Nested classes:
----

we can create a class inside another class is known as a nested class.

syntax

```
class classname:
        -------
        -------
        -------
        class classname:
                ------
                ------
                ------
```

when we are using nested classes?
----

without existing of any one type of object there's no chance to existing another type of object,in that case we are using nested classes.

ex:

```
class Animal:
    Legs=4
    class Cat:
        def bark(self):
            print("Cat's are haveing %d-Legs"%Animal.Legs)
            print("Cat's are barkeing Maew Maew...")
    class Dog:
        def bark(self):
            print("Dog's are haveing %d-Legs"%Animal.Legs)
            print("Dog's are barkeing Bow Bow....")
a=Animal()
c=a.Cat()
d=a.Dog()
c.bark()
d.bark()
```

output:

```
Cat's are haveing 4-Legs
Cat's are barkeing Maew Maew...
Dog's are haveing 4-Legs
Dog's are barkeing Bow Bow....
```

ex2:

```
class Animal:
    Legs=4
    class Cat:
        def bark(self):
            print("Cat's are haveing %d-Legs"%Animal.Legs)
            print("Cat's are barkeing Maew Maew...")
    class Dog:
        def bark(self):
            print("Dog's are haveing %d-Legs"%Animal.Legs)
            print("Dog's are barkeing Bow Bow....")
#a=Animal()
c=Cat()
d=Dog()
c.bark()
d.bark()
```

output:

```
Traceback (most recent call last):
  File "C:/Python310/e.py", line 12, in <module>
    c=Cat()
NameError: name 'Cat' is not defined
```

Nested Methods:
----

we can create/define a method inside another method,is known as a nested method.

```
        syntax
        ------
        class classname:
                -------
                -------
                def methodname(self):
                        -------
                        -------
                        def methodname():
                                --------
                                --------
```

when we are using nested methods:
----

if we want achive the reusability of code within the methods in that case we are using nested methods concept.

ex1:

```
class test:
    def arith(self):
        print("hai")
        self.x=4
        self.y=5
        print(self.x+self.y)
        print("hello")
        self.x="siva"
        self.y="krishna"
        print(self.x+self.y)
        print("good morning")
        self.x=2.3
        self.y=3.2
        print(self.x+self.y)
        print("bye")
t1=test()
t1.arith()

output:
------
hai
9
hello
sivakrishna
good morning
5.5
bye

ex2:
----
class test:
    def arith(self):
        print("hai")
        def add(x,y):
            print(x+y)
        add(4,5)
        print("hello")
        add("siva","krishna")
        print("good morning")
        add(2.3,3.2)
        print("bye")
t1=test()
t1.arith()

output:
------
hai
9
hello
sivakrishna
good morning
5.5
bye
```