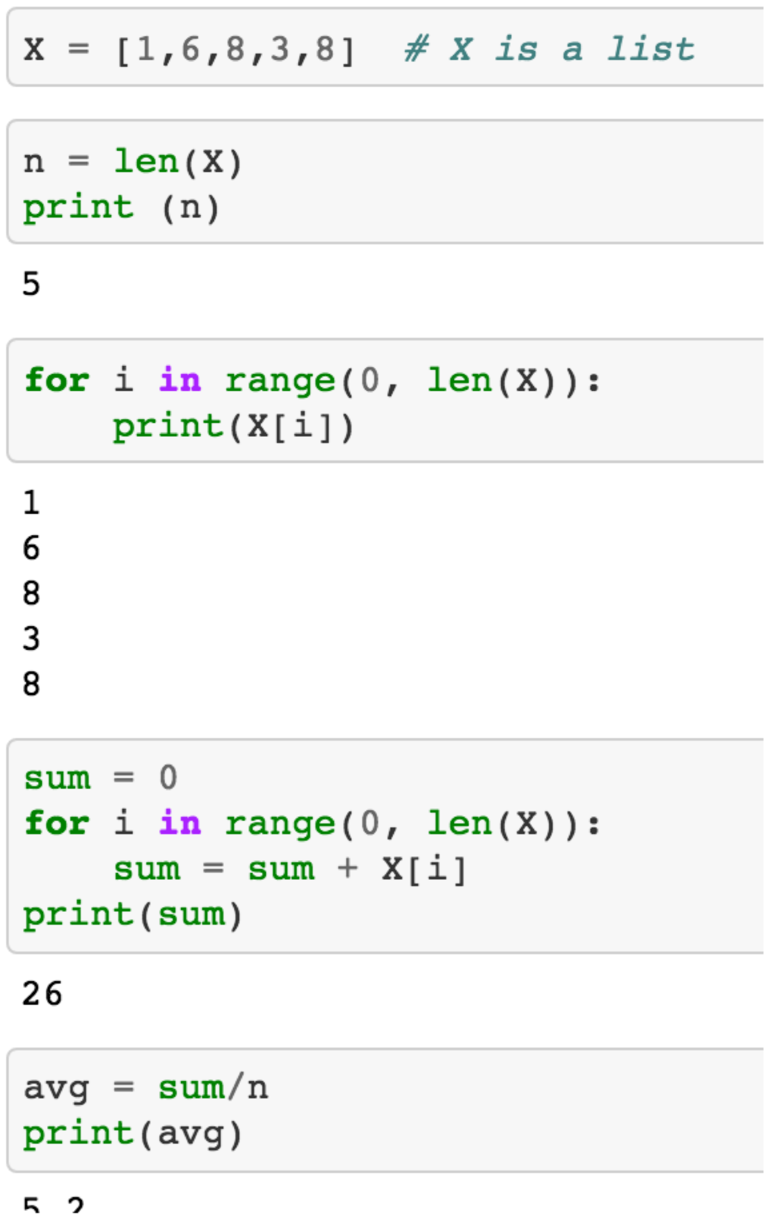
Homework 3

Iteration

notebook



```
X = [1,6,8,3,8]  # X is a list
```

```
n = len(X)
print (n)
```

```
5
```

```
for i in range(0, len(X)):
    print(X[i])
```

