Object Oriented Programming (OOP)

Outline:

- Why OOP.
- What is OOP.
- Classes & Objects.
- Data Hiding & Encapsulation.
- Inheritance.

Why OOP ?!

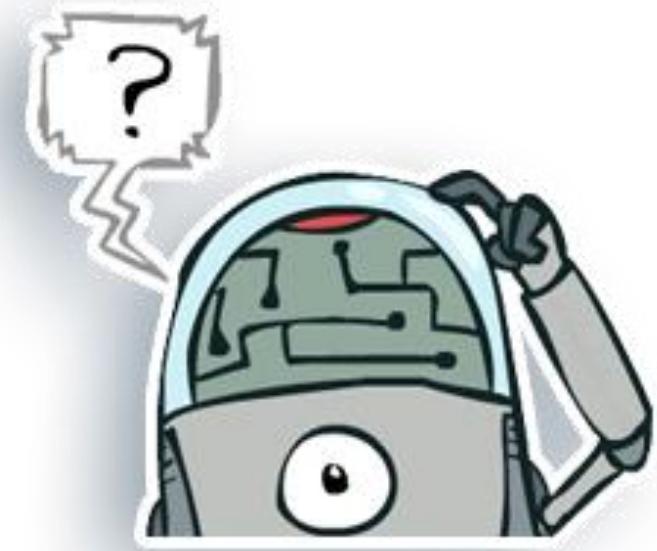
Let’s see Why ^_^

Calculate Sum 10000 times ?!

```
1 x = 10
2 y = 20
3 print(x + y)
4
5 x = 50
6 y = 70
7 print(x + y)
8
9 x = 90
10 y = 200
11 print(x + y)
```

Calculate Sum 10000 times ?!

What if I want to modify code to multiply ?



Calculate Sum 10000 times ?!

What if I want to modify code to multiply ?

```python
x = 10
y = 20
print(x + y)

x = 50
y = 70
print(x + y)

x = 90
y = 200
print(x + y)
```

Calculate Sum 10000 times ?!

What if I want to modify code to multiply ?

```python
def calculate(x, y):
  return x + y # to multiply just make '+' -> '*'

calculate(1, 2) # 3
calculate(10, 20) # 30
calculate(50, 40) # 90
...
```

Make Full Calculator ?

```python
def summ(x, y):
  return x + y

def multiply(x, y):
  return x * y

def subtract(x, y):
  return x - y

def divide(x, y):
  return x / y

summ(1, 2) # 3
multiply(10, 20) # 200
subtract(50, 40) # 10
divide(50, 2) # 25
...
```

Make Lots of Features ?

```python
def summ(x, y):
  pass

def subtract(x, y):
  pass

def read_file(path):
  pass

def write_data_in_file(data, path):
  pass

def get_data_from_internet(url):
  pass

def delete_data_from_database(data):
  pass

...
```

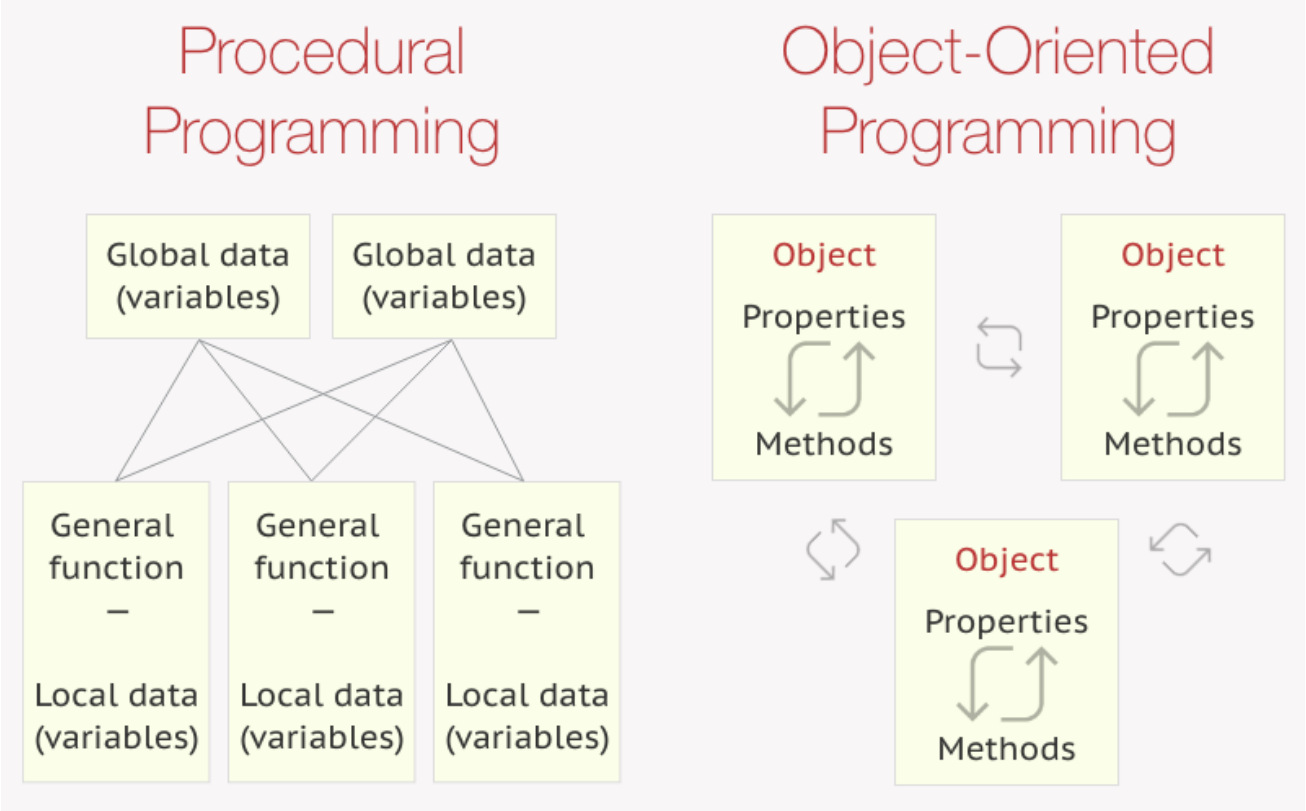
Make Lots of Features ?

```python
def summ(x, y):
  pass

def subtract(x, y):
  pass

def read_file(path):
  pass

def write_data_in_file(da
  pass

def get_data_from_interne
  pass

def delete_data_from_data
  pass

...
```

