OOP (Object-Oriented Programming)

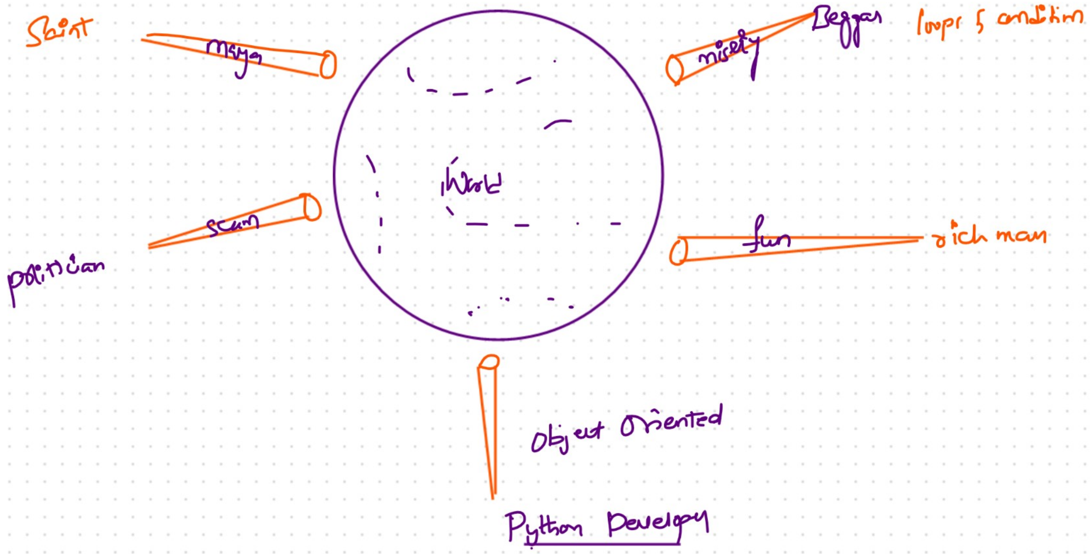
— functional / Structured pogramming

Orientation: persective (the way of looking)

function

express / call()

loops & condition



Rules of O-O

(1, The world is collection of object

(2, every object is useful object , and no object is useless

every object in constant interaction with other object , no object is isolated

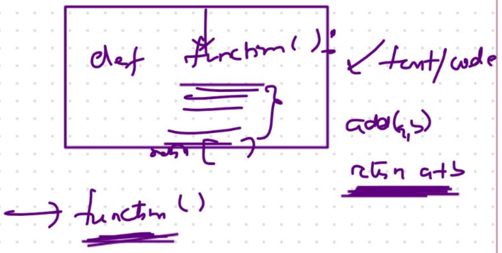
(3, every object is in constant interaction

(4, Every Objed belong to a type [class ⟶ blueprint] that type exists in reality

(5 Type doesn't exists in reality , only the objects of

Class: blue print (or) template for creating object

Object: An instance of a class



((6) Every objed has 2 parts

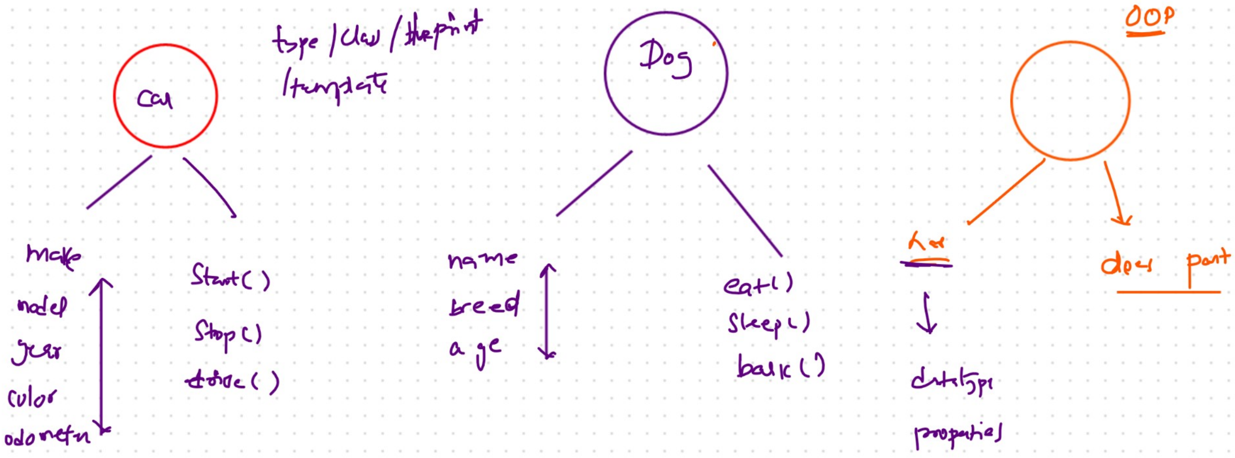
(1 has part (attributes ⟶ data)

