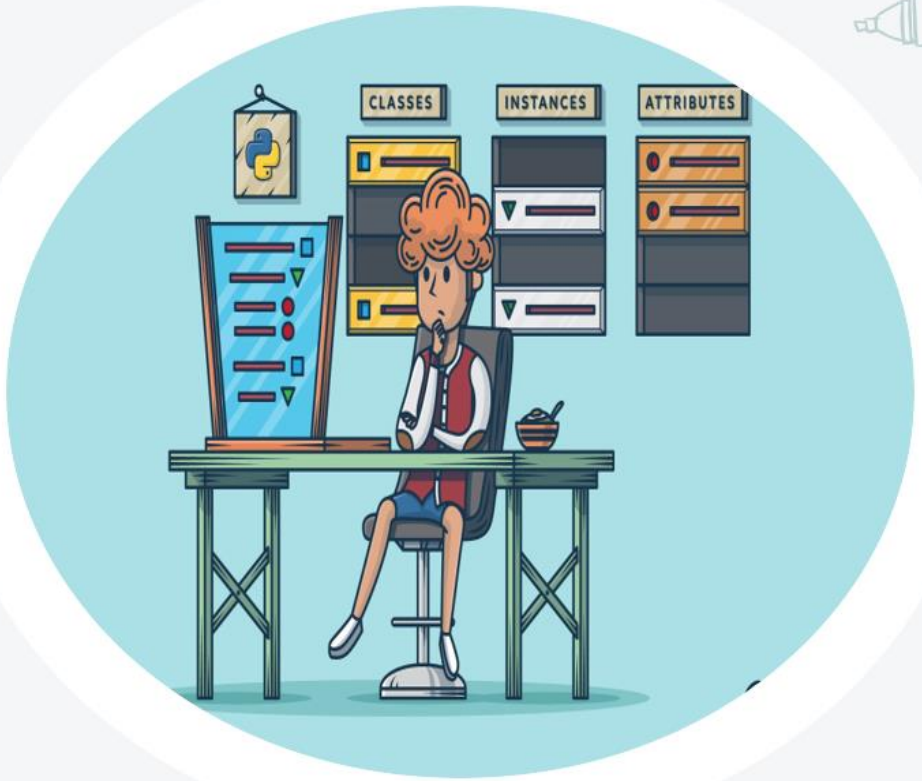


9. Objective Oriented Programming – part1

An introduction to OOP concept and class definition and structure in Python

+

Some clean code notes

So far we have seen...

02

Data Structures

Basic data structures in Python, int, string, list, dict and etc.

04

Control Flow

if elif else , for while. and the concept of Iteration.

01

Introduction to Programming

What programming languages are and Where Python stands among them.

03

Operators

Operators in Python such as logical or comparison operators.

And now...

Object Oriented Programming

The most famous method of programming…

Contents

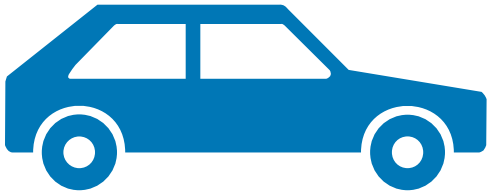
1. An Example to Begin With

Let’s see an example in which we are to code some program, that can

- get and store different information about cars
- do some tests and make some models
- displays it to us
- and it has many other features…



1. An Example to Begin With

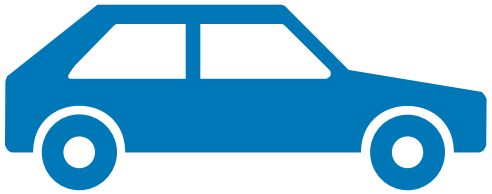


- Each car has different properties and attributes, some are static,
    - type (SUV, Saloon, Sports)
    - width and length
    - price
    - brand
    - production year
    - highest speed
    - …

- Some are dynamic and needs to updated during the program,
    - speed and acceleration
    - location
    - direction
    - …

1. An Example to Begin With

- Also cars have different functionalities too that changes those dynamic attributes,
  - moving forward or backward
  - accelerating or braking
  - changing gear
  - …



