In [2]:
```
l=[1,2,3]
r=map(lambda x:x+x,l)
print(list(r))
```

```
[2, 4, 6]
```

In [3]:
```
n=map(lambda n:pow(n,2),l)
print(list(n))
```

```
[1, 4, 9]
```

In [10]:
```
name="vyshu"
(lambda name:print(name)) (name)
```

```
vyshu
```

In [24]:
```
from math import sqrt as st
l=[i for i in range(1,15) if i%2==0]
print(l)
r=map(lambda x:st(x),l)
print(list(r))
```

```
[2, 4, 6, 8, 10, 12, 14]
[1.4142135623730951, 2.0, 2.449489742783178, 2.8284271247461903, 3.162277660168
3795, 3.4641016151377544, 3.7416573867739413]
```

In [32]:
```
from abc import ABC, abstractmethod
class abstractmethod(ABC):
    @abstractmethod #used to make the class abstarct
    def display(self):
        None
    def show(self):
        None
```

In [30]:
```
class demo(abstractmethod):
    def display(self):
        print("Not Hiding")
    def show(self):
        print("Details")
abt=demo()
abt.display()
abt.show()
```

```
Not Hiding
Details
```

In [36]:
```
#single
class parent:
    def display(self):
        print("parent class")
class child(parent):
    def show(self):
        print("child class")
c=child()
c.display()
c.show()
```

```
parent class
child class
```

In [37]:
```
class a:
    n=30
class b(a):
    def sum(self):
        c=self.n+70
        print(c)
bb=b()
bb.sum()
```

```
100
```

In [ ]:
```

```