Python Objected Programming

Mils Burasakorn

fundamental programming paradigm

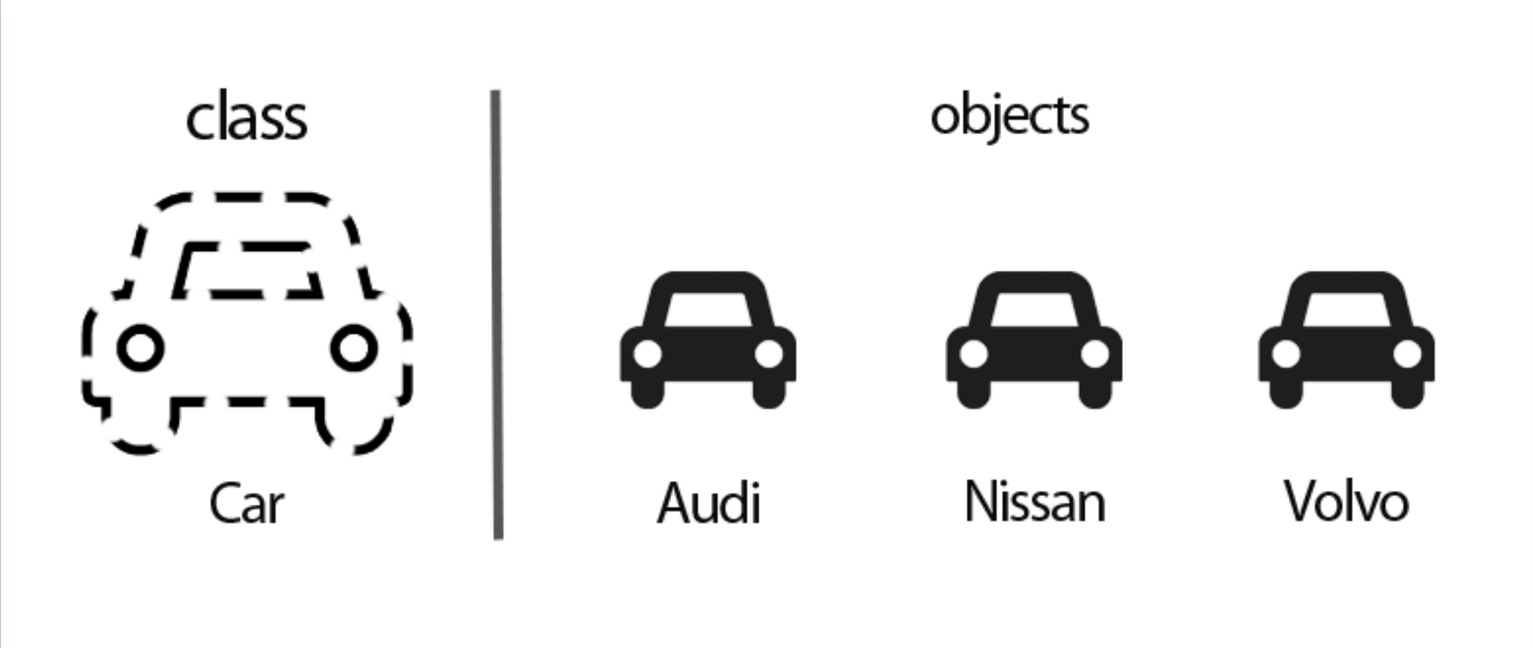
C++, Java, Python, etc.

Object Oriented programming (OOP)

is a programming paradigm that includes or relies on the concept of classes and objects.

https://www.educative.io/blog/object-oriented-programming

It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes) which are used to create individual instances of objects.



https://www.mynotepaper.com/understanding-oop-concepts-oop-class-object

car.py

```python
class Car:
    color = 'red'

audi = Car()
print(audi.color)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
→  bootcamp-python-django git:(master) ✗ python car.py
red
```

https://www.educative.io/blog/object-oriented-programming

Attributes & Methods

```
Attributes:
    - speed
    - size
    - color
    - capacity
    - fuel

Method:
    - increase speed
    - decrease speed
    - brake
    - turn on headlight
```

