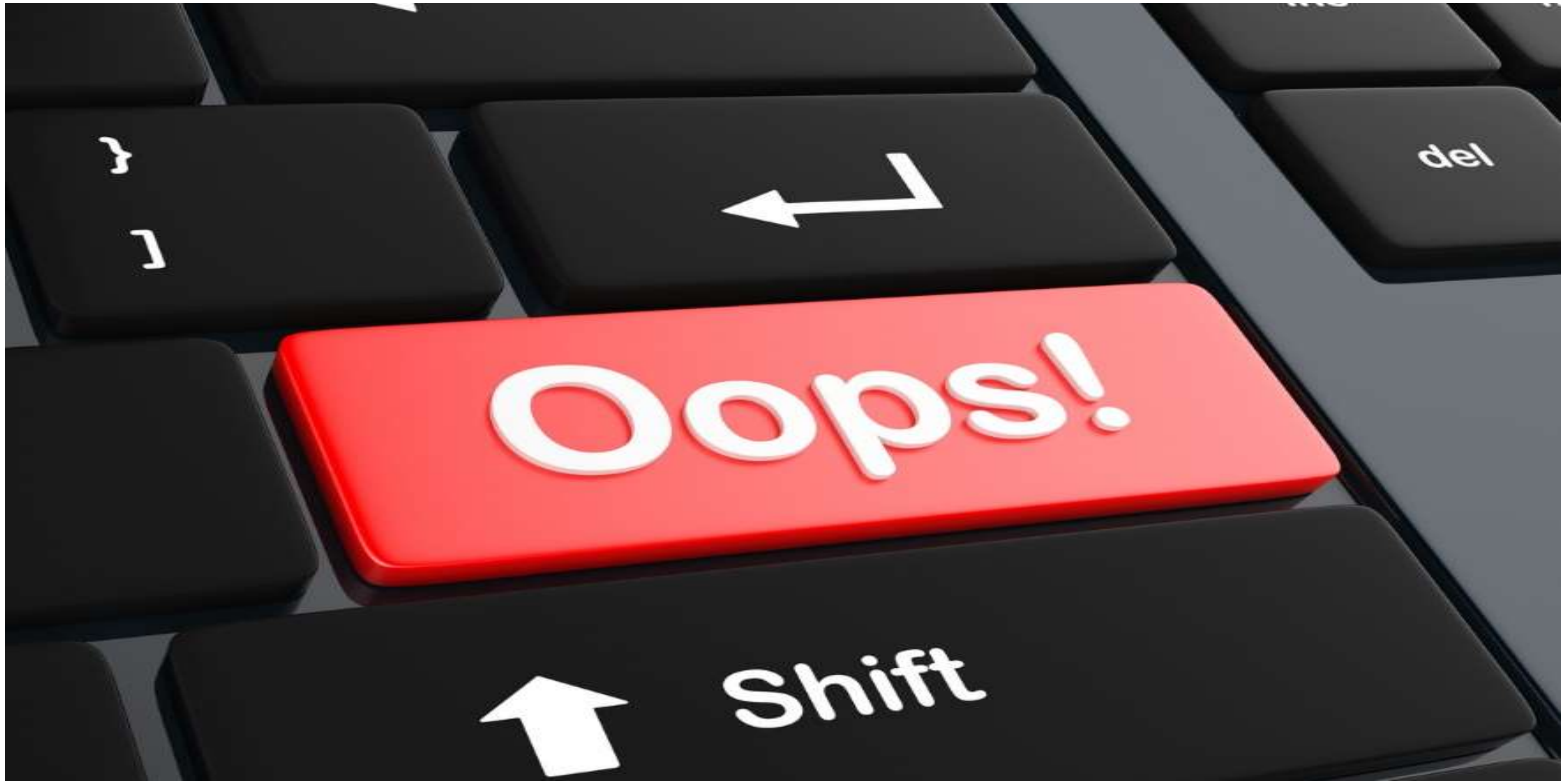
FACE



FACE

Why OOPs?

