

पायथॉन किवर्डस् (Python Keywords):-

True	class	finally	is	return
False	continual	for	lambda	try
None	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

एस्केप सिक्वेन्स (Escape Sequence):-

\n	New Line	print “Hello \n World”
\t	Tab	print “Hello \t World”
\v	Vertical tab	print “Hello \v World”
\r	Carriage Return	print “Hello \r World”
\b	Backspace	print “Hello \b World”
\a	Audio bell	print “Hello \a World”
\\	Single slash	print “Hello \\ World”

Type	Data Type	Example
Numeric Types:	int, ()	x = int(20)
	float, ()	x = float(20.5)
	complex ()	x = complex(1j)
Sequence Types:	list, []	x = list(("apple", "banana", "cherry"))
	tuple, ()	x = tuple(("apple", "banana", "cherry"))
	range ()	x = range(6)
	str ()	x = str("Hello World")
Mapping Type:	dict { }	x = dict(name="John", age=36)
Set Types:	set, { }	x = set(("apple", "banana", "cherry"))
	frozenset ()	x = frozenset(("apple", "banana", "cherry"))
Boolean Type:	bool ()	x = bool(5)

ऑपरेटर्स (Operators):-

1. गणिती ऑपरेटर्स (Arithmetic Operators)	+ - * / % // **
2. रिलेशनल ऑपरेटर्स (Relational Operators)	< > <= >= == !=
3. असाइनमेंट ऑपरेटर्स (Assignment Operators)	= += -= *= /= %= //=
4. लॉजिकल ऑपरेटर्स (Logical Operators)	and or not
5. बिटवाइज ऑपरेटर्स (Bitwise Operators)	& ^ ~ << >>
6. मेंबरशिप ऑपरेटर्स (Membership Operators)	in not in
7. आयडेंटिटी ऑपरेटर्स (Identity Operators)	is is not

गणिती ऑपरेटर्स (Arithmetic Operators):-

Operator	Name		Example
+	Addition	बेरीज	$4+7 == 11$
-	Subtraction	वजाबाकी	$12-5 == 7$
*	Multiplication	गुणाकार	$6 * 6 == 36$
/	Division (Float value)	भागाकार (अपूर्ण अंक)	$12/5 == 2.4$
%	Modulus	बाकी	$10\% 4 == 2$
//	Floor Division (Integer value)	भागाकार (पूर्ण अंक)	$12//5 == 2$
**	Exponent	घातांक	$3 ** 5 == 243$

रिलेशनल ऑपरेटर्स (Relational Operators):-

Operator	Name		Example
>	Greater than	च्या पेक्षा मोठे	$x > y$
<	Less than	च्या पेक्षा लहान	$x < y$
<=	Less than or equal to	च्या पेक्षा लहान किंवा समान	$x <= y$
>=	Greater than or equal to	च्या पेक्षा मोठे किंवा समान	$x >= y$
==	Equal	समान	$x == y$
!=	Not equal	असमान	$x != y$

असाइनमेंट ऑपरेटर्स (Assignment Operators)

Operator	Example		Output
=	$x = 5$	$x = 5$	5
+=	$x += 3$	$x = x + 3$	8
-=	$x -= 3$	$x = x - 3$	2
*=	$x *= 3$	$x = x * 3$	15
/=	$x /= 3$	$x = x / 3$	1.6666666666666667
%=	$x \% = 3$	$x = x \% 3$	2
//=	$x //= 3$	$x = x // 3$	1

लॉजिकल ऑपरेटर्स (Logical Operators):-

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

बिटवाइज ऑपरेटर्स (Bitwise Operators):-

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
 	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

मेंबरशिप ऑपरेटर्स (Membership Operators):-

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

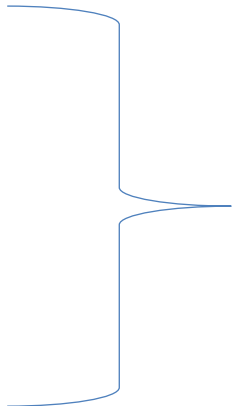
आयडेंटिटी ऑपरेटर्स (Identity Operators):-

