In [1]:
```
import numpy as np
```

In [2]:
```
cvalues = [20.1, 20.8, 21.9, 22.5, 22.7, 22.3, 21.8, 21.2, 20.9,
20.1] #list of temperature in degree calcius
C = np.array(cvalues)
print(cvalues)
print(type(cvalues))
print(C)
print(type(C))
```

```
[20.1, 20.8, 21.9, 22.5, 22.7, 22.3, 21.8, 21.2, 20.9, 20.1]
<class 'list'>
[20.1 20.8 21.9 22.5 22.7 22.3 21.8 21.2 20.9 20.1]
<class 'numpy.ndarray'>
```

Element-wise Operations in Numpy (Scalar Operations)

In [3]:
```
F = C * 9/5 + 32
print(F)
# examples of scalar operations
A = np.array([[1,2,3],[4,5,6]])
print(A)
print(A.shape)

B = np.array([[7,8,9],[10,11,12]])
print(B)
print(B.shape)

C = A + B
print(C)
print(C.shape)
```

```
[68.18 69.44 71.42 72.5  72.86 72.14 71.24 70.16 69.62 68.18]
[[1 2 3]
 [4 5 6]]
(2, 3)
[[ 7  8  9]
 [10 11 12]]
(2, 3)
[[ 8 10 12]
 [14 16 18]]
(2, 3)
```

Array Indexing

In [4]:
```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
b = a[:,0:2]
print(b)
print(a[0,0])
print(a)
```

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

Boolean Array Indexing (for Filtering)

In [5]:
```
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2)
print(bool_idx)
print(a[bool_idx])

print(a[a > 2])
```

```
[[False False]
 [ True  True]
 [ True  True]]
[3 4 5 6]
[3 4 5 6]
```

Numpy Simple Math examples

In [6]:
```
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

print(x + y)# Elementwise sum
print(np.add(x, y))

print(x - y)# Elementwise difference
print(np.subtract(x, y))

print(x * y)# Elementwise product
print(np.multiply(x, y))

print(x / y)# Elementwise division
print(np.divide(x, y))

print(np.sqrt(x))# Elementwise square root
```

```
[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
[[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[1.         1.41421356]
 [1.73205081 2.        ]]
```

Numpy Dot product and Vector and Matrix Multiplication

In [7]:
```
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)
v = np.array([9,10])
w = np.array([11, 12])

print(v.dot(w))# Inner product of vectors
print(np.dot(v, w))

print(x.dot(v))# Matrix / vector product
print(np.dot(x, v))

print(x.dot(y))# Matrix / matrix product
print(np.dot(x, y))
```

```
219
219
[29. 67.]
[29. 67.]
[[19. 22.]
 [43. 50.]]
[[19. 22.]
 [43. 50.]]
```

Numpy Mathematical Functions

In [8]:
```
x = np.array([[1,2],[3,4]])

print(np.sum(x)) # Compute sum of all elements
print(np.sum(x, axis=0)) # Compute sum of each column
print(np.sum(x, axis=1)) # Compute sum of each row
```

```
10
[4 6]
[3 7]
```

Numpy Statistical Functions

In [9]:
```
data1 = np.arange(1.5)
print(np.average(data1))
data2 = np.arange(6).reshape(3,2)
print(data2)
print(np.average(data2, axis = 0))
print(np.average(data2, axis = 1))
```

```
0.5
[[0 1]
 [2 3]
 [4 5]]
[2. 3.]
[0.5 2.5 4.5]
```

Broadcasting

Adding a constant vector to each row of a matrix

In [11]:
```
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = np.empty_like(x)

for i in range(4):
    y[i, :] = x[i, :] + v
```

