

# Classes and OOP

Guttag Chapter 10

# Goals

- Understand terms and syntax related to defining classes
- Understand terms and syntax related to using classes
- Explore Python Notebook Examples

# Terms and Syntax for Defining Classes

Guttag Chapter 10

# Class

## Definition

- blueprint for objects!
- Something in code that can flexibly define any useful object

## Details

- Classes define how information is stored
- Classes define how information can be accessed
- Classes define how information can be updated

# Class

## Keyword

- **class**

## Example

- **class** Vehicle():

```
class Vehicle():  
    """Abstract data type representing a vehicle."""
```

# Constructor

## Definition

- a special function inside a class that creates objects with data

## Details

- the name is always `__init__`
- the first parameter is conventionally ``self``
- variables, also known as **attributes**, can store information

```
class Vehicle():  
    """Abstract data type representing a vehicle."""  
  
    def __init__(self, num_seats: int, num_doors: int, engine_type: str):  
        """Define the constructor."""  
        self._seats = num_seats  
        self._doors = num_doors  
        self._engine = engine_type  
        self._milage = 0.0
```

# Methods

## Definition

- functions inside a class that handle the object data

## Details

- the first formal parameter is always `self`
- the function can be public or private
- names with `_` or `__` in front are not to be used directly
  - there are other ways to use private methods or **dunder** methods

# \_\_repr\_\_(self)

## Definition

- A dunder method that defines how an object is **repr**esented in text

## Example

- 

```
def __repr__(self):  
    """Define the printable representation of the vehicle."""  
    return f"{self._engine} vehicle with {self._seats} seats, " +\  
           f"{self._doors} doors, and {self._milage} miles."
```



# Methods

Constructor (private)

Method (public)

Method (public)

Dunder Method  
(private)

```
class Vehicle():
    """Abstract data type representing a vehicle."""

    def __init__(self, num_seats: int, num_doors: int, engine_type: str):
        """Define the constructor."""
        self._seats = num_seats
        self._doors = num_doors
        self._engine = engine_type
        self._milage = 0.0

    def drive(self, num_miles: float):
        """Add milage to the vehicle."""
        self._milage += num_miles
        return None

    def milage(self):
        """Get the milage of the vehicle."""
        return self._milage

    def __repr__(self):
        """Define the printable representation of the vehicle."""
        return f"{self._engine} vehicle with {self._seats} seats, " + \
            f"{self._doors} doors, and {self._milage} miles."
```

# Overloading

## Definition

- defining common functions that other types also have
- ==, +, >, <=, etc!
- Everything in a class must be defined, including how to add, subtract, compare

## Example

```
def __eq__(self, other_vehicle) -> bool:  
    """Define how to check for equality."""  
    return self.milage() == other_vehicle.milage()
```

# Overloading operators

- `+: __add__`
- `-: __sub__`
- `**: __pow__`
- `<<: __lshift__`
- `*: __mul__`
- `/: __truediv__`
- `//: __floordiv__`
- `?: __mod__`
- `|: __or__`
- `<: __lt__`
- `^: __xor__`
- `>: __gt__`
- `>>: __rshift__`
- `==: __eq__`
- `<=: __le__`
- `&: __and__`
- `!=: __ne__`
- `>=: __ge__`
- `str: __str__`
- `len: __len__`
- `hash: __hash__`
- `repr: __repr__` - <https://stackoverflow.com/questions/1436703/what-is-the-difference-between-str-and-repr>

# Terms and Syntax for Using Classes

Guttag Chapter 10

# Instantiate

## Definition

- To call the constructor function with special syntax

## Details

- the `__init__` function is called by using the class name!
- skip over the parameter ``self``

```
# Instantiate a vehicle with the constructor
```

```
new_sports_car = Vehicle(num_seats = 2, num_doors = 2, engine_type = "gas")
```

# Instance

## Definition

- An object that has been instantiated with actual data
- that object type is the class

## Example

```
# Instantiate a vehicle with the constructor
new_sports_car = Vehicle(num_seats = 2, num_doors = 2, engine_type = "gas")

type(new_sports_car)
```

### **Vehicle**

```
def __init__(num_seats: int, num_doors: int, engine_type: str)
```

Abstract data type representing a vehicle.

# Print

## Definition

- Print out information about an object by calling print.
- Under the hood, print uses the `__repr__` function

## Example

```
# Instantiate a vehicle with the constructor
new_sports_car = Vehicle(num_seats = 2, num_doors = 2, engine_type = "gas")

print(new_sports_car)
```

gas vehicle with 2 seats, 2 doors, and 0.0 miles.

# Dot notation

## Definition

- syntax to access public attributes and methods
- uses the object name, a dot (.) and the attribute or method name
- skip over `self`

## Example

```
# Instantiate a vehicle with the constructor
new_sports_car = Vehicle(num_seats = 2, num_doors = 2, engine_type = "gas")
# Use a vehicle method to drive the car
new_sports_car.drive(1000)

print(new_sports_car)
```

gas vehicle with 2 seats, 2 doors, and 1000.0 miles.



# Self

## Definition

- **self** is the conventional name given to the first formal parameter in class methods
- when any method is called, self refers to the instantiated object itself
- when any method is called, self can be skipped

# Explore Python Notebook

# Summary

- Classes allow programmers to define convenient data types and convenient data structures
- Everything must be defined by the programmer!
- Class instances contain actual data and methods that operate on that data

## Reminders

- Read Guttag Chapter 10
- Specification Lab on Matrix Processing out on Wed
- Engineering Lab on Object Processing out on Thurs
- Midterm 2 is Friday, Nov 8th, in class

# Summary

- attributes
  - values or data associated with an object
  - accessed with . notation
  - not callable
- methods
  - function that operates on the object (and it's attributes ^^^)
  - accessed with . notation
  - callable!

<https://stackoverflow.com/questions/46312470/difference-between-methods-and-attributes-in-python>