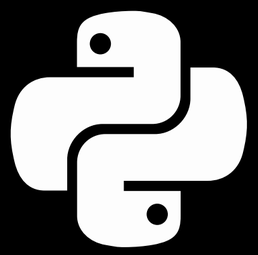
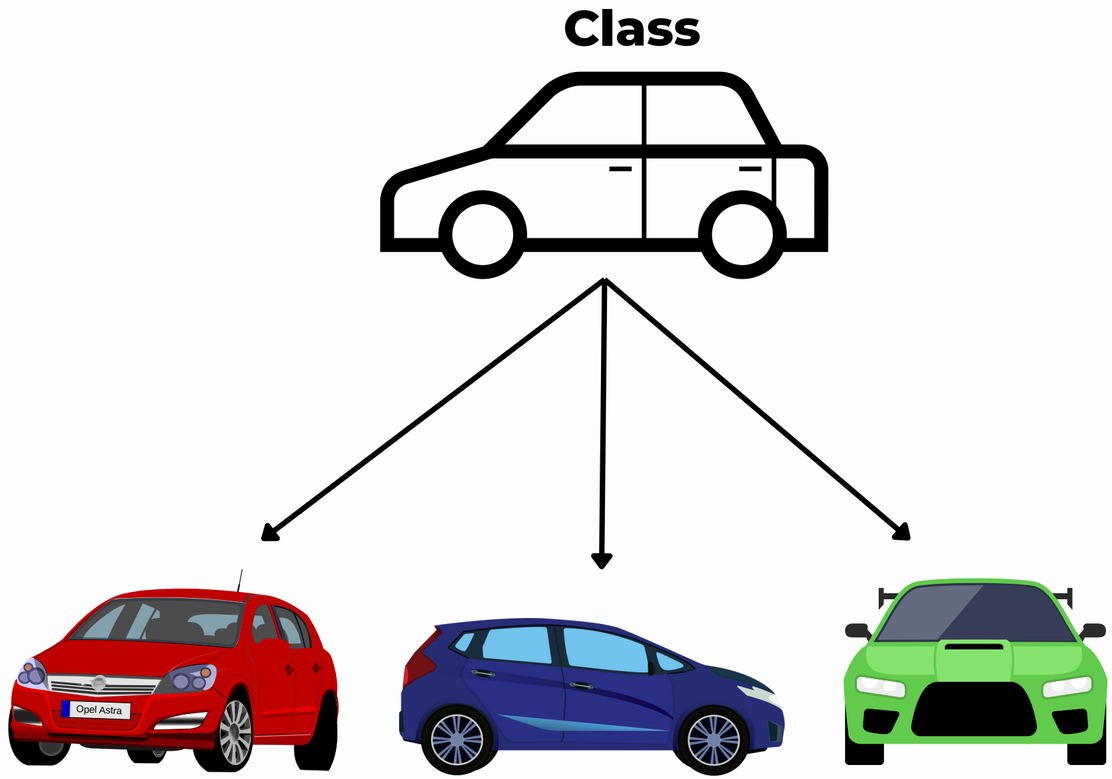