In [12]:
```
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))
y = x + vv
print(y)
```

```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

Using Broadcasting

In [13]:
```
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])

y = x + v # Add v to each row of x using broadcasting
print(y)
```

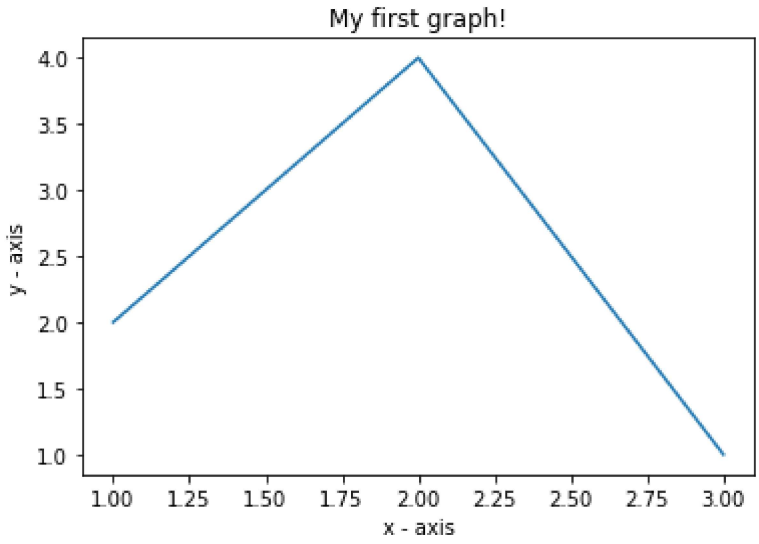
```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

Some special Numpy Arrays

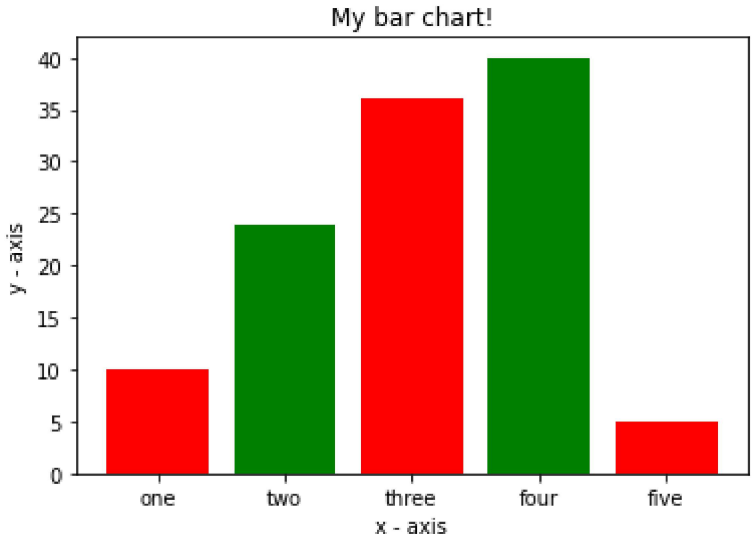
In [15]:
```
np.zeros(5)
np.zeros((2,3))
np.random.rand(2,3)
np.full((2,2),7)
np.eye(3)
np.arange(2,10,2)
np.linspace(0,1,5)
a = np.array([3,6,9,12])
np.reshape(a,(2,2))
a = np.ones((2,2))
b = a.flatten()
a = np.array([[1,2,3],
[4,5,6]])
b = np.transpose(a)
```

In [16]:
```
import matplotlib.pyplot as plt
```

In [18]:
```
x = np.array([1,2,3])# x axis values

y = np.array([2,4,1])# y axis values

plt.plot(x, y)# plotting the points

plt.xlabel('x - axis')# naming the x axis

plt.ylabel('y - axis')# naming the y axis

plt.title('My first graph!')# setting a title to my graph

plt.show()# function to show the plot
```



In [19]:
```
left = [1, 2, 3, 4, 5]# x-coordinates of left sides of bars

height = [10, 24, 36, 40, 5]# heights of bars

tick_label = ['one', 'two', 'three', 'four', 'five']# labels for bars

plt.bar(left, height, tick_label = tick_label, width = 0.8, color =
['red', 'green'])# plotting a bar chart

plt.xlabel('x - axis')# naming the x-axis

plt.ylabel('y - axis')# naming the y-axis

plt.title('My bar chart!')# setting title

plt.show()# function to show the plot
```



In [20]:
```
A = np.array([1,2,3,4,5,6])
print(A)

B = np.array([10,20,30,40,50,60])
print(B)

print(A+B)

print(A-B)
```

```
[1 2 3 4 5 6]
[10 20 30 40 50 60]
[11 22 33 44 55 66]
[ -9 -18 -27 -36 -45 -54]
```

