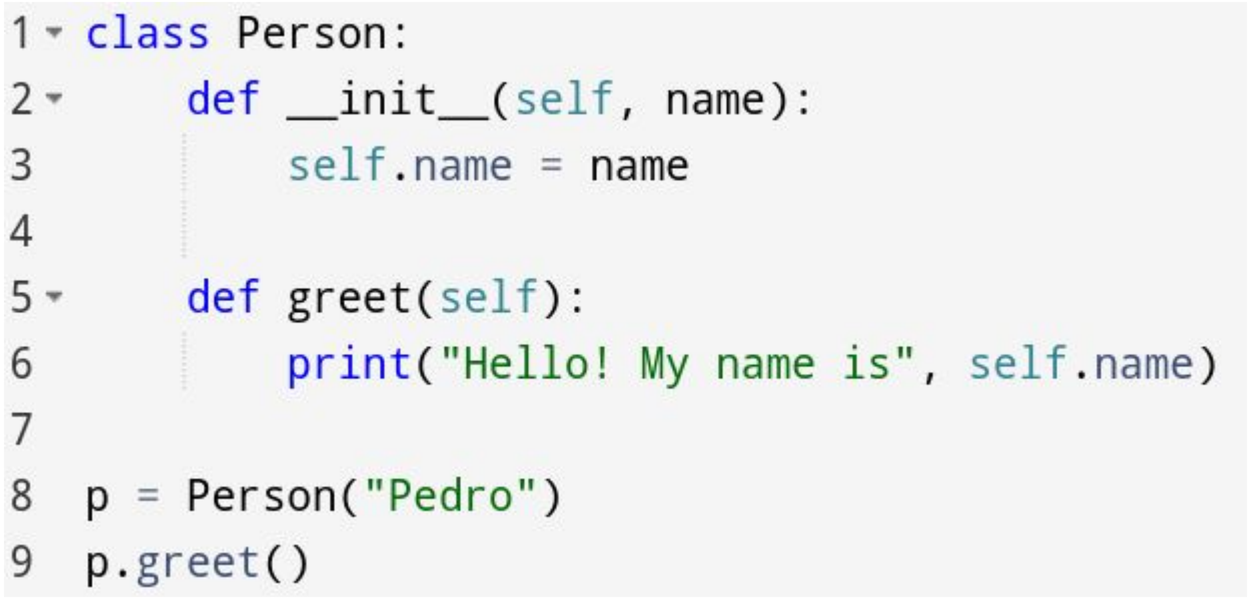
Why Classes?

- **Encapsulation**: Classes let us collect attributes and methods into a single unit
- **Code Reusability**: Once a class is defined, we can use it to create multiple objects
- **Abstraction**: A way to represent real-world or abstract concepts in code
- **Inheritance**: Creating new classes based on existing ones
- **Polymorphism**: Ability of objects of different classes to respond to the same method in different ways *(we are not going to look too much into this)