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

फिचर्स ऑफ पायथॉन (Features of Python):-

1. **Easy to use** – Due to simple syntax rule
2. **Interpreted language** – Code execution & interpretation line by line
3. **Cross-platform language** – It can run on Windows, Linux and Macintosh etc. equally
4. **Expressive language** – Less code to be written as it itself express the purpose of the code.
5. **Completeness** – Support wide range of library
6. **Free & Open Source** – Can be downloaded freely and source code can be modify for improvement
7. **Dynamically Typed** – This means that the type for a value is checked at the runtime of the code.
8. **Object-Oriented** – Python supports both procedure-oriented and object-oriented programming which is one of the key python features.

टोकन्स ऑफ पायथॉन (Tokens of Python):-

- **K** : keywords
 - **I** : identifiers
 - **L** : literals
 - **O** : operators
 - **P** : punctuators
- 
- # **KILOP**

Types of Arguments in Python Function Definition:-

- | | |
|--------------------------------|---------------------------------------|
| डिफॉल्ट अर्गुमेंट्स | :- (default arguments) |
| किवर्ड अर्गुमेंट्स | :- (keyword arguments) |
| पोझिशनल अर्गुमेंट्स | :- (positional arguments) |
| पोझिशनल आणि किवर्ड अर्गुमेंट्स | :- (positional and keyword arguments) |
| आर्बिट्ररी पोझिशनल अर्गुमेंट्स | :- (arbitrary positional arguments) |
| आर्बिट्ररी किवर्ड अर्गुमेंट्स | :- (arbitrary keyword arguments) |

डिफॉल्ट अर्गुमेंट्स (default arguments):-

This function can be called in 3 ways

```
def Friends(my_name, your_name="Jerry"):
    print(f"{my_name} & {your_name}")
```

`Friends("Tom")`

Output: Tom & Jerry

```
def add(a, b=5, c=10):
```

```
    return a + b + c
```

The assignment operator = is used to assign a default value to the argument.

<code>print(add(3))</code>	<code># a=3</code>	<i>Output 18 => Giving only the mandatory argument.</i>
----------------------------	--------------------	--

<code>print(add(3, 4))</code>	<code># a=3 b=4</code>	<i>Output 17 => Giving one of the optional arguments.</i>
-------------------------------	------------------------	--

<code>print(add(2, 3, 4))</code>	<code># a=2 b=3 c=4</code>	<i>Output 9 => Giving all the arguments.</i>
----------------------------------	----------------------------	---

किवर्ड अर्गुमेंट्स (keyword arguments):-

kword = value

```
def Friends(my_name, your_name):  
    print(f"{my_name} & {your_name}")
```

```
Friends(your_name="Tom", my_name="Jerry")
```

Output: Tom & Jerry

```
def add(a, b=5, c=10):
```

```
    return a + b + c
```

Calling the function add by giving keyword arguments

```
print(add(b=10, c=15, a=20))    # b=10, c=15, a=20 Output 45 => no need to maintain the same  
parameters order.
```

```
print(add(a=10))                # a=10 b=5 c=10    Output 25 => Optional default arguments are  
skipped.
```

पोज़िशनल अर्गुमेंट्स (positional arguments):-

Order of parameters

```
def Friends(my_name, your_name):  
    print(f"{my_name} & {your_name}")
```

```
Friends("Tom", "Jerry")
```

Output: Tom & Jerry

```
def add(a, b, c):  
    return (a + b + c)
```

Keyword arguments should follow positional arguments only

```
print(add(10, 20, 30))  # a=10 b=20 c=30  Output 60 => all arguments are given as positional
```

पोज़िशनल आणि किवर्ड अर्गुमेंट्स (positional and keyword arguments):-

```
def Friends(my_name, your_name):  
    print(f'{my_name} & {your_name}')
```

```
Friends("Tom", your_name="Jerry")
```

Output: Tom & Jerry

```
def add(a, b, c):  
    return (a + b + c)
```

Calling the function add by giving keyword arguments

```
print(add(10, 20, c=30))    # a=10,          Output 60 => Giving mix of positional and  
keyword arguments
```

```
print(add(10, c=30, b=20))  # a=10 b=20 c=30    Output 60 => Giving mix of positional and  
keyword arguments
```