- Provides a clear structure
- Easy to map the real world problems
- Easily maintain and modify the existing code

Python OOPs

- Multi-paradigm programming language
- OOPs - develop applications using object oriented approach
- Focuses on creating reusable code
- Easily solved by creating object

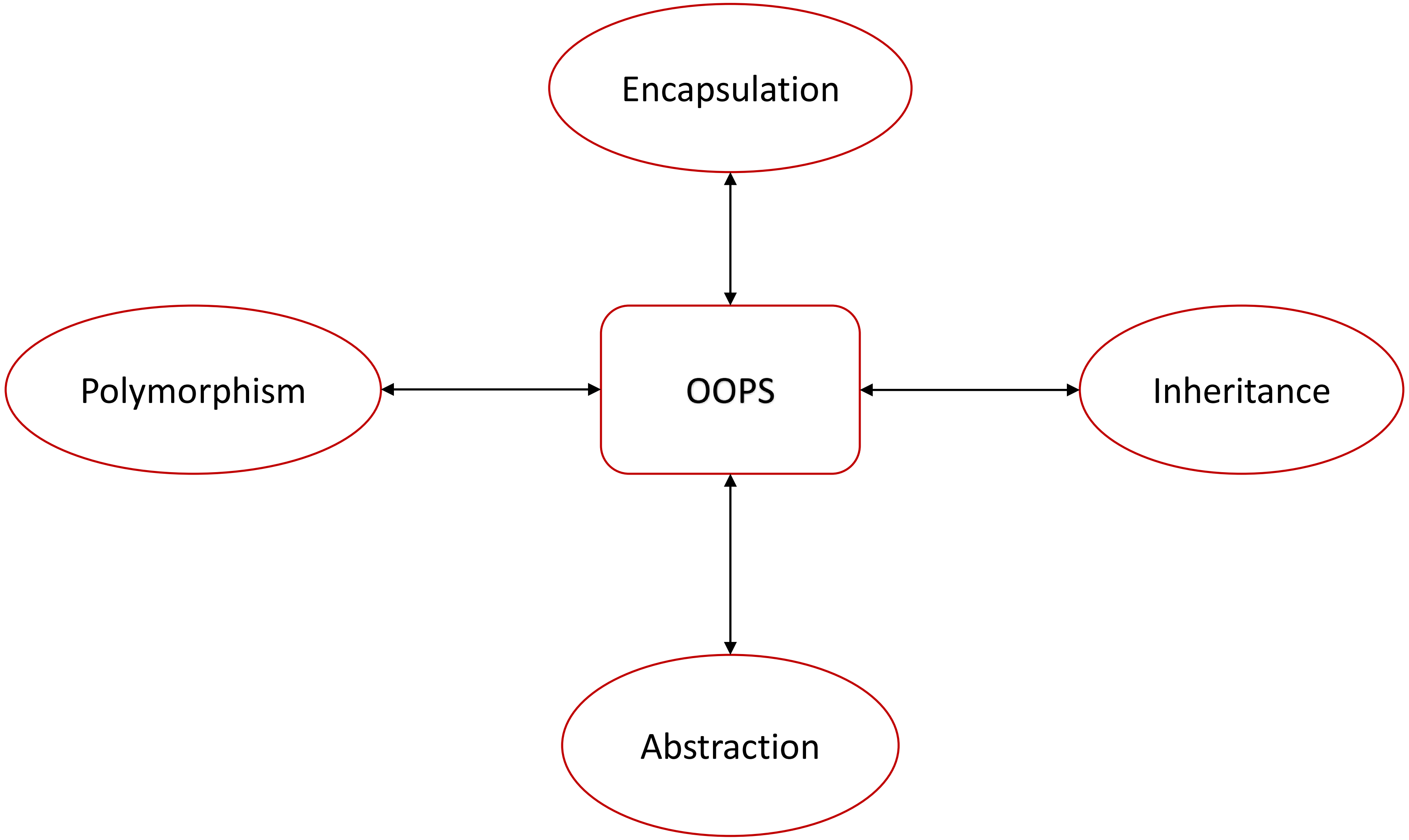
FACE

What is meant by object oriented?

