Recap

In [1]:
```
x_val = [1, 2, 3]
for x in x_val: ##Looping without indices
    print(x)
```

```
1
2
3
```

In [2]:
```
for i in range(len(x_val)): ##Looping with indices
    print(x_val[i])
```

```
1
2
3
```

In [3]:
```
names = ['A', 'B']
marks = ["C", 'D']
dict(zip(names, marks))
```

Out[3]:
```
{'A': 'C', 'B': 'D'}
```

In [4]:
```
cities = ('Dhaka', 'Tokyo', 'Seoul', 'Tehran', 'Doha')
countries = ('BD', 'JP', 'SK', 'IR', 'QR')

for city, country in zip(cities, countries):
    print(f'The city is {city} and corresponding country {country}')
```

```
The city is Dhaka and corresponding country BD
The city is Tokyo and corresponding country JP
The city is Seoul and corresponding country SK
The city is Tehran and corresponding country IR
The city is Doha and corresponding country QR
```

In [5]:
```
for index, number in enumerate(x_val):
    print(f'xa_val[{index}] = {number}')
```

```
xa_val[0] = 1
xa_val[1] = 2
xa_val[2] = 3
```

Home study % - use % modulo printing

Practice at your own interest

.format()

Function

In [6]:
```
def f(string):
    count = 0
    for letter in string:
        if letter == letter.upper() and letter.isalpha():
            count = count + 1
```

```
    return count

f('Winter iS Bautiful but Scary')
```

Out[6]: 4

Find root of ax^2 + bx + c, consider fixed x value but take multiple values for coeffs. Hints: def f(x, coeff), coeff = (2, 1).

Write a function in Python which takes two sequences as arguments and returns True if every element in a sequence is also an element of second sequnce, else False.

In [7]:
```
def f(x):
    return x**3
```

In [8]:
```
f(3)
```

Out[8]: 27

In [9]:
```
from scipy.integrate import quad
print(quad(lambda x: x**2, 0 , 3))
```

```
(9.000000000000002, 9.992007221626411e-14)
```

In [10]:
```
f = (lambda x: x**3)(3)
f
```

Out[10]: 27

Numpy

In [11]:
```
import numpy as np
```

In [12]:
```
a = np.zeros(3, dtype = int)
```

In [13]:
```
a
```

Out[13]: array([0, 0, 0])

In [14]:
```
type(a)
```

Out[14]:
```
numpy.ndarray
```

In [15]:
```
a.shape
```

Out[15]:
```
(3,)
```

In [16]:
```
a = np.zeros(10)
a
```

Out[16]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [17]:
```
a.shape = (5, 2)
a
```

Out[17]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [18]:
```
a = np.empty(3)
a
```

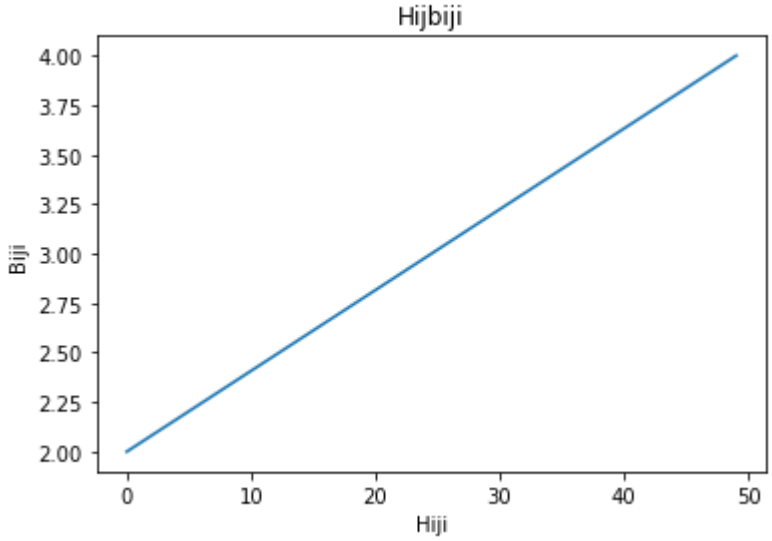
Out[18]:
```
array([1.20830661e-311, 0.00000000e+000, 1.33360289e+241])
```