Car
model : string
brand : string
price : int
speed : int
move()
change_gear()
accelerate()

1. An Example to Begin With

- The main challenge is still ahead of us…
- How to bundle all these properties and behaviors together and make it as one single property in our program?
- So far we have learnt how to make different variables with different types of data and how to write functions…
- Coding the program in the same style we have learnt in this example has some major problems…



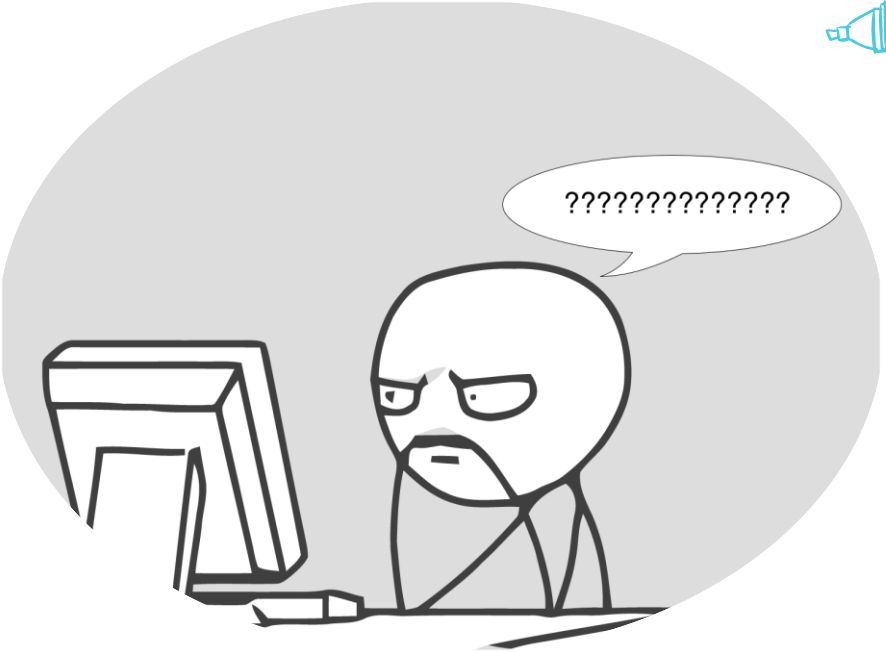
Contents

2. An Introduction to **O**bject **O**riented **P**rogramming (OOP)

What does OOP mean?

- A method of structuring a program by bundling related properties and behaviors into individual objects.
- In another word, OOP is a programming paradigm based on the concept of objects.

Read More



Too complicated? Let's talk simply!