In Conclusion OOP is important because:

- Easier to troubleshoot code written in OOP style.
- Allows reuse of code.
- Stops you from repeating code (DRY)
- Easier to extend and maintain code written in OOP style.

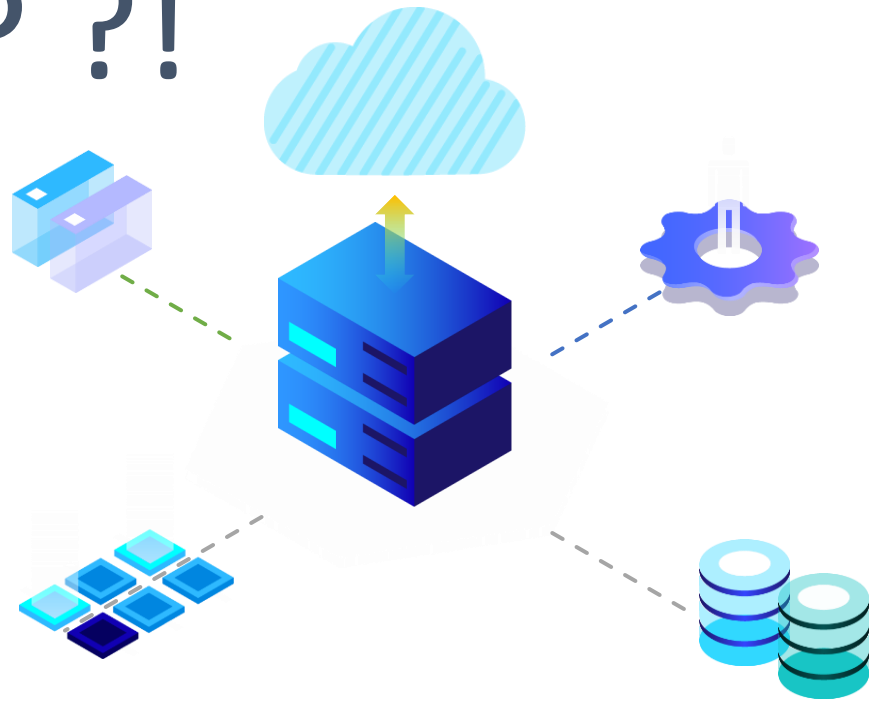
Functional Programming VS OOP:



What is OOP ?!