In [21]:
```
A = np.array([1,2,3,4,5,6])
print(A)

B = np.array([10,20,30,40,50,60])
print(B)

print(A+B)

print(A-B)
```

```
[1 2 3 4 5 6]
[10 20 30 40 50 60]
[11 22 33 44 55 66]
[ -9 -18 -27 -36 -45 -54]
```

In [23]:
```
temp = np.array([10, 15, 20.5, 30, 37])

temp_fahrenheit = temp * 1.8 + 32

print(temp_fahrenheit)
```

```
[50.  59.  68.9 86.  98.6]
```

In [24]:
```
print(A.shape)
```

```
(6,)
```

In [25]:
```
M1 = np.array([[1,2,3],[4,5,6]])
# 1 2 3
# 4 5 6
M2 = np.array([[7,8,9],[3,4,5]])
# 7 8 9
# 3 4 5

print(M1)
print(M2)
print(M1.shape)
```

```
[[1 2 3]
 [4 5 6]]
[[7 8 9]
 [3 4 5]]
(2, 3)
```

In [26]:
```
M3 = M1+M2
M4 = M1-M2
M5 = M1*M2 #scalar multiplication
M6 = M1/M2

print(M3)

print(M4)

print(M5)

print(M6)
```

```
[[ 8 10 12]
 [ 7  9 11]]
[[-6 -6 -6]
 [ 1  1  1]]
[[ 7 16 27]
 [12 20 30]]
[[0.14285714 0.25       0.33333333]
 [1.33333333 1.25       1.2       ]]
```

In [27]:
```
M1 = np.array([[1,2,3],[4,5,6],[7,8,9]])

M7 = M1[:,0:2]

print(M7)
```

```
[[1 2]
 [4 5]
 [7 8]]
```

In [28]:
```
M8 = M1[-1,-2:]# slice the last rows with last two columns
print(M8)
```

```
[8 9]
```

In [29]:
```
M9 = np.array([[1,2,3,4],[56, 43, 23, 78],
               [100, 101, 102, 103]])
#   1   2   3   4
#  56  43  23  78
# 100 101 102 103

bool_idx = (M9%2==0)

print(bool_idx)
```

```
[[False  True False  True]
 [ True False False  True]
 [ True False  True False]]
```

In [30]:
```
print(M9[bool_idx])
```

```
[  2   4  56  78 100 102]
```

In [32]:
```
M1 = np.array([[1,2,3],[4,5,6],[7,8,9]])

M2 = np.array([[-1,-2,-3],[-4,-5,-6],[-7,-8,-9]])
print(np.add(M1, M2))

print(np.subtract(M1, M2))

print(np.multiply(M1, M2))

print(np.divide(M1, M2))

print(np.sqrt(M1))
```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
[[ 2  4  6]
 [ 8 10 12]
 [14 16 18]]
[[ -1  -4  -9]
 [-16 -25 -36]
 [-49 -64 -81]]
[[-1. -1. -1.]
 [-1. -1. -1.]
 [-1. -1. -1.]]
[[1.         1.41421356 1.73205081]
 [2.         2.23606798 2.44948974]
 [2.64575131 2.82842712 3.        ]]
```

In [33]:
```
x = np.array([[1,2],[3,4]])
# 1 2       5 6
# 3 4       7 8

# 1.5+2.7   1.6+2.8     19  22
# 3.5+4.7   3.6+4.8     43  50

y = np.array([[5,6],[7,8]])

print(x.dot(y))
```

```
[[19 22]
 [43 50]]
```

In [34]:
```
print(np.dot(x,y))
print(x@y)
print(np.matmul(x,y))
```

```
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
```

Python code (without using any numpy function)

performing Matrix-Matrix multiplication)

In [35]:
```
v1 = np.array([1,2,3]) # i+2j+3k
v2 = np.array([-1,3,-2]) # -i+3j-2k

# 1.-1+2.3-3.2 = -1

print(np.dot(v1, v2))
```

```
-1
```

In [36]:
```
x = np.array([[1,2],[3,4],[5,6]]) # shape(3, 2)
# 1 2
# 3 4
# 5 6

