

**DAY**

**16**

**RECAP**

DAY  
16

keep calm,  
wear mask  
&  
study hard

A person is working on a laptop, with their hands visible on the keyboard. The laptop is on a wooden desk. In the background, there are some books and a blurred figure of another person. The text 'WORK HARD IN SILENCE. LET SUCCESS MAKE THE NOISE.' is overlaid on the image in white, bold, uppercase letters.

WORK HARD IN SILENCE.  
LET SUCCESS  
MAKE THE  
NOISE.

**Everything in python  
is an object**

# Object Oriented Programming

**classes & objects**

A class acts as a **blueprint** for creating **objects**.

**It is a **template** from  
which similar objects  
are created.**

```
print(type(obj))
```

**A class is a type  
of an object**



Why was OOP introduced ?

**To reduce the  
complexity of code  
in software  
engineering**

**OOP** is a **methodology**  
of programming.

**OOP** is a **technique** of programming

**It is not a language.**

A object is a real world entity having some  
**properties** and **behaviour**.

**Object has two things:**

Attribute  
Behaviour

**Attributes** are the  
**characteristics** of the  
class that help to  
distinguish it from other  
classes.

**Attributes** are the **characteristics** of the class that help to **distinguish** it from other classes.

**Behaviors** are the **tasks** that an object performs.

**Attributes** are the **characteristics** of the class that help to distinguish it from other classes.

**Behaviors** are the **tasks** that an object performs.

A **person's attributes**, for example, include their **age, name, and height**, while their **behaviors** include the fact that a person can **speak, run, walk, and eat**.

# Characteristics of an **object**

Identity (address in memory)

Attributes (properties of the object)

Behaviour (tasks that it can perform)

# Object in programming

An instance of a class which has certain properties and certain methods.

It occupies some memory in computer.



**A class can be used to  
create  
many many objects**

**Different objects will  
be different due to  
different values of  
properties involved.**

Python OOP is different from other languages.

# What is '**self**' in Python ?

**self** is a word in python which is used in the **definition** of each **method** in the class.

**self** means this **current object**

**self** in Python is like **this** in other languages.

**How to create a class in  
Python ?**

**using `class` keyword**

**Demo Example of  
an empty class**

**What is a  
constructor method ?**



It is a **special method** in Python  
that is used for the **creation**  
of an object.

It is the **first method** to be called  
when **creating** an object

How to write constructor in Python ?

```
def __init__(self):
```

**How to write attributes  
in side class ?**

**Inside the constructor  
function by using self  
keyword**

# **Demo Example of** **Student class**

# How to create objects

```
s = Student([parameters])
```

parameters are optional

# **Some other demo examples**

**We will see in next class**

**Day-15**

**Ask Doubts**

*IN THE GROUPS*

**THANK  
YOU**

**INSPIRE PYTHON COURSE**

**dunder methods**



They are the methods starting and  
ending in

          

(double underscore)

ex. `def __init__()`

ex. `def __str__()`

**They are called magic methods**

**Example Demo**

**instance vs class variables**

Instance Variable —

Declared inside the constructor method of class  
(the `__init__` method).

They are tied to the particular object instance of  
the class, hence the contents of an instance  
variable are completely independent from one  
object instance to the other.

## Class Variables —

Declared inside the class definition  
(but outside any of the instance methods).

They are not tied to any particular object of the  
class,

hence shared across all the objects of the class.

Modifying a class variable affects all objects  
instance at the same time.

**Accessing the instance variable name  
is pretty straight forward.**

**However there is a little more flexibility  
when it comes to access the class variable.**

**As above, we can access wheels via  
the object instance or the Class itself.**

**Therefore modifying a class  
variable on the class namespace affects  
all the instances of the class.**

**How to check if an object  
belongs to a class or not**

`isinstance(obj, Class)`



# Destructor in Python

**def \_\_del\_\_(self)**

class Point:

```
def __init__( self, x=0, y=0):self.x = x  
    self.y = y
```

```
def __del__(self):  
    class_name = self.__class__.__name__  
    print class_name, "destroyed"
```

instance vs class vs static  
method

## **Instance method**

**They are most widely used methods.**

**Instance method receives the instance of the class as the first argument, which by convention is called self, and points to the instance of our class**

# What is a class method?

**A class method is a method that is bound to a class rather than its object.**

**It doesn't require creation of a class instance, much like static method.**

It takes **cls** as first argument

It is defined by using **@classmethod** syntax

## Static Methods

A static method is marked with a `@staticmethod` decorator to flag it as static.

It does not receive an implicit first argument (neither `self` nor `cls`).

It can also be put as a method that **“doesn't know its class”**.

Hence a static method is merely attached for convenience to the class object.

Difference between **class** and **static** method

# Day 17

OOPS continued.....

# Demo examples

## Design a **Employee** class



Problem Solving Through Class

**Reverse a string**

Problem No. 1

Problem Solving through Class

# **Reverse a number**

Problem No. 2

**THANK  
YOU**

# Sorting & Searching

# Binary Search Algorithm

**Fast Search Algorithm**

Search for an element in a sorted region

# Bubble Sort Algorithm

Sort a list in ascending order

# **Selection Sort Algorithm**

**THANK**  
**YOU**