- The direction towards the objects as well as the functionality directed towards objects
- Focuses on objects and classes to design and build applications

Concepts of OOPs

- Inheritance
- Polymorphism
- Abstraction
- Encapsulation


Encapsulation
Polymorphism
OOPS
Inheritance
Abstraction

Class

- A **blueprint** for the object
- A template to create an object
- Created using the keyword **class**
- It supports Camel Case Letters
- Logical Entity

Creating a Class

- The syntax for creating a class :

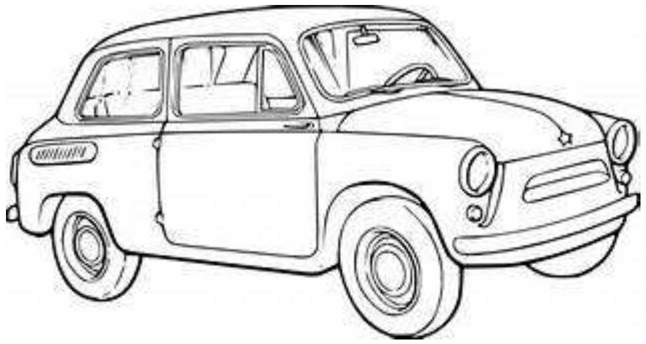
```
class className:
    class documentation
    class_suite
```

Class Members:

Brand
Color

Class Functions:

Apply Break()
Increasing speed()
Decreasing speed()



Class

How to write a class?

```
Class Car:
        //Class Members
        //Functions
```

Class name should start with Capital letter

How will you access the Functions??

