

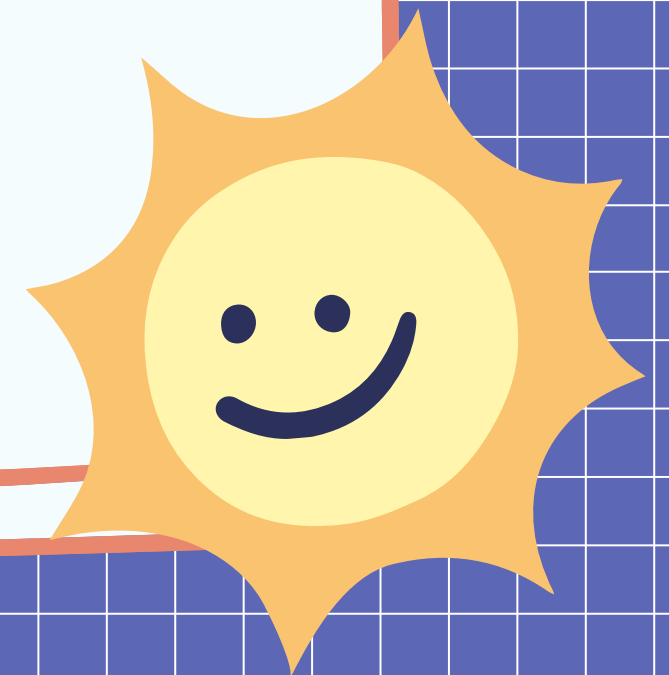
# Object Oriented Programming

## Advanced Python 3

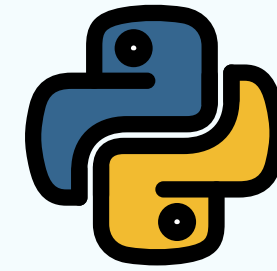
with

**MD. Mohaiminul Islam Imran**

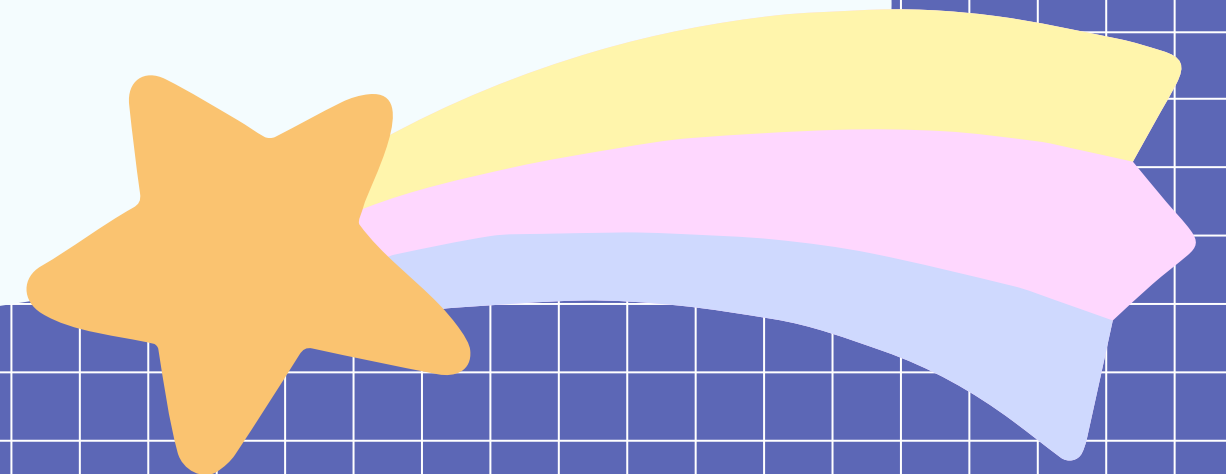
email: [emukhan568@gmail.com](mailto:emukhan568@gmail.com)



# Class



A Python class is a group of attributes  
and methods.



# What is Attribute ?

Attributes are represented by variable that contains data.