(2, does part (methods — functions) / operates on those attributes

(7, Python is a multi-pardigm languge that fully support OOP principles

OOP Features / paradigm

(1) Inheritance
(2, polymorphism (3) encapsulation (4 Abstraction

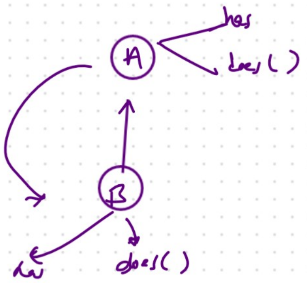


has part : class attributes

instance attributes

Inheritance

allows a class to inherit attributes & methods from another class

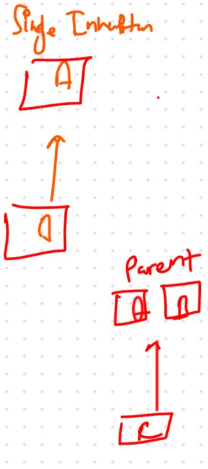


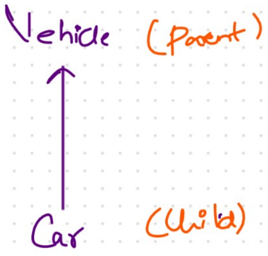
→ single inheritance

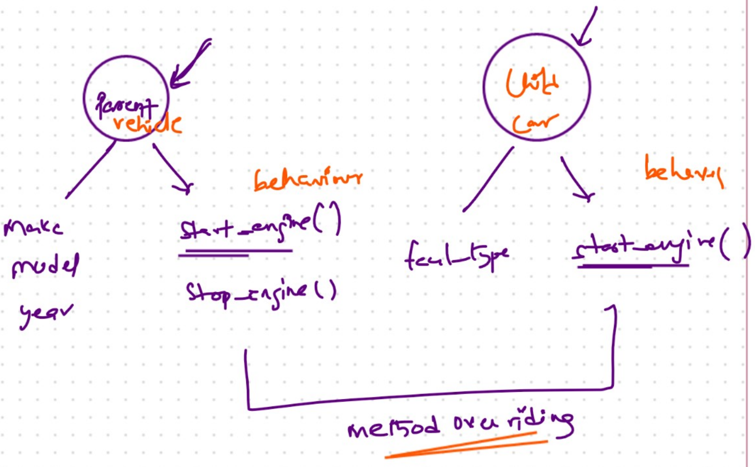
* multiple inheritance

* multi-level inheritance

* Hierarchical Inheritance







MRO (method resolution order)

* helps us the order in which python looks for methods in a hierarchy of class

Python Modules & Packages

Module : modules are python files containg code such(variables, functions, classes) that can be used in other programs

In buit Modules : comes with python as a standard library of buit-in modules

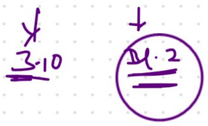
Syntax :

```
(1) import module_name
(2) from module import < >
```

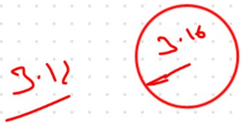
< > → vaiable, functim, class

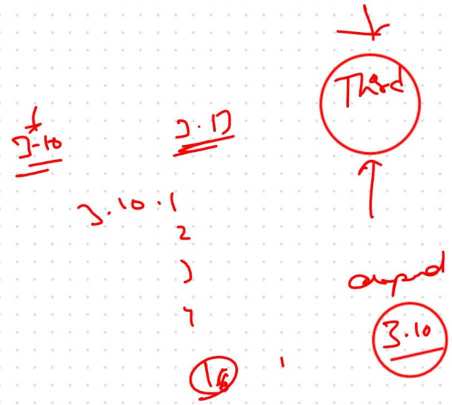
Use pip for third party Packages

```
$ pip install <package_name>    // latest version from release
$ pip install <package_name>==1.2.3
$ pip install --upgrade <package_name>
$ pip uninstall <pkg>
$ pip list
$ pip show <pkg>
```



Virtual Environments :





Creating a virtual environment

```
→ $ python -m venv <name>
                   <myenv>
```

Activate:

```
$ myenv\Scripts\activate (window)
$ source myenv/bin/activate (Unix/Mac)

$ deactivate
```