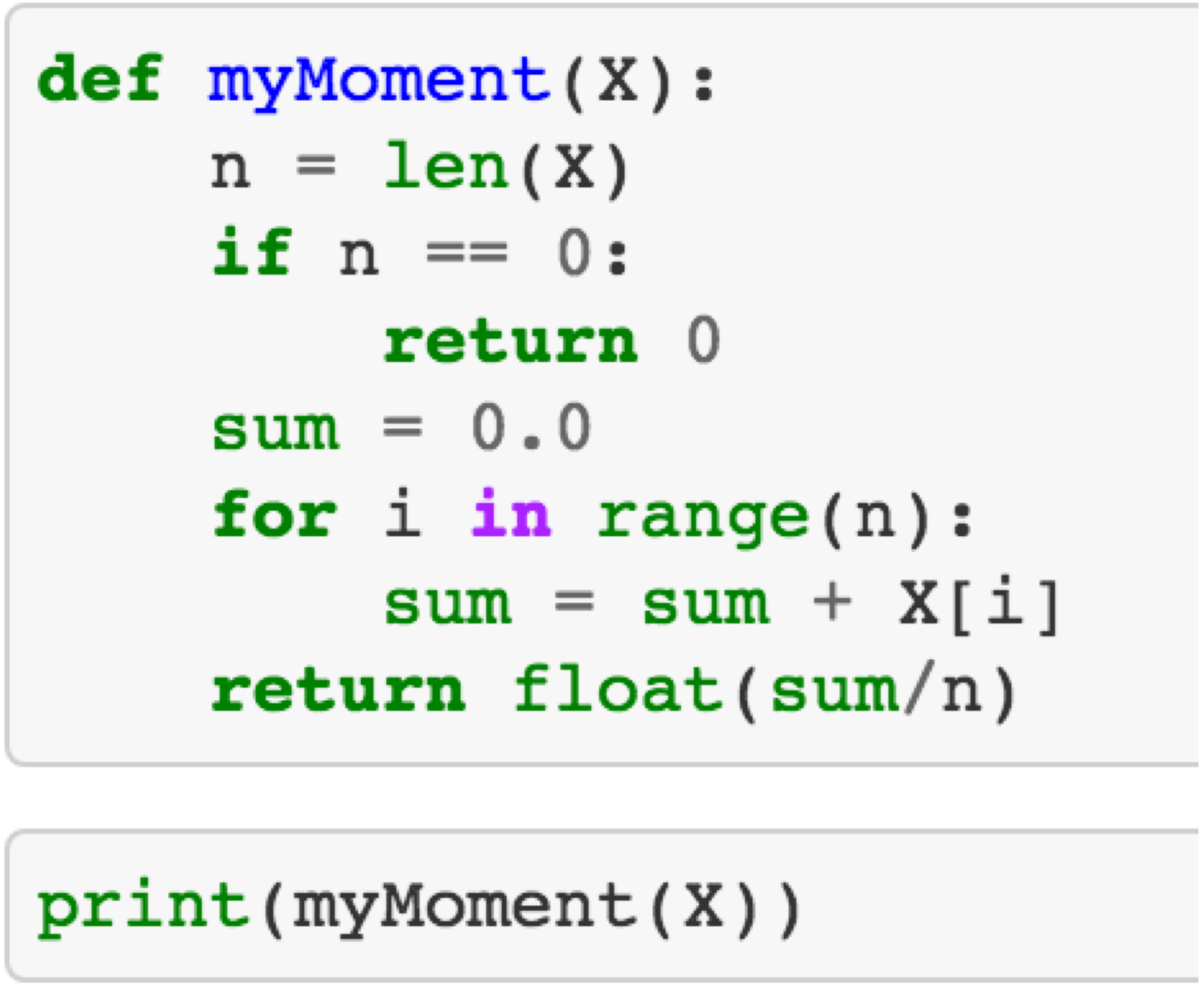
```
1
6
8
3
8
```

```
sum = 0
for i in range(0, len(X)):
    sum = sum + X[i]
print(sum)
```

```
26
```

```
avg = sum/n
print(avg)
```

