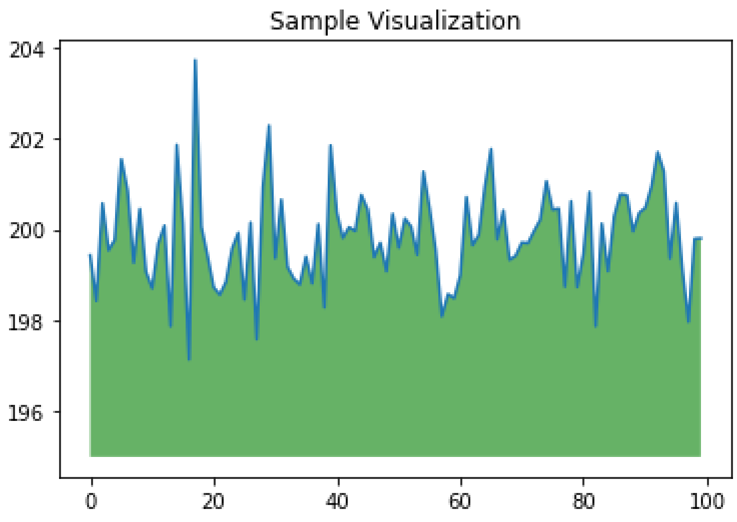
MOST useful numpy function

using nditer function for traversing and array multiplication by a constant

In [33]:
```
import numpy as np
a= np.array([[[ 1,  2 , 3]],[[ 4 , 5 , 6]],[[ 7 , 8 , 9]],[[10, 11, 12]]])
arr=a
arr*3
for i in np.nditer(a):
    print(i)
print(arr*3)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
[[[ 3  6  9]]

 [[12 15 18]]

 [[21 24 27]]

 [[30 33 36]]]
```

In [36]:
```
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Sample Visualization")
plt.show()
```



returns dimension of the the matrix

In [41]:
```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
print (a.shape)
```

```
(3, 3)
```

changes the dimesion of array

In [ ]:
```
import numpy as np

a= np.array([[1,2,3],[4,5,6]])
b=np.array([[1,2,3],[4,5,6]])
a.shape = (3,2)
b.shape = (6,1)

#here array is of 2*3 which can be converted to 6*1 ,1*6 or 3*2
print (a)
print(b)
```

reshapinG ARRAY

In [56]:
```
import numpy as np
a= np.array([[[ 1,  2 , 3]],[[ 4 , 5 , 6]],[[ 7 , 8 , 9]],[[10, 11, 12]]])
arr=a.reshape(6,2)
#here array is of dim 4*3 i.e is 12 so we can reshape it into 3*4 ,2*6,6*2,12*1,1*12
#it should have space for every element
print(arr)
```

```
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]]
```

an array of evenly spaced numbers

In [62]:
```
import numpy as np
```

```
a = np.arange(24)
print (a)
b=a.reshape(8,3)
print(b)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]
```

CREATING ARRAY using ndim function

In [65]:

```
# this is one dimensional array
import numpy as np
a = np.arange(24)
a.ndim

# now reshape it
b = a.reshape(2,4,3)
print (b)
# b is having three dimensions
```

```
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]
  [ 9 10 11]]

 [[12 13 14]
  [15 16 17]
  [18 19 20]
  [21 22 23]]]
```

creating array with 0 and 1

In [70]:

```
import numpy as np
x = np.ones([2,2], dtype = int)
print (x)
import numpy as np
x = np.zeros([2,2], dtype = int)
print (x)
```

```
[[1 1]
 [1 1]]
[[0 0]
 [0 0]]
```

convert list to ndarray

In [73]:

```
import numpy as np

x = [1,2,3]
a = np.asarray(x, dtype = float)
print (a)
```

```
[1. 2. 3.]
```

numpy.arange(start, stop, step, dtype=' ')

In [76]:
```python
import numpy as np
x = np.arange(10,20,2)
print (x)
```

```
[10 12 14 16 18]
```

slice(start, stop, and step)

In [79]:
```python
import numpy as np
a = np.arange(10)
s = slice(2,7,2)
print (a[s])
```

```
[2 4 6]
```

slicing using index

In [84]:
```python
import numpy as np
a = np.array([[1,2,3],[3,4,5],[4,5,6]])
print (a )

# slice items starting from index
print ('Now we will slice the array from the index a[1:]')
print (a[1:])
```

```
[[1 2 3]
 [3 4 5]
 [4 5 6]]
Now we will slice the array from the index a[1:]
[[3 4 5]
 [4 5 6]]
```

indexing multiple values from array

In [86]:
```python
import numpy as np

x = np.array([[1, 2], [3, 4], [5, 6]])
y = x[[0,1,2], [0,1,0]]
#here [0,1,2] represent x coordinate and [0,1,0] represents y coordinate so it finaa
print (y)
```

```
[1 4 5]
```

printing corner elements of the array

In [90]:
```python
import numpy as np
x = np.array([[ 0,  1,  2],[ 3,  4,  5],[ 6,  7,  8],[ 9, 10, 11]])

print ('Our array is:')
print (x)
print ('\n')

rows = np.array([[0,0],[3,3]])
cols = np.array([[0,2],[0,2]])
y = x[rows,cols]

print ('The corner elements of this array are:' )
print (y)
```

```
Our array is:
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]


The corner elements of this array are:
[[ 0  2]
 [ 9 11]]
```

printing elements greater than a certain value

In [92]:

```
import numpy as np
x = np.array([[ 0,  1,  2],[ 3,  4,  5],[ 6,  7,  8],[ 9, 10, 11]])

print ('Our array is:')
print (x)
print ('\n')

# Now we will print the items greater than 5
print ('The items greater than 5 are:')
print (x[x > 5])
```

```
Our array is:
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]


The items greater than 5 are:
[ 6  7  8  9 10 11]
```

Example of traversing

In [95]:

```
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)

print ('Original array is:')
print (a)
print ('\n')

print ('Modified array is:')
for x in np.nditer(a):
   print (x)
```

```
Original array is:
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


Modified array is:
0
5
10
15
20
25
30
35
40
45
50
55
```

Transpose of a matrix

In [97]:

```python
import numpy as np
a = np.arange(0,60,5)
a = a.reshape(3,4)

print ('Original array is:')
print (a)
print ('\n')

print ('Transpose of the original array is:')
b = a.T
print (b)
print ('\n')

print ('Modified array is:')
for x in np.nditer(b):
   print (x)
```

```
Original array is:
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


Transpose of the original array is:
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]


Modified array is:
0
5
10
15
20
25
30
35
40
45
50
55
```

Flattening array into 1D from any dimesion

using 'flatten' function

In [99]:

```
import numpy as np
a = np.arange(8).reshape(2,4)

print ('The original array is:' )
print (a)
print ('\n')

print ('The flattened array is:' )
print (a.flatten())
```

```
The original array is:
[[0 1 2 3]
 [4 5 6 7]]


The flattened array is:
[0 1 2 3 4 5 6 7]
```

determinant

In [102…

```
import numpy as np
b = np.array([[6,1,1], [4, -2, 5], [2,8,7]])
print (b)
print (np.linalg.det(b))
```

```
[[ 6  1  1]
 [ 4 -2  5]
 [ 2  8  7]]
-306.0
```

In [ ]:

```
```