Python
Class and object

Class and Object



- **Red**
- **Ford**
- **Mustang**

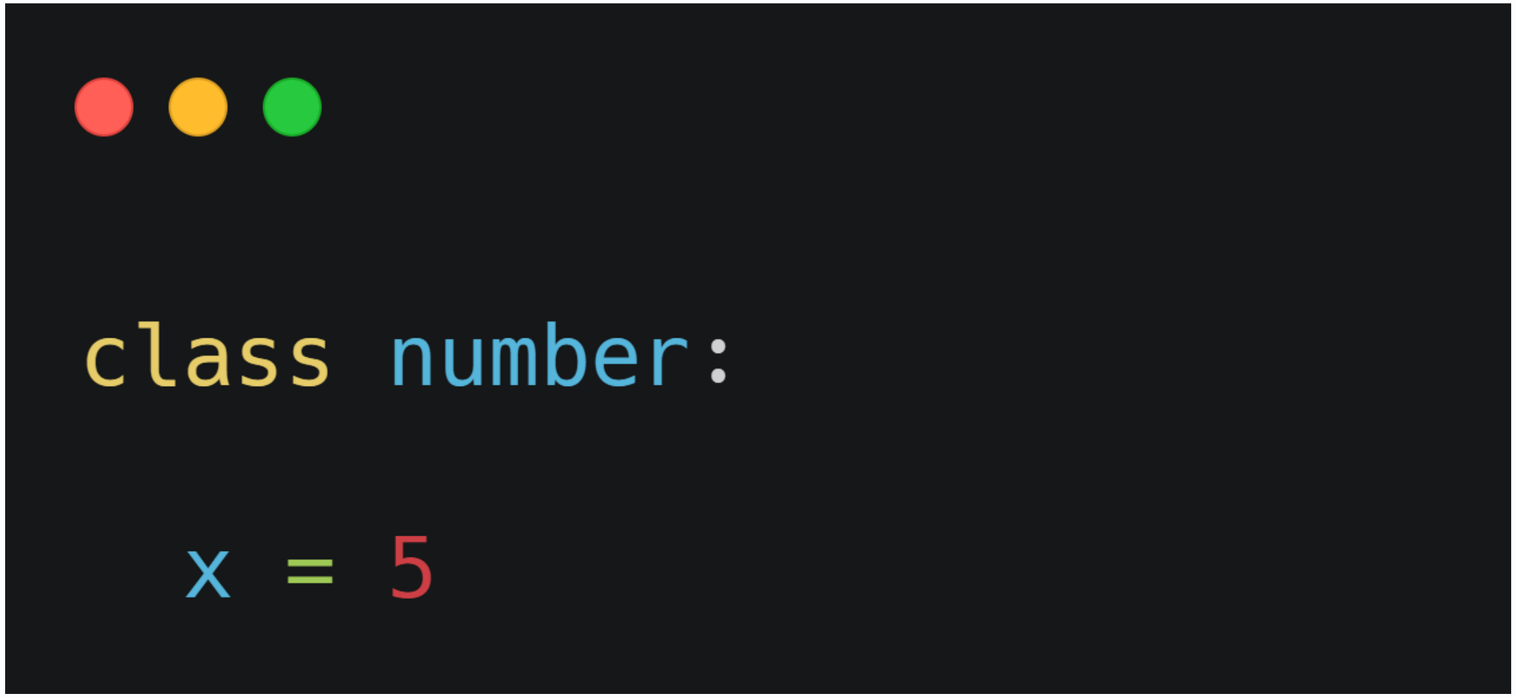
- **Blue**
- **Toyota**
- **primus**

- **Green**
- **Volkswagen**
- **Golf**

- Python is an object oriented programming language, almost everything in python is object, with its properties and methods

- A class is like object constructor, or a “blueprint” for creating objects

Create a class



Create object

```
class number:

   x = 5

obj = number()

print(obj.x)
```

Function(Method) inside class

```
class course:

  def session(self):

    print("This is object and class session")



obj = course()

obj.session()
```

Constructors

```
class myclass:

def __init__(self)

        body of the constructor
```

- Constructor is used for instantiating an object.
- The task of the constructor is to assign values to the data members of the class
- In python the __init__() method is called the constructor

Example - Class

```python
class person:
  def __init__(self, name, age):
    self.name = name
    self.age = age
  def one(self):
    print(f"My name is {self.name},my age is {self.age}")
  def two(self):
    print(f"I am {self.name}, and i am {self.age} years old")
  def three(self):
    print(f"Hi, i am {self.name}, {self.age} yrs")
```

Example - Object

```
person1 = person("Manish",20)

person2 = person("Tom", 23)

person3 = person("sam", 25)


person1.two()

person2.three()
```

Types of variables

1. Instance variables
2. Class variables

Instance variables

```
class car:
  def __init__(self):
    self.milage = 20
    self.company = "BMW"

one = car()
two = car()
print(one.milage, one.company)
print(two.milage, two.company)
```

```
two.milage = 30

print(one.milage, one.company)

print(two.milage, two.company)
```

Class variables

```
class car:

  wheel = 4

  def __init__(self):

    self.milage = 20

    self.company = "BMW"


one = car()

two = car()


print(one.wheel)

print(two.wheel)
```

```
car.wheel = 8


print(one.wheel)


print(two.wheel)
```

Types of methods

1. Instance method
2. Class method
3. Static method

Instance method

```
class student:

  school = "abc school"

  def __init__(self, name, id):

    self.name = name

    self.id = id

  def details(self):

    return f"student name: {self.name}, student id: {self.id}"



obj = student("john",12)

print(obj.details())
```

Class method

```python
class student:
  school = "abc school"
  def __init__(self, name,id):
    self.name = name
    self.id = id

  def details(self):
    return f"student name: {self.name}, age:{self.id}"
  @classmethod
  def schoolname(cls):
    return cls.school

obj = student("John",14)
print(obj.details())
print(student.schoolname())
```

Static method

```python
class student:

  school = "abc school"

  def __init__(self, name,id):

    self.name = name

    self.id = id


  def details(self):

    return f"student name: {self.name}, age:{self.id}"

  @staticmethod

  def info():

   return ("This is information about students")


obj = student("John",14)

print(student.info())
```

Inheritance

```
class one:
  def first(self):
    return "this is first"
  def second(self):
    return "this is second"


class two(one):
  def third(self):
    return "this is third"
  def fourth(self):
    return "this is fourth"


obj2 = two()
```

Operator overloading

```
class point:

  def __init__(self, x,y):

    self.x=x

    self.y=y

p1 = point(2,3)

p2 = point(3,4)

p3 = p1+p2

print(p3)
```

Operator overloading

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

  def __add__(self, other):
    p1 = self.x + other.x
    p2 = self.y + other.y
    return p1 , p2

p1 = p(2,3)
p2 = p(3,4)
p3 = p1+p2
print(p3)
```