Self and init

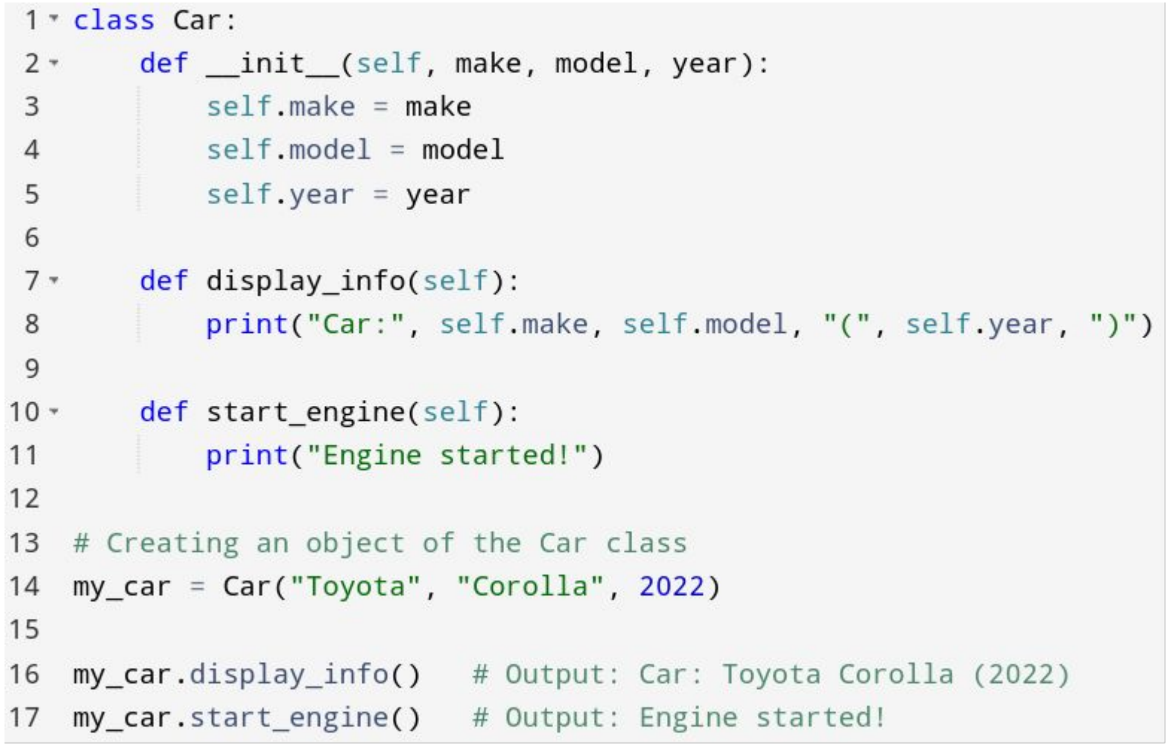
- *self* is used to access attributes and methods of a class in python
- *self* is a convention and not a python keyword
- Class functions must have an extra first parameter in the method definition
- __*init*__(also called constructor) function of a class is invoked when we create an object variable or an instance of the class

A longer Person example



```python
class Person:
    def __init__(self, name):
        self.name = name

    def greet(self):
        print("Hello! My name is", self.name)

p = Person("Pedro")
p.greet()
```

More examples

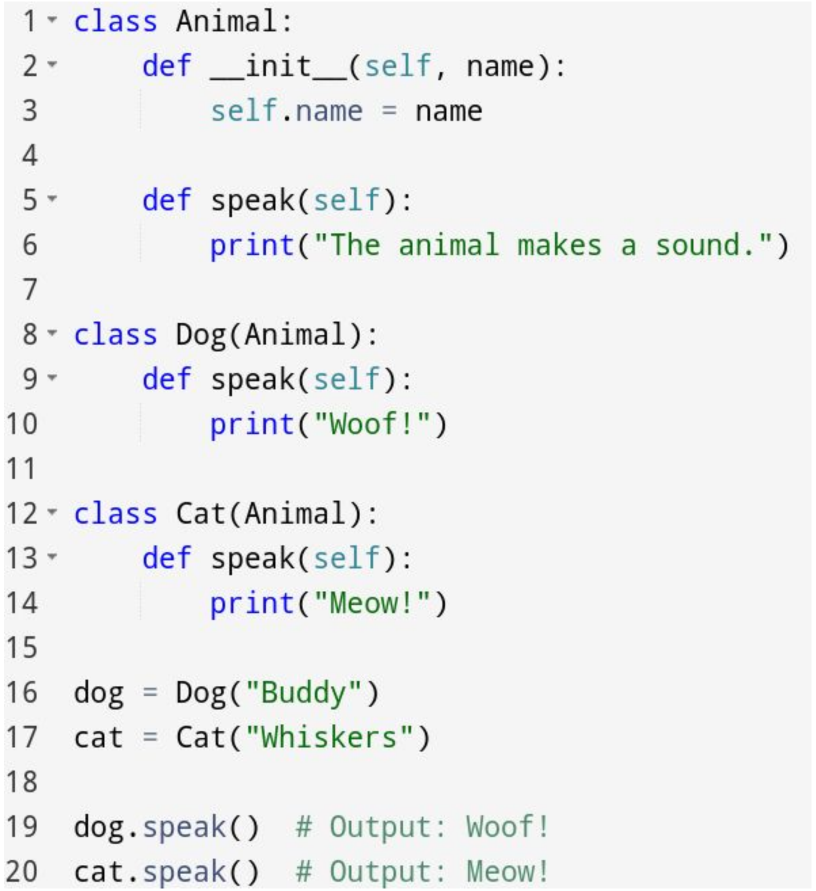


```python
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display_info(self):
        print("Car:", self.make, self.model, "(", self.year, ")")

    def start_engine(self):
        print("Engine started!")

# Creating an object of the Car class
my_car = Car("Toyota", "Corolla", 2022)

my_car.display_info()   # Output: Car: Toyota Corolla (2022)
my_car.start_engine()   # Output: Engine started!
```