# What is Method?

Method performs an action or task. It is similar to function.



# Object Oriented Programming

## Rules(OOP)

- The class name can be any valid identifier.
  - It can't be Python reserved word.
- A valid class name starts with a letter, followed by any number of letter, numbers or underscores.
- A class name generally starts with Capital Letter.



# How to Create Class

```
class Mobile:  
    def __init__(self):  
        self.model = 'realme'  
    def show_model(self):  
        print('Model:', self.model)
```





**class** Classname:

```
def __init__(self):  
    self.variable_name = value  
    self.variable_name = 'value'
```

```
def method_name(self):  
    Body of Method
```

```
def method_name(self, f1, f2):  
    Body of Method
```

Formal Argument



**class** **Classname:**

**def \_\_init\_\_(self, f1, f2):**

**self.variable\_name = value**

**self.variable\_name = 'value'**

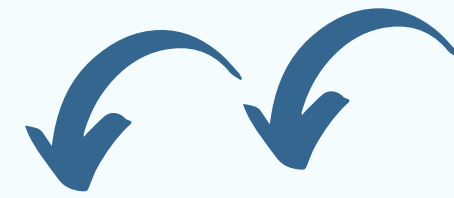
**def method\_name(self):**

**Body of Method**

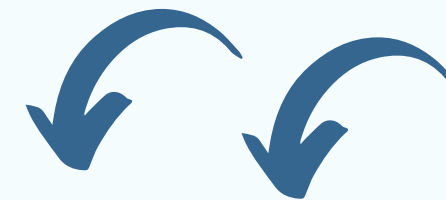
**def method\_name(self, f1, f2):**

**Body of Method**

Formal Argument



Formal Argument





# How to Create Class

```
class Mobile:
    def __init__(self, m):
        self.model = m
    def show_model(self, p):
        price = p                # Local Variable
        print('Model:', self.model, 'Price:',
              price)
```





# Object



Object is class type variable or class instance. To use a class, we should create an object to the class.

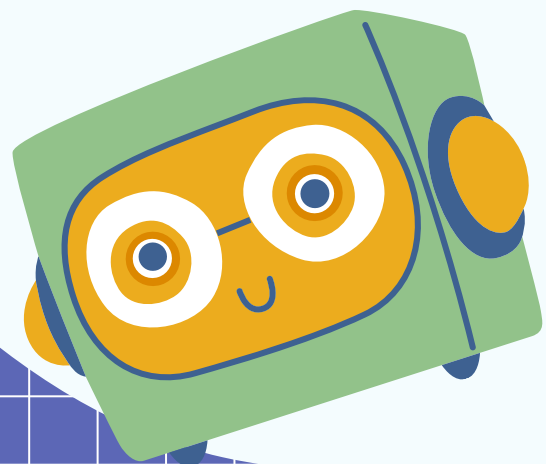
Instance creation represents allotting memory necessary to store the actual data of the variables.

Each time you create an object of a class a copy of each variables defined in the class is created.

In other words you can say that each object of a class has its own copy of data members defined in the class.