आर्बिट्ररी पोज़िशनल अर्गुमेंट्स

(arbitrary positional arguments):-

For arbitrary positional argument, an **asterisk (*)** is placed before a parameter in function

```
def add(*b):
```

```
    result = 0
```

```
    for i in b:
```

```
        result = result + i
```

```
    return result
```

```
print(add(1, 2, 3, 4, 5))
```

Output: 15

```
print(add(10, 20))
```

Output: 30

आर्बिट्ररी किवर्ड अर्गुमेंट्स

(arbitrary positional arguments):-

For arbitrary positional argument, a **double asterisk (**)** is placed before a parameter in a function

```
def fname(**a):
```

```
    for i in a.items( ):
```

```
        print ( i )
```

```
fname(numbers=5, colors="blue",
```

```
fruits="apple")
```

Output:

```
('numbers', 5)
```

```
('colors', 'blue')
```

```
('fruits', 'apple')
```

Object Oriented Programming (OOP)

For short information:

1. C ++
2. Java
3. Python
4. C Sharp
5. PHP
6. Type Script

Class = Characteristics (variable) + Behavior (method/function)

Class Variable = A class variable is a variable that is declared inside of class, but outside `__init__()` method.

Class Variable = Declared inside the constructor method of class (the `__init__` method)

Method = `def method name(self):`

Object = `obj = Class Name`

Object Oriented Paradigm:-

Encapsulation - It describes the idea of wrapping data and the methods that work on data within one unit.

Abstraction – Hiding something outside world.

Polymorphism – Signal name multiple behavior.

Inheritance – Reusability.

Class

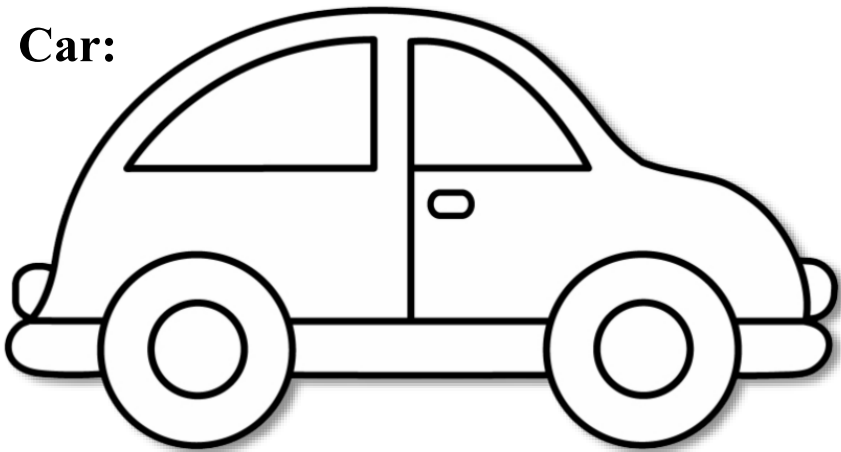
(Pre – Poduction)
Blueprint

What it has?

- Color
- Speed
- CNG
- Type(Ferrari, McLaren)

Instance variable

Car:



