20/6

Encapsulation:-

The major reason for using is

the main thing that we want from bank / bank application is security.

In python there are no access modifiers.

```
Class Demo:
    a=10
    def m1(self):
        print("m1")

o=Demo()
o.m1()
print(o.a)
```

o/p:- m1
10

```
Class Demo:
    __a=10
    def m1(self):
        print("m1")

o=Demo()
o.m1()
print(o.__a)
```

o/p:- m1

Error

using __ we can make that variable as private modifier

```
Class Demo:
    __a=10
    def __m1(self):
        print("m1")
o=Demo()
o.__m1()
print(o.__a)
```

o/p:- Error

```
Class Demo:
    __a=10
    def __m1(self):
        print("m1")
Class Demo1(Demo):
    def m2(self):
        print("Demo1")
o=Demo1()
print(o.__a)
```

o/p: Error

Note: __ is double underscore

We can also make method as a private method.

```
class Demo:
    __a=10
    def __m1(self):
        print(Demo.__a)

o=Demo()
o.__m1()
print(o.__a)
```

o/p: error will be displayed.

```
class Demo:
    __a=10
    def __m1(self):
        print(Demo.__a)
    def m2(self):
        self.__m1()

o=Demo()
o.m2()
```

o/p:- 10

Abstraction:-

In simple level we can say abstraction means hiding. It is also an example for security.

** Is it possible create a object for abstract class? [Interview Question]

No, in abstract class there will be no proper implementation so we cannot create a object.

Ex:-

```
from abc import ABC, abstractmethod

class Demo(ABC):
    @abstractmethod
    def calculate(self):
        pass

class A(Demo):
    def calculate(self, a):
        print("square", a*a)
class B(Demo):
    def calculate(self, a):
        print("cube", a*a*a)

o=A()
p=B()
o.calculate(10)
p.calculate(10)
```

o/p:- Square 100

Cube 1000.