Syntax: -

```
object_name= class_name()  
object_name= class_name(arg)
```

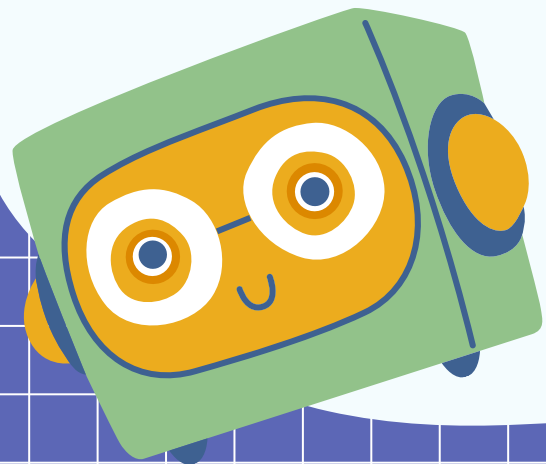


## How to Create Object



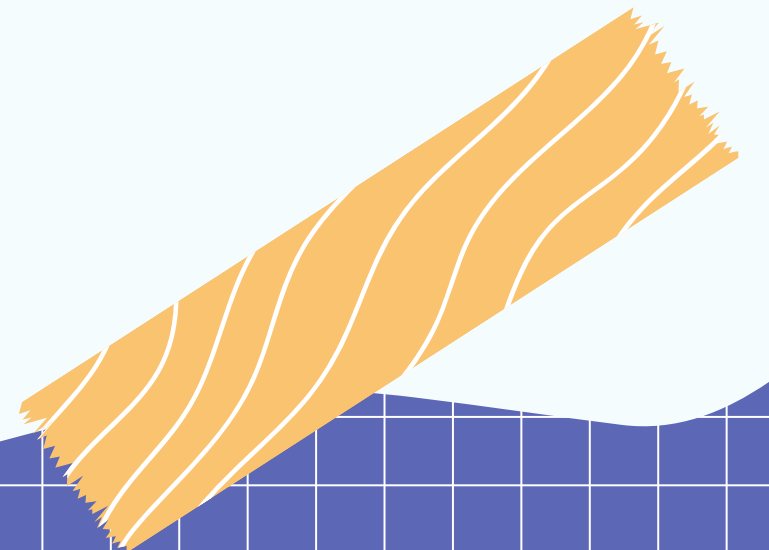
```
class Mobile:
    def __init__(self):
        self.model = 'RealMe X'
    def show_model(self):
        print('Model:',
              self.model)
```

```
realme = Mobile()
```

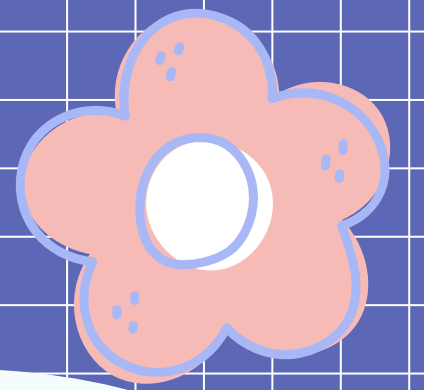


```
class Mobile:
    def __init__(self, m):
        self.model = m
    def show_model(self):
        print('Model:',
              self.model)
```

```
realme= Mobile('RealMe X')
```



# How it works



`realme = Mobile()`

- A block of memory is allocated on heap. The size of allocated memory is to be decided from the attributes and methods available in the class (Mobile).
- After allocating memory block, the special method `__init__()` is called internally. This method stores the initial data into the variables.
- The allocated memory location address of the instance is returned into object (realme).  
The memory location is passed to self.



# Accessing class member using object

We can access variable and method of a class using class object or instance of class.