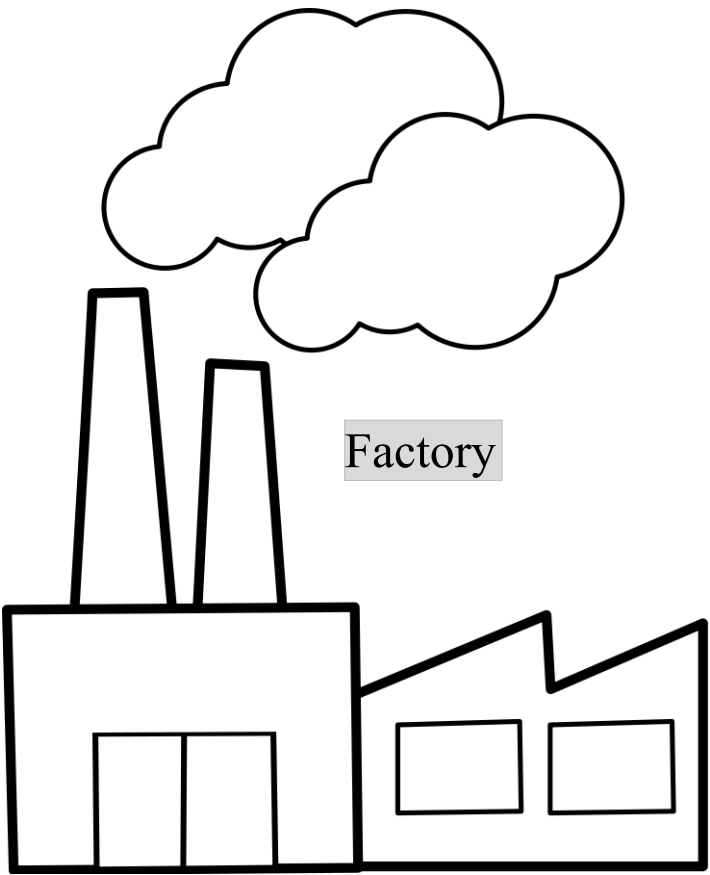
What is does?

- Accelerate
- Breaking
- Starting
- Stopping

Method

Constructor

(Poduction)
Sequence of Methods



Objects

(Output)
**Can create many
objects from a class**



Class as Car
Instance Variable Color, Speed, Type

```
# Class Variable Wheels
# Class Methods as Accelerate, Brake
# Obj as Ferrari

class Car:
    Wheels = 4

    def __init__(self):
        self.Color = str(input("Car Color: "))
        self.Speed = float(input("Car Speed: "))
        self.Type = str(input("Car Name: "))

    def Display(self):
        print("Car Details:\nColor is: {0}\nSpeed is: {1}\nBrand is: {2}".format(self.Color, self.Speed, self.Type))

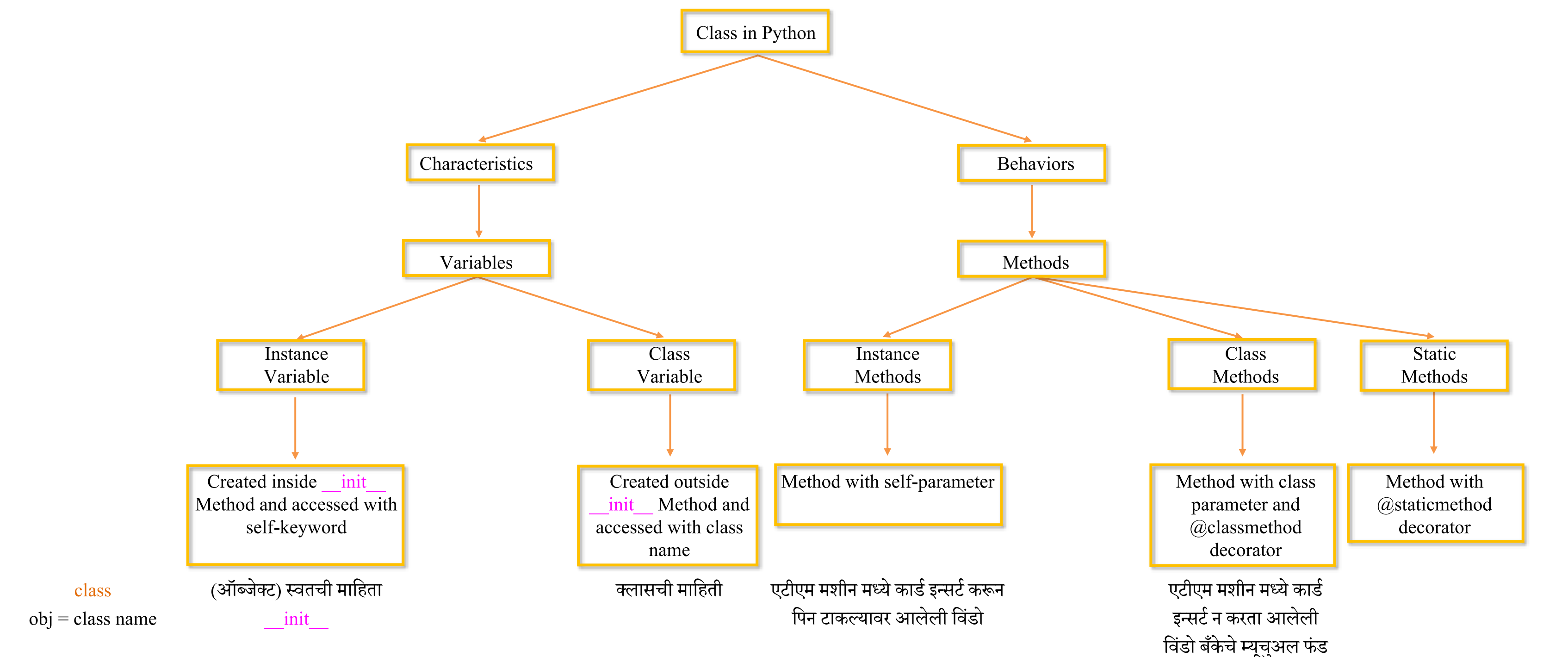
    def Accelerate(self):
        print("High Speed:", self.Speed + 20)

    def Brake(self):
        print("Brake@:", self.Speed - 150)

def main():
    Ferrari = Car()

    Ferrari.Display()
    Ferrari.Accelerate()
    Ferrari.Brake()

if __name__ == "__main__":
    main()
```