In [19]:
```
a = np.linspace(2, 4, 50)
a
```

Out[19]:
```
array([2.        , 2.04081633, 2.08163265, 2.12244898, 2.16326531,
       2.20408163, 2.24489796, 2.28571429, 2.32653061, 2.36734694,
       2.40816327, 2.44897959, 2.48979592, 2.53061224, 2.57142857,
       2.6122449 , 2.65306122, 2.69387755, 2.73469388, 2.7755102 ,
       2.81632653, 2.85714286, 2.89795918, 2.93877551, 2.97959184,
       3.02040816, 3.06122449, 3.10204082, 3.14285714, 3.18367347,
       3.2244898 , 3.26530612, 3.30612245, 3.34693878, 3.3877551 ,
       3.42857143, 3.46938776, 3.51020408, 3.55102041, 3.59183673,
       3.63265306, 3.67346939, 3.71428571, 3.75510204, 3.79591837,
       3.83673469, 3.87755102, 3.91836735, 3.95918367, 4.        ])
```

In [20]:
```
import matplotlib.pyplot as plt
```

In [21]:
```
plt.plot(a)
plt.title("Hijbiji")
plt.xlabel("Hiji")
plt.ylabel("Biji")
plt.show()
```



In [22]:
```
x = np.identity(4)
x
```

Out[22]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [23]:
```
a = np.array((10, 20), dtype = float)
a
```

Out[23]:
```
array([10., 20.])
```

In [24]:
```
z = np.linspace(1, 2, 5)
z[0]
```

Out[24]:
```
1.0
```

In [25]:
```
z[-1]
```

Out[25]:
```
2.0
```

In [26]:
```
b = np.array([[1,2], [3,4]])
b
```

Out[26]:
```
array([[1, 2],
       [3, 4]])
```

In [27]:
```
b[0,1]
```

Out[27]:
```
2
```

In [28]:
```
b[0, :]
```

Out[28]:
```
array([1, 2])
```

In [29]:
```
b[:, 1]
```

Out[29]:
```
array([2, 4])
```

In [30]:
```
d = np.array((12, 16 , 14, 18), dtype = float)
e = np.array((13, 17, 19, 21))

d@e
```

Out[30]:
```
1072.0
```

In [31]:
```
a = np.random.randn(5)
a
```

Out[31]:
```
array([-0.2412457 ,  0.68791722,  0.25215555,  0.77168298,  0.80861295])
```

In [32]:
```
b = a
b[0] = 0.0
a
```

Out[32]:
```
array([0.        , 0.68791722, 0.25215555, 0.77168298, 0.80861295])
```

In [33]:
```
b
```

Out[33]:
```
array([0.        , 0.68791722, 0.25215555, 0.77168298, 0.80861295])
```

In [34]:
```
b = np.copy(a)
a
```

Out[34]:
```
array([0.        , 0.68791722, 0.25215555, 0.77168298, 0.80861295])
```

In [35]:
```
b[0] = 0
b
```

Out[35]:
```
array([0.        , 0.68791722, 0.25215555, 0.77168298, 0.80861295])
```

In [36]:
```
b[:] = 1
b
```

Out[36]:
```
array([1., 1., 1., 1., 1.])
```

In [37]:
```
a
```

Out[37]:
```
array([0.        , 0.68791722, 0.25215555, 0.77168298, 0.80861295])
```

In [38]:
```
x = np.array([1,2,3])
np.sin(x)
```

Out[38]:
```
array([0.84147098, 0.90929743, 0.14112001])
```

In [39]:
```
np.sqrt(2 * np.pi)
```

Out[39]:
```
2.5066282746310002
```

In [40]:
```
'''
Hints for plotting
x = np.linspace
y = np.sin(x)
--->> put both x and y in plotting
'''
```

Out[40]:
```
'\nHints for plotting\nx = np.linspace\ny = np.sin(x)\n--->> put both x and y in plottin
g\n'
```

Plot sin, cosine, and tan function using np library. Use linspace to generate values.

In [41]:
```
def f(x):
    return 1 if x > 0 else 0
```

In [42]:
```
f(1)
```

Out[42]:
```
1
```

In [43]:
```
x = np.random.randn(4)
f(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-43-46a31cfbc466> in <module>
      1 x = np.random.randn(4)
----> 2 f(x)

<ipython-input-41-fa2da181b9df> in f(x)
      1 def f(x):
----> 2     return 1 if x > 0 else 0

ValueError: The truth value of an array with more than one element is ambiguous. Use a.a
ny() or a.all()
```

In [44]:
```
np.where(x > 0, 1, 0)
```

Out[44]:
```
array([1, 1, 0, 1])
```

In [45]:
```
f = np.vectorize(f)
f(x)
```

Out[45]:
```
array([1, 1, 0, 1])
```

In [ ]:
```

```