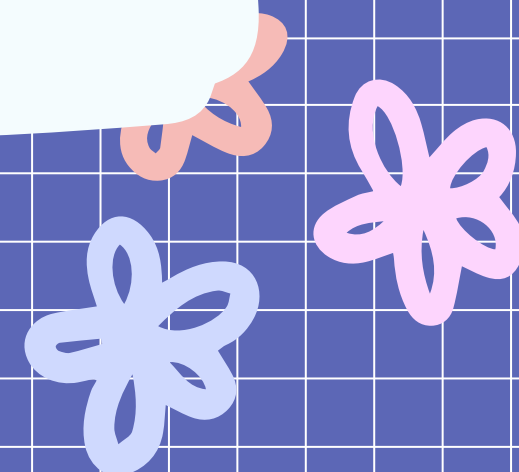
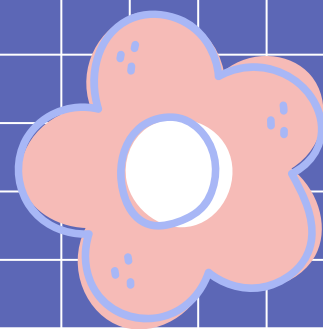
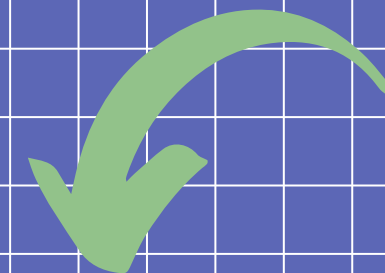
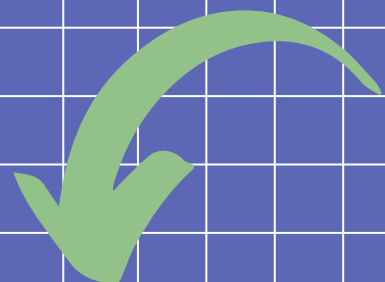
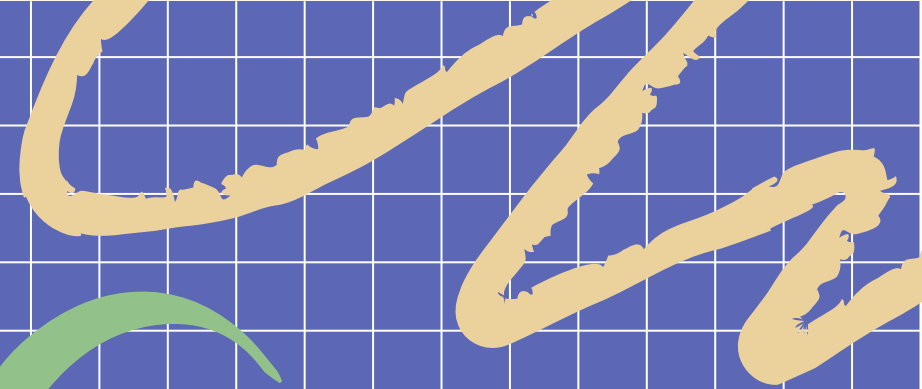
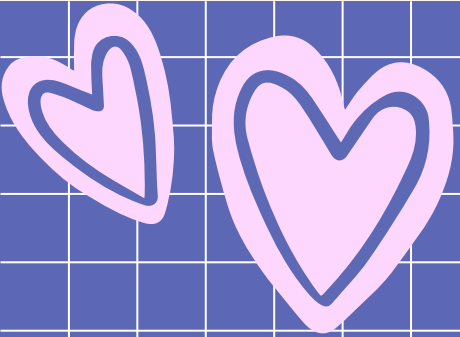
```
object_name.variable_name  
realme.model
```

```
object_name.method_name ( )  
realme.show_model ( );
```

```
object_name.method_name (parameter_list)  
realme.show_model(1000);
```









# self Variable

self is a default variable that contains the memory address of the current object.

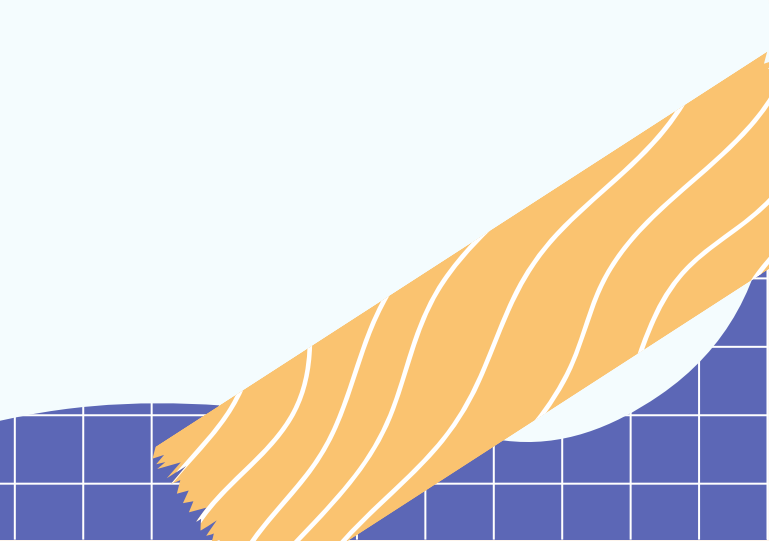

This variable is used to refer all the instance variable and method.