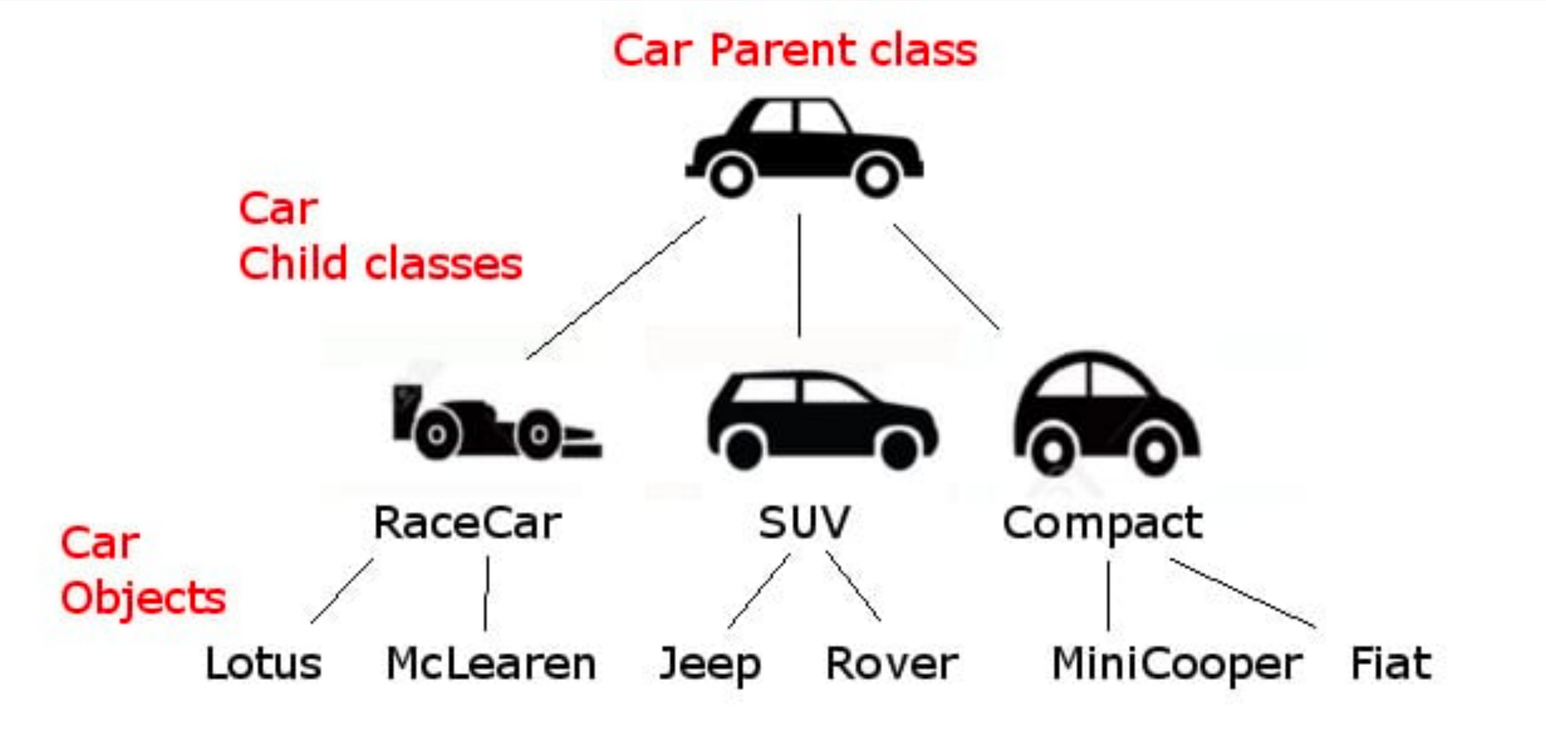
car.py

```python
class Car:
    def __init__(self):
        self.average_speed = 80

    def increase_speed(self):
        self.average_speed += 20
        print(self.average_speed)

    def brake(self):
        self.average_speed = 0

tesla = Car()
tesla.increase_speed()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
→  bootcamp-python-django git:(master) ✗ python car.py
100
```

Inheritance



http://www.danamuise.net/blog/2016/1/12/v12kquly0pw63trxg7bg1kvr8qwguj

inheritance.py

```python
class Car:
    def __init__(self):
        self.average_speed = 80
        self.fuel = 'electric: empty'
        print(self.fuel)

    def increase_speed(self):
        self.average_speed += 20

    def brake(self):
        self.average_speed = 0

class ElectricCar(Car):
    def refuel(self):
        self.fuel = 'electric: full'
        print(self.fuel)


tesla = ElectricCar()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
→  bootcamp-python-django git:(master) ✗ python inheritance.py
electric: empty
electric: full
```

Module & Package

376

A module is a single file (or files) that are imported under one import and used. e.g.

```
import my_module
```

A package is a collection of modules in directories that give a package hierarchy.

```
from my_package.timing.danger.internets import function_of_love
```

1

Documentation for modules

Introduction to packages

share edit follow

answered Oct 30 '11 at 22:55

Jakob Bowyer

27.6k 6 65 87

https://stackoverflow.com/questions/7948494/whats-the-difference-between-a-python-module-and-a-python-package

Module

Module is .py script that can be called in another .py script.

calculator.py

```
oop > calculator.py
1  def sum(num1, num2):
2      result = num1 + num2
3      return result
```

module.py

```
oop > module.py
1  import calculator
2
3  result = calculator.sum(2, 3)
4  print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
→  oop git:(master) ✗ python module.py
5
```

Using Built-in Module

```python
import math

ceil = math.ceil(123.25)
print(f'output: {ceil}')
## output: 124
```

Package

Packages are collection of module.