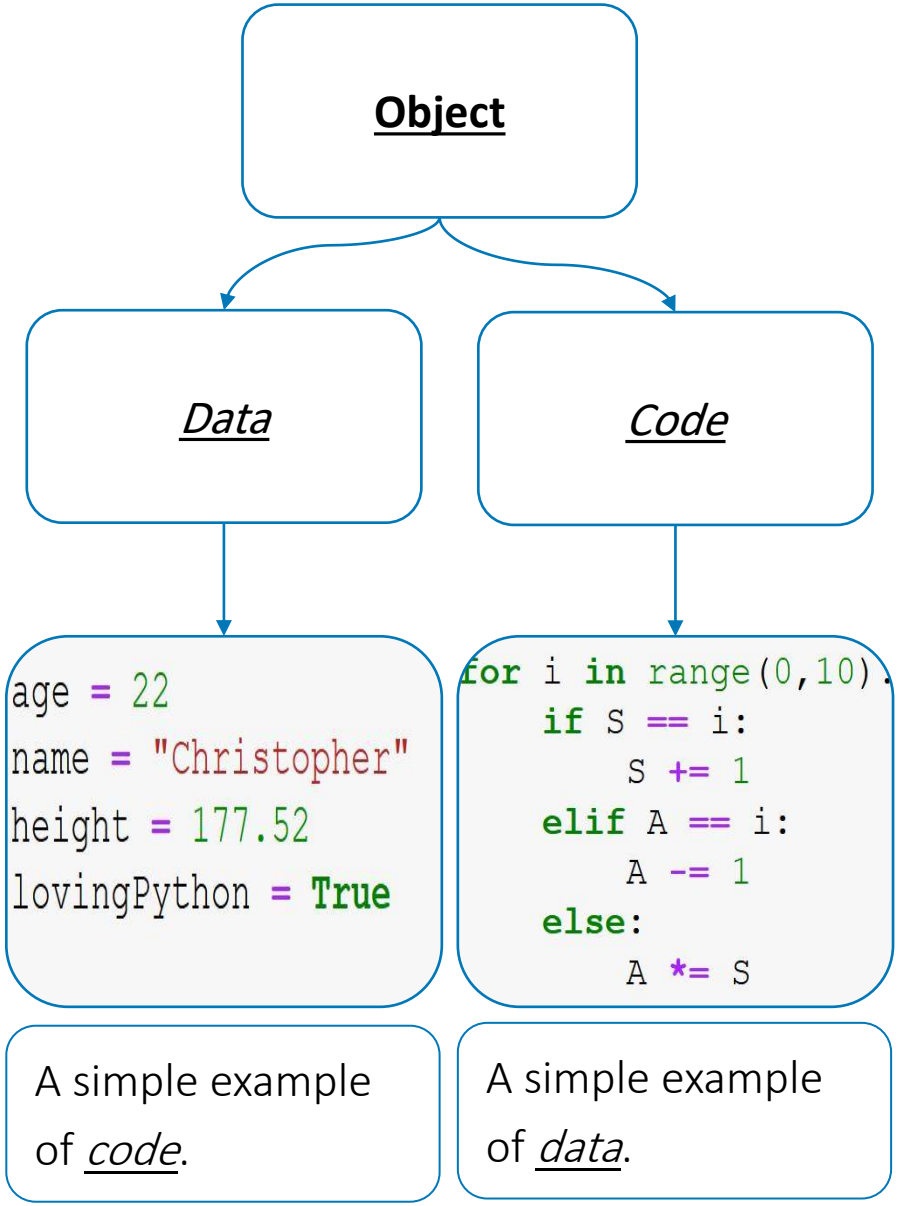
2. An Introduction to OOP

- In OOP, we're trying to break to code to sth called "Objects".

- An object, is sth that can have two elements, *data* and *code*.

- *Data* is an element of the program containing data and values.

- *Code* is an element like for loops or conditional statements, it changes data and controls it or determines the flow of the program.



But have we ever seen any **objects** in Python so far?

—

Ofc we have! Let's see an example of objects we have seen so far

2. An Introduction to OOP

- String, Sets, Lists and Tuples → all objects
- Let's take a look at the example below…

```
In [33]: Food = ["Pizza" , "Pasta"]
         Food.append("Burger")
         Food.append("Sushi")
         Food.append("Pizza")
         Food.append("Hotdog")
         Food.pop()
         Food.pop()
         Food.pop()
         Food.sort()
         Food.reverse()
         Food

Out[33]: ['Pizza', 'Pasta', 'Burger']
```

Data

Code

In this example, we observed that we are capable of both adding *data* and modifying it by *code*, **Because lists are objects.**

An Important Tip!

In OOP,

- this *code* we have talked about is called

“Method”

- And this *data* we have talked about is called

“Field” or “Attribute”

Used in Python

In OOP,

We learn not only how to use built-in objects in Python

and modify them

but also

We learn how to make our own brand new objects with

different attributes and methods…

2. An Introduction to OOP

- There's still this important question left… after all, Why OOP?

- Theoretically speaking → all programs can be made without any objects

- But this is just theoretical → in practice and most specially in big projects
  - We should have very meaningful structures otherwise…
  - Without OOP, it's almost impossible to bundle different attributes and methods.
  - OOP allows us to better model real-life applications to code and the code makes much more sense and it's easier to debug and develop.