<code>class SBI:</code> <code>CardNumber = SBI()</code>	<code>self.name = name</code> <code>self.amount = amount</code> <code>print(CardNumber.name)</code> <code>print(CardNumber.amount)</code>		<code>transactionLimit = 5</code> <code>print(SBI. transactionLimit)</code>	<code>def miniStatement (self):</code> <code>def balance (self):</code> <code>CardNumber.miniStatement()</code> <code>CardNumber.balance()</code>		<code>@class method</code> <code>SBI.MutualFund</code> <code>SBI.GoldLoan</code>	<code>@staticmethod</code>
<code>class Car:</code> <code>Fortuner =Car(White)</code> <code>Verna =Car(Black)</code>	<code>self.color = color</code> <code>print(Fortuner.color)</code> <code>print(Varna.color)</code>	<code>__(init__)</code> <code>Car.car_count +=1</code>	<code>car_count = 0</code> <code>print(Car.car_count)</code>	<code>def accelerate (self):</code> <code>print(Fortuner. accelerate)</code> <code>print(Varna. accelerate)</code>		<code>@class method</code> <code>Car.Fortuner</code> <code>Car.Varna</code>	<code>@staticmethod</code>

```
# Instance variable : Name, Amount, Address, AccountNo
# Instance method : CreateAccount(), DisplayAccountInfo()
# Class variable : Bank_Name ,ROI_On_FD
# Class method : DisplayBankInformation
# Static method : DisplayKYCInfo
```

```
class Bank_Account:
    Bank_Name = "HDFC bank PVT LTD"
    ROI_On_FD = 6.7

    def __init__(self):
        self.Name = ""
        self.AccountNo = 0

    def CreateAccount(self):
        self.Name = input("Enter your name: ")
        self.AccountNo = int(input("Enter your Account Number: "))
    def DisplayAccountInfo(self):
        print("Name of Account Holder : ", self.Name)
        print("Account Number : ", self.AccountNo)
    @classmethod
    def DisplayBankInformation(cls):
        print("Name of our bank is : ", cls.Bank_Name)
        print("Rate of interest we offer on fixed deposit is : ", cls.ROI_On_FD)
    @staticmethod
    def DisplayKYCInfo():
        print("Please consider below KYC information")
        print("According to the rules of Government of India you have to submit below documents")

def main():
    Bank_Account.DisplayKYCInfo()
    print("Name of bank : ", Bank_Account.Bank_Name)
    print("Rate of Interest on Fixed deposit : ", Bank_Account.ROI_On_FD)
    Bank_Account.DisplayBankInformation()
    User1 = Bank_Account()
    User1.CreateAccount()
    User1.DisplayAccountInfo()

if __name__ == "__main__":
    main()
```