From X.Y import Z

Module_1.py

Class A:

pass

Class B:

pass

Module_2.py

Class C:

pass

Class D:

pass

Package

__init__.py

Module_1.py

Module_2.py

Packages, modules & imports

Top-level regular package

Subpackage

```
root/
    __init__.py
    first/
        __init__.py
        foo.py
        bar.py
    second/
        __init__.py
        foo.py
        bar.py
```

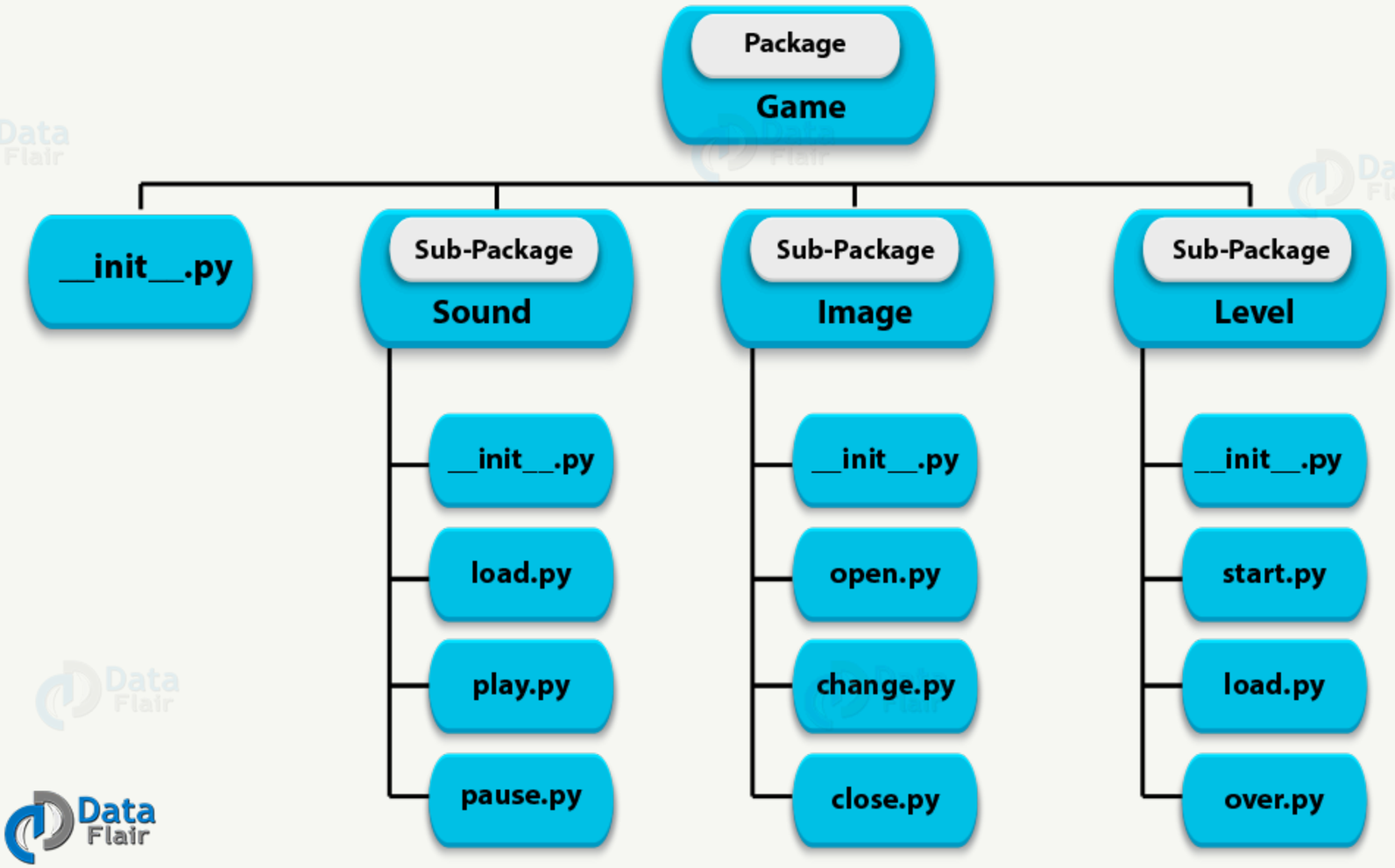
```
# empty, but necessary
```

```
__all__ = ['foo']
```

```
from . import bar
from .. import first
from ..first import foo
```

https://www.slideshare.net/Kevlin/python-advanced-building-on-the-foundation/164-Packages_modules_importsrootinitpyfirstinitpyfoopybarpysecondinitpyfoopybarpyToplevel_regular_packageSubpackage

Package Module Structure



https://data-flair.training/blogs/python-packages/

Advantages of using OOP

- Reusable : Follow DRY concept (Don’t Repeat Yourself)
- Simple/clean
- Easy to maintain