When we create object of a class, the object name contains the memory location of the object.

This memory location is internally passed to self, as self knows the memory address of the object so we can access variable and method of object.

self is the first argument to any object method because the first argument is always the object reference. This is automatic, whether you call it self or not.

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



# Object

Each time you create an object of a class a copy of each variables defined in the class is created.

```
class Mobile:
```

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

```
        self.model = 'RealMe X'
```

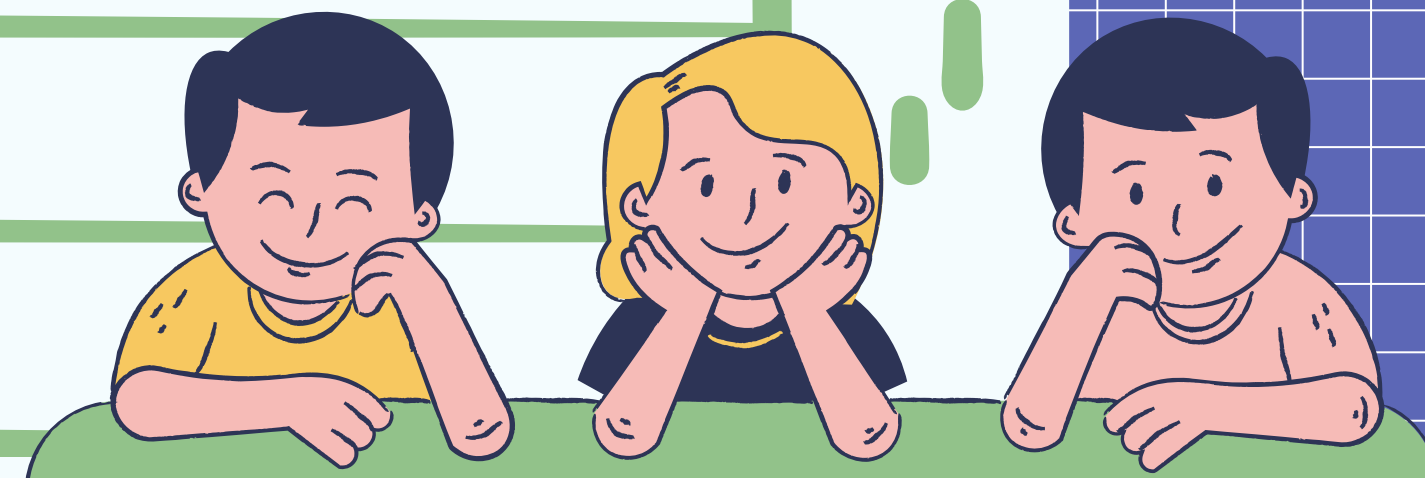
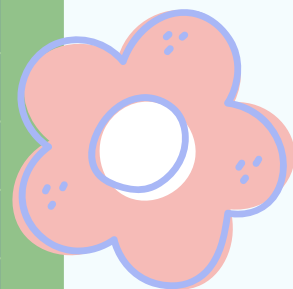
```
    def show_model(self):
```

```
        print('Model:', self.model)
```

```
realme = Mobile()
```

```
redmi = Mobile()
```

```
imran = Mobile()
```



# Thanks

**Md Mohaiminul Islam Imran**



github: [github.com/emumia](https://github.com/emumia)



Twitter: [twitter.com/imran\\_bin\\_MD](https://twitter.com/imran_bin_MD)



Linke-in: [www.linkedin.com/in/muhammad-imran-](https://www.linkedin.com/in/muhammad-imran-014299117/)

014299117/



Youtube:

[www.youtube.com/@code\\_2\\_learn6666](https://www.youtube.com/@code_2_learn6666)

## practice

To succeed in life in any particular field or subject, one need to practice regularly with full commitment and planned strategies.

