what is Data Abstarction?

Data Abstraction means to hide the unnecessory data/functionality and to provide necessory data/functionality to the customer or user.

we can implement the Data abstarction practically to required Abstarct classes and abstrct methods.

we can create abstract class in python by using ABC class of abc module.

abstarct class contains minimum one abstarct method.

we can create a abstract method by using abstractmethod decorator of abc module.

ex:

```
from abc import ABC,abstractmethod
class animal(ABC):
    legs=4
    def __init__(self):
        self.color="red"
    @abstractmethod
    def bark(self):
        pass
class cat(animal):
    def bark(self):
        print("Cat's are haveing %d-Legs"%cat.legs)
        print("my cat color is",self.color)
        print("Cat's are barkeing maew maew...")
c1=cat()
c1.bark()
print('*'*20)
a=animal()
```

output:

```
Cat's are haveing 4-Legs
my cat color is red
Cat's are barkeing maew maew...
********************
    a=animal()
TypeError: Can't instantiate abstract class animal with abstract method bark
```

how to hide the property of class?
---
we can hide the property of a class,to put double underscores(__) at the begening of that property.

directly we can't access the hidden property of a class from outside of that class.

ex:

```
class test:
    x=10
    __y=20
    def m1(self):
        print(test.x)
        print(test.__y)
t1=test()
t1.m1()
print(t1.x)
print(t1.__y)
```

output:

```
10
20
10
AttributeError: 'test' object has no attribute '__y'
```

directly we can't access the hidden property of a class with in another class.

ex2:

```
class test:
    x=10
    __y=20
    def m1(self):
        print(test.x)
        print(test.__y)
class demo(test):
    def m2(self):
        print(demo.x)
        print(demo.__y)
d1=demo()
d1.m1()
d1.m2()
```

output:

```
10
20
10
AttributeError: type object 'demo' has no attribute '_demo__y'. Did you mean:
'_test__y'?
```

note:
----
whenever we are hide the property of a class,internally that hidden property stored

into the memory location mingle with class name,is known as a name-mangling.

we can access the hidden property of a class from outside the class or with in the another class by using name-mangling syntax,

```
        _classname__attributename
```

ex:

```
class test:
    x=10
    __y=20

print(dir(test))
```

output:

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', ..............,
'_test__y', 'x']
```

ex2:

```
class test:
    x=10
    __y=20
    def m1(self):
        print(test.x)
        print(test.__y)
class demo(test):
    def m2(self):
        print(demo.x)
        print(demo._test__y)
d1=demo()
d1.m1()
d1.m2()
print(d1.x)
print(d1._test__y)
```