But wait to see it on your own… Let's get a little more practical…

Contents

3. Class vs Instance

- What is a Class?
  - Classes are used to create user-defined data structures. Classes have special functions called "methods".
  - Methods define behaviors or actions a class can perform (like .append() in lists).

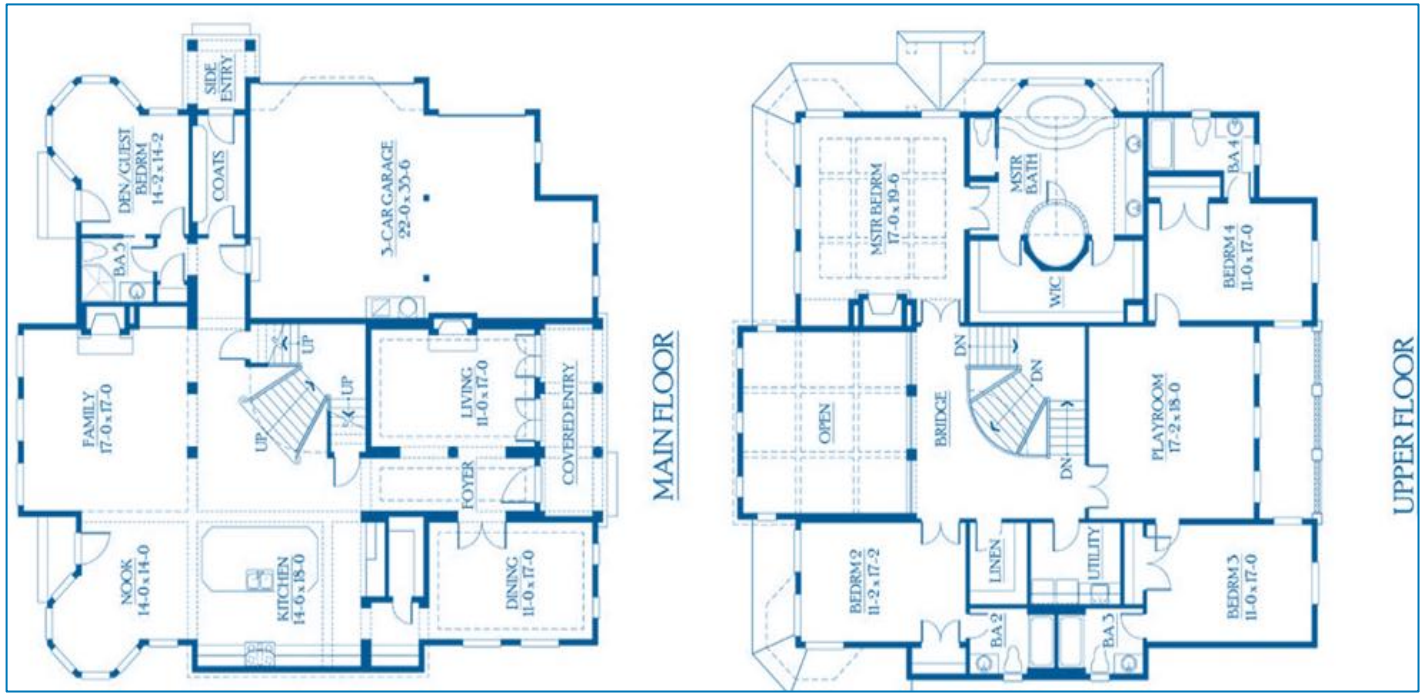
- What is an instance?
  - Building a class → Building the structure of an object.
  - The class →a blueprint.
  - An "instance" → an object built from a class that contains real data.

Let's see an example to better understand this difference

3. Class vs Instance

- The original design (like a class) → on the left.
  - Being unique.
  - Shows structural design.

- The actual building (like an instance) → on the right.
  - Create multiple instances from a class.
  - Instances are actual objects, instances from the same class are different from each other.



Contents

Let’s check out the code now and explore these concepts in Python…

Get in Touch

Keep us informed of your ideas and suggestions! We are most pleased to review them and improve this course in anyway!

+98 939 326 4892

kavehmasoumi@protonmail.com

Thank You

For the attentions