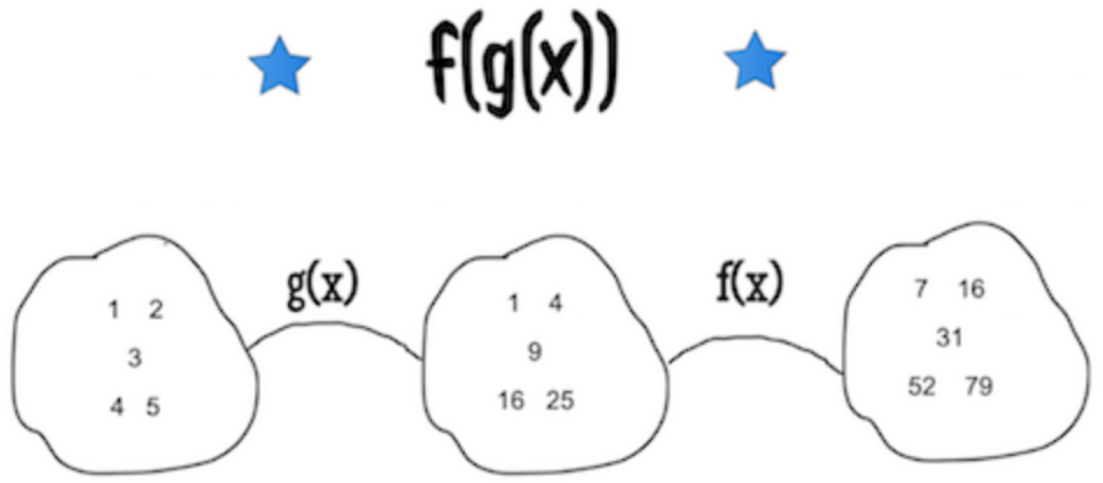
```
5.2
```



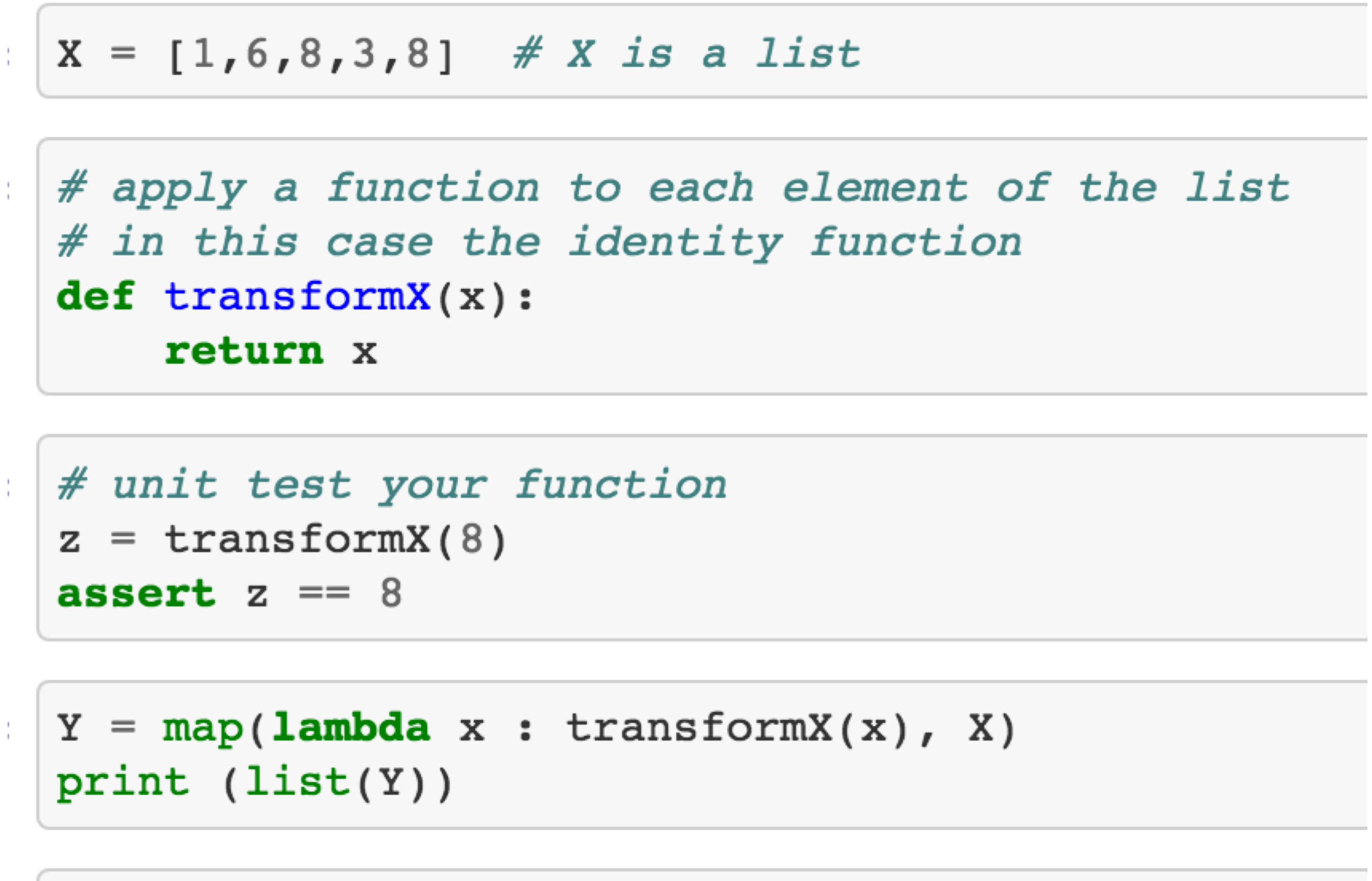
```python
def myMoment(X):
    n = len(X)
    if n == 0:
        return 0
    sum = 0.0
    for i in range(n):
        sum = sum + X[i]
    return float(sum/n)
```

```python
print(myMoment(X))
```