Everything is an

Object



Dog

Attributes

- Size
- Color
- Breed
- Age
- …

Methods

- Run()
- Park()
- Eat()
- Sleep()
- …

Car

Attributes

- Model
- Color
- Plate Number
- Speed
- …

Methods

- Steer()
- Back()
- Break()
- Throttle()
- …

Person

Attributes

- Name
- Age
- Address
- Gender
- …

Methods

- Eat()
- Sleep()
- Walk()
- Pray()
- …

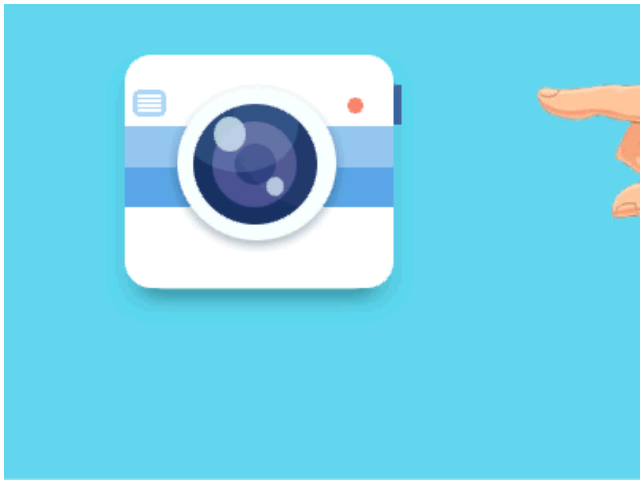
Camera

Attributes

- Lens_width
- Has_Flash
- Depth
- …

Methods

- Take_Photo()
- Take_Video()
- Toggle_Flash()
- Zoom()
- …



In conclusion OOP is:

- Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic.
- An object can be defined as a data field that has unique attributes and behavior.

Which among the following feature is not in the general definition of OOPS?

A) Modularity.

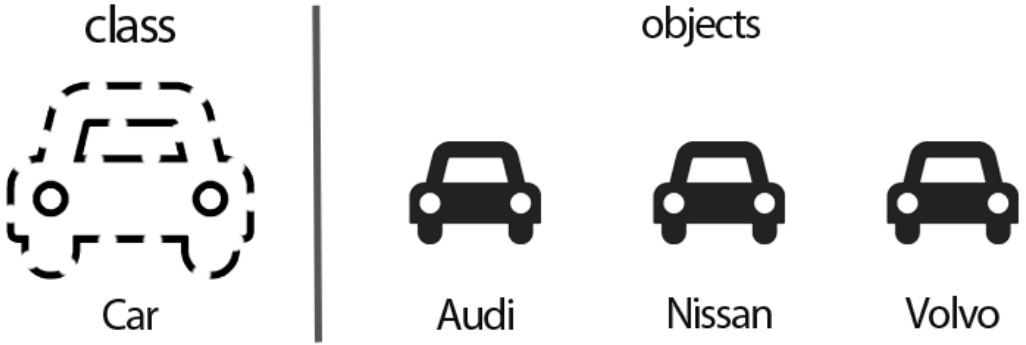
B) Efficient Code.

C) Code reusability.

D) Duplicate or Redundant Data.

Multiple Choice

Classes & Objects



Person

Attributes

- Name
- Age
- Address
- Gender
- …

Methods

- Eat()
- Sleep()
- Walk()
- Pray()
- …

Class: Person

Object 1

- Name: Eslam
- Age: 26
- Gender: Male

- Eat Meat
- Sleep at 9 pm

Object 2

- Name: Ahmed
- Age: 15
- Gender: Male

- Eat Chicken
- Sleep at 12 am

Object 3

- Name: Sara
- Age: 30
- Gender: Female

- Eat Fish
- Sleep at 6 am

```python
class Circle:
    pi = 3.14

    # Circle gets instantiated with a radius (default is 1)
    def __init__(self, radius=1):
        self.radius = radius


    # Method for resetting Radius
    def getArea(self):
        return (self.radius ** 2) * self.pi


    # Method for getting Circumferenc
    def getCircumference(self):
        return self.radius * self.pi * 2


c1 = Circle()
print(c1.radius)                # 1
print(c1.getArea())             # 3.14
print(c1.getCircumference())    # 6.28


c2 = Circle(10)
print(c2.radius)                # 10
print(c2.getArea())             # 314.0
print(c2.getCircumference())    # 62.8
```

OOP in Python ?

Circle

```python
class Person:

    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender


    def greet(self):
        if self.gender == 'male':
          print('Hello, Mr. ' + self.name)
        elif self.gender == 'female':
          print('Hello, Mrs. ' + self.name)


    def is_old(self):
        return (self.age >= 60)


ahmed = Person('ahmed', 20, 'male')
mohammed = Person('mohammed', 67, 'male')
sara = Person('sara', 30, 'female')

ahmed.is_old() # false
mohammed.is_old() # true

ahmed.greet()  # Hello, Mr. ahmed
sara.greet()  # Hello, Mrs. sara

```

OOP in Python ?

Person

In conclusion :

- Class:
  - A class describes the contents of the objects that belong to it: it describes an aggregate of data fields (called instance variables) and defines the operations (called methods).
- Object:
  - An object is an element (or instance) of a class.
  - Objects have the behaviors of their class.
- The object is the actual component of programs.
- while the class specifies how instances are created and how they behave.

Which of the following best defines a class?