By creating a object

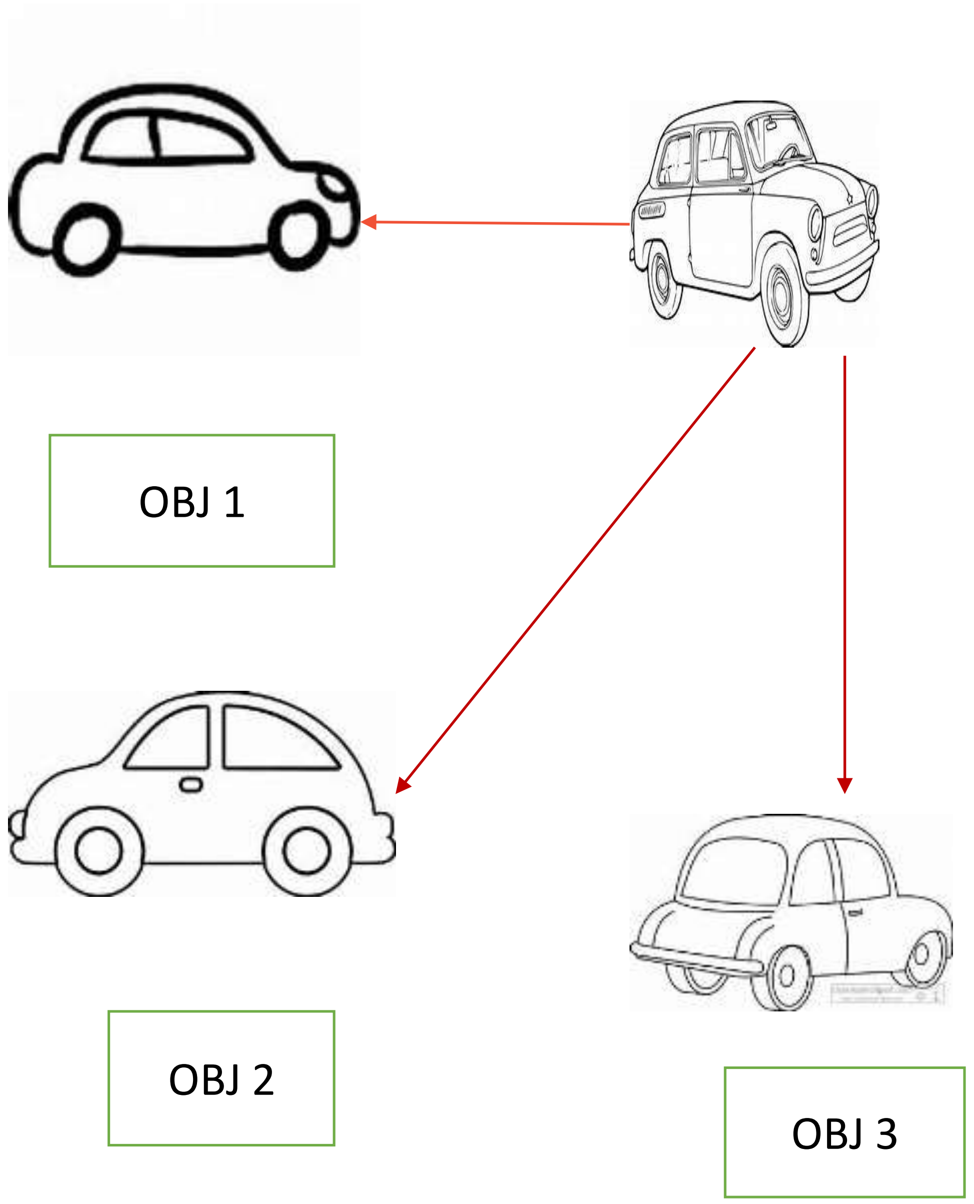
Object

- A collection of data (variables) and methods (functions)
- Created using the constructor of the class
- Physical entity

Creating a Object

- The syntax for creating a object :

  ob=MyClass



```
// Predict the output
class Python:
        def fun(self):
                print("Welcome to oops concept")
pyt = Python()
pyt.fun()
```

```
// Predict the output
class Car:
    'Common_car'
    car=0
    def __init__(self,name,id):
        self.name=name
        self.id=id
        Car.car+=1

    def printCardata(self):
        print("Name: ",self.name,"Id: ",self.id)
c=Car("AUDI",2000000)
c.printCardata()
```

OUTPUT

```
Name:  AUDI Id:  2000000
```

Accessing the attributes

- We can access the objects attributes using the dot(.) operator.
- Class variables would be accessed using class names.

```
// Predict the output
class Car:
    'Common_car'
    car=0
    def __init__(self,name,id):
        self.name=name
        self.id=id
        Car.car+=1

    def printCardata(self):
        print("Name: ",self.name,"Id: ",self.id)
c=Car("AUDI",2000000)
c1=Car("BENZ",3000000)
c2=Car("BMW",1000000)
print("Total Cars: ",Car.car)
c.printCardata()
c1.printCardata()
c2.printCardata()
```

OUTPUT

```
Total Cars:  3
Name:  AUDI Id:  2000000
Name:  BENZ Id:  3000000
Name:  BMW Id:  1000000
```

Accessing using Functions

Functions	Description
getattr(obj,name[,default])	To access the attribute of the object.
hasattr(obj,name)	To check if an attribute exists or not.
setattr(obj,name,value)	To set attribute. If not it would be created.
delattr((obj,name)	To delete an attribute.

Question 1

Which Of The Following Is Required To Create A New Instance Of The Class?

A) A constructor

B) A class

C) A value-returning method

D) A None method

Question 2

What is the output of the below code?

```
class test:
     def __init__(self,a="Welcome to Python oops"):
         self.a=a

     def display(self):
         print(self.a)
obj=test()
obj.display()
```

FACE

Question 2

A) The program has an error because constructor can’t have default arguments

B) Nothing is displayed

C) “Welcome to python oops” is displayed

D) The program has an error display function doesn’t have parameters

Question 3

What is the output of the below code?

```
class test:
    def __init__(self):
        self.variable = 'Car'
        self.Change(self.variable)
    def Change(self, var):
        var = 'Bike'
obj=test()
print(obj.variable)
```

Question 3

A) Error because function change can’t be called in the __init__ function

B) ‘Bike’ is printed

C) ‘Car’ is printed

D) Nothing is printed

Question 4

What is Instantiation in terms of OOP terminology?

A) Deleting an instance of class

B) Modifying an instance of class

C) Copying an instance of class

D) Creating an instance of class

Question 5

What is the output of the below code?

```
class B(object):
  def first(self):
    print("Apple")
  def second():
    print("Mango")
ob = B()
B.first(ob)
```

Question 5

A) It isn’t as the object declaration isn’t right

B) It isn’t as there isn’t any __init__ method for initializing class members

C) Yes, this method of calling is called unbounded method call

D) Yes, this method of calling is called bounded method call