Functional Programming



Map

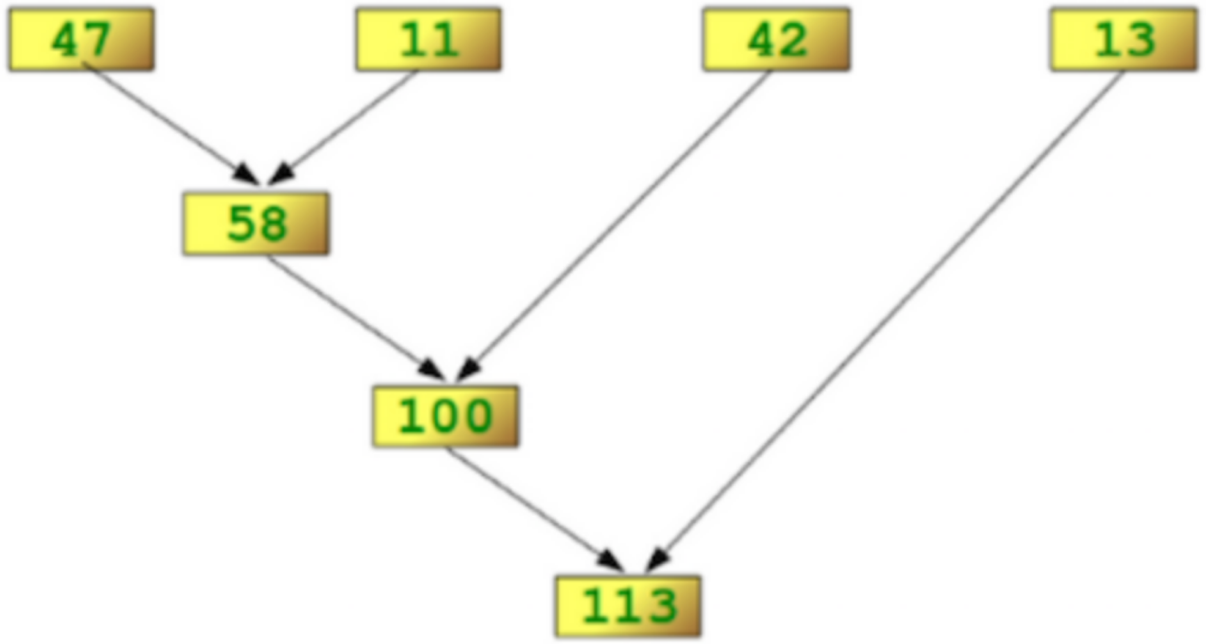

```
X = [1,6,8,3,8]  # X is a list
```

```
# apply a function to each element of the list
# in this case the identity function
def transformX(x):
    return x
```

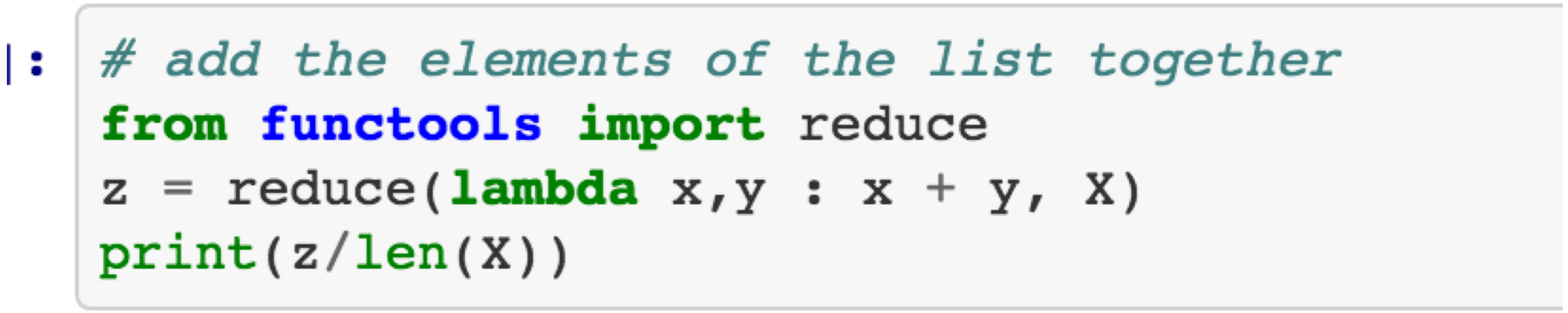
```
# unit test your function
z = transformX(8)
assert z == 8
```

```
Y = map(lambda x : transformX(x), X)
print (list(Y))
```

Reduce takes a list, a binary operation, and an identity element for the operation. It returns a value representing the binary operator applied to successive application of the operations to intermediate results.



Reduce



```
# add the elements of the list together
from functools import reduce
z = reduce(lambda x,y : x + y, X)
print(z/len(X))
```

```python
def moment(X):
    n = len(X)
    if n == 0 :
        return 0.0
    transformedX = map(lambda x : transformX(x), X)
    sumX = reduce(lambda x, y : x + y, transformedX)
    return sumX/float(n)
```

```python
print (moment(X))
```