A) Parent of an object

B) Instance of an object

C) Blueprint of an object

D) Scope of an object

Data Hiding (Encapsulation)

Data Hiding (Encapsulation)

- It is a way of combining various data members and member functions that operate on those data members (attributes) into a single unit.
- An object's attributes may or may not be visible outside the class definition.
- You need to name attributes with a double underscore prefix, and those attributes then are not be directly visible to outsiders.
- To get access to these attributes we use setters and getters.

Data Hiding (Encapsulation)

```python
class Circle:
    pi = 3.14


    # Circle gets instantiated with a radius (default is 1)
    def __init__(self, radius=1):
        self.__radius = radius

    # Method for resetting Radius
    def getArea(self):
        return self.__radius * self.__radius * self.pi

    # Method for getting Circumference
    def getCircumference(self):
        return self.__radius * self.pi * 2



c1 = Circle(10)

print('Radius is: ',c1.__radius)

"""
AttributeError: 'Circle' object has no attribute '__radius'
"""
```

Private members

Data Hiding (Encapsulation)

Setter and Getter

```python
class Circle:
    pi = 3.14


    # Circle gets instantiated with a radius (default is 1)
    def __init__(self, radius=1):
        if type(radius) == int:
            self.__radius = radius
        else:
            self.__radius = 1


    # Setter
    def set_radius(self, new_radius):
        if type(new_radius) == int:
            self.__radius = new_radius
        else:
            print('this is not an interger')

    # Getter
    def get_radius(self):
        print(f'the radius is: {self.__radius}')
```

What is encapsulation in OOP?

A) It is a way of combining various data members and member functions that operate on those data members into a single unit.

B) It is a way of combining various data members and member functions into a single unit which can operate on any data.

C) It is a way of combining various data members into a single unit.

D) It is a way of combining various member functions into a single unit.

Multiple Choice

Inheritance

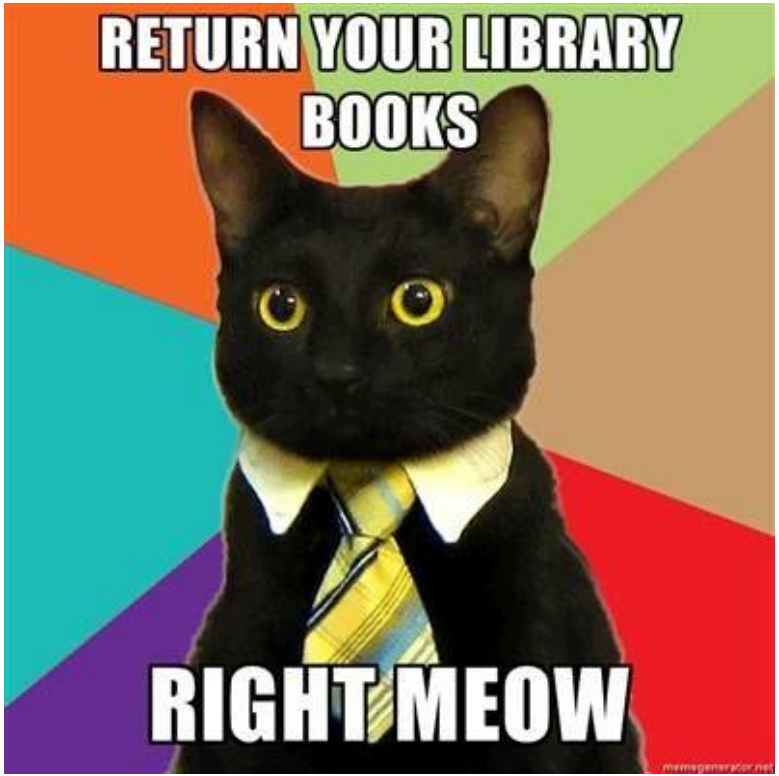
Inheritance

- It is a way to form new classes using classes that have already been defined.
- The newly formed classes are called derived classes, the classes that we derive from are called base classes.

Inheritance

```python
class Animal:

    def __init__(self):
        self.species = 'mammal'
        print("Animal created")

    def whoAmI(self):
        print("Animal")

    def eat(self):
        print("Eating")

# ------------------------------------------------ #

class Dog(Animal):

    def __init__(self):
        Animal.__init__(self)     # call parent __init__
        self.sound = 'High'
        self.love_bones = True
        print("Dog created")

    def bark(self):
        print(f'Woof Woof with {self.sound} Sound')

    def eat(self):
        if self.love_bones:
            print('Love eating bones')
        else:
            print('Love meat')

# ------------------------------------------------ #


sam = Dog()
# Animal created
# Dog created

sam.species      # mammal
sam.love_bones   # True

sam.whoAmI()     # Animal
sam.eat()        # Love eating bones
sam.bark()       # Whao Whao with High Sound
```

Project 4 – Library system



Project 5 – Library system

Questions ?!

Thanks!

```
>_ Live long and prosper
```