Inheritance

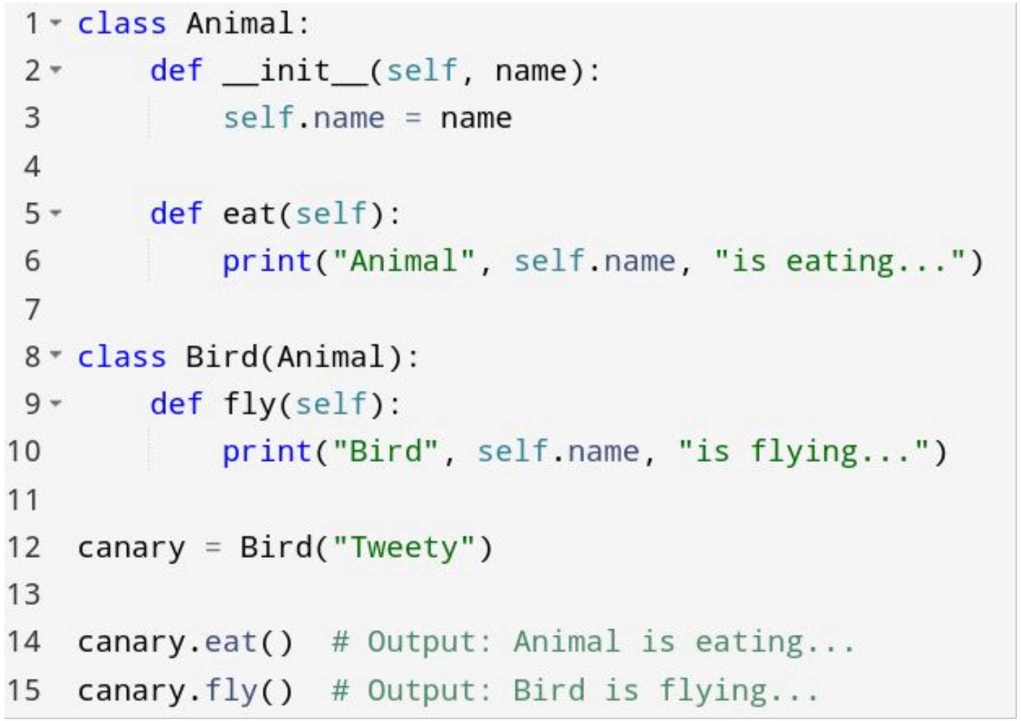
- Inheritance allows derived classes (child classes) to inherit attributes and methods from a base class (parent class)
- Every class inherits from a built-in basic class called ‘object’
- Inheritance is achieved by specifying the base class name in parentheses after the derived class name during class declaration
- If a method is defined in both the base class and the derived class, the method in the derived class overrides the one in the base class

More examples



```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("The animal makes a sound.")

class Dog(Animal):
    def speak(self):
        print("Woof!")

class Cat(Animal):
    def speak(self):
        print("Meow!")

dog = Dog("Buddy")
cat = Cat("Whiskers")

dog.speak()  # Output: Woof!
cat.speak()  # Output: Meow!
```

More examples



```python
class Animal:
    def __init__(self, name):
        self.name = name

    def eat(self):
        print("Animal", self.name, "is eating...")

class Bird(Animal):
    def fly(self):
        print("Bird", self.name, "is flying...")

canary = Bird("Tweety")

canary.eat()  # Output: Animal is eating...
canary.fly()  # Output: Bird is flying...
```

What’s next?

No worksheet/homework this week! Work on your project!

Class 9: Intro to Data Science

What is Data Science?

Collecting, Analyzing and Plotting data with Python