# 1 3 5     # 1 2 3     1.1+3.2+5.3 = 22
# 2 4 6                 2.1+4.2+6.3 = 28

v1 = np.array([1,2,3])  #shape(3,) here 3 is no. of col
# print(x.shape)
# print(v1.shape)
print(np.dot(np.transpose(x),v1))
```

```
[22 28]
```

In [37]:
```
data1 = np.arange(10)
print(data1)
print(np.average(data1))
```

```
[0 1 2 3 4 5 6 7 8 9]
4.5
```

In [38]:
```
data2 = np.arange(12).reshape(4,3)
print(data2)
```

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

In [44]:
```
print(np.average(data2, axis = 0))
print(np.average(data2, axis = 1))
```

```
[4.5 5.5 6.5]
[ 1.  4.  7. 10.]
```

In [45]:
```
print(np.sum(data2))
```

```
66
```

In [46]:
```
M11 = np.zeros((3,3))
print(M11)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [47]:
```
M12 = np.random.rand(3,3)
print(M12)
```

```
[[0.01949705 0.45312803 0.87915095]
 [0.64752859 0.32065125 0.88089956]
 [0.98676638 0.36128093 0.01377339]]
```

In [48]:
```
M13 = np.linspace(0, 90, 10).reshape(2, 5)
print(M13)
```

```
[[ 0. 10. 20. 30. 40.]
 [50. 60. 70. 80. 90.]]
```

In [49]:
```
M14 = np.eye(4)
print(M14)
```

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

In [50]:
```
data2 = np.arange(12).reshape(4,3)
print(data2)
```

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

In [51]:
```
v3 = np.array([10, 20, 30])

Z = data2 + v3
print(Z)
```

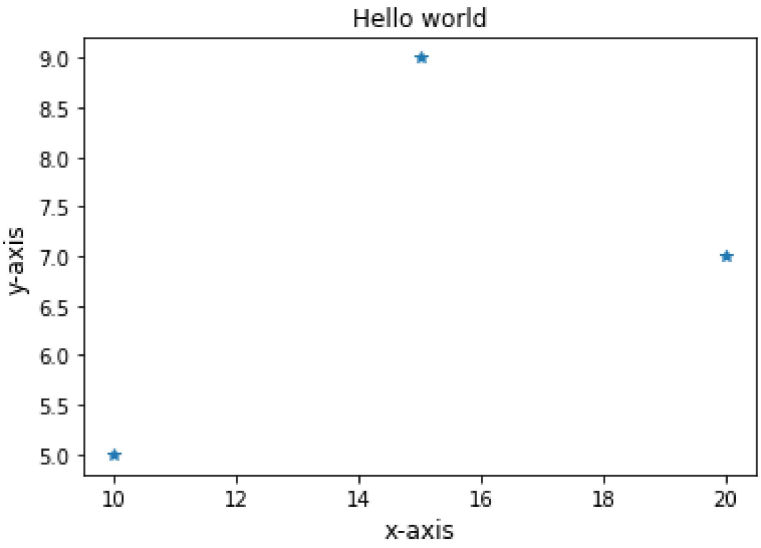
```
[[10 21 32]
 [13 24 35]
 [16 27 38]
 [19 30 41]]
```

In [52]:
```
v4 = np.array([1, 2, 3, 4])

Z1 = np.transpose(data2) + v4

print(Z1)
```

```
[[ 1  5  9 13]
 [ 2  6 10 14]
 [ 3  7 11 15]]
```

In [53]:
```
x = np.array([10, 15, 20])
y = np.array([5, 9, 7])

plt.plot(x, y, '*')

plt.xlabel('x-axis', fontsize = 12)
plt.ylabel('y-axis', fontsize = 12)

plt.title('Hello world')
```

Out[53]:
```
Text(0.5, 1.0, 'Hello world')
```



In [54]:
```
x1 = np.arange(6)
print(x1)

freq = np.linspace(20, 50, 6)
print(freq)

ticklabel = ['Ban', 'Ind', 'Pak', 'Sri', 'Mal', 'Nep']
plt.bar(x1, freq, tick_label = ticklabel, width = 0.8)

plt.xlabel ('x-axis')
plt.ylabel('Frequency')

plt.title ('Bar chart')

plt.show()
```

```
[0 1 2 3 4 5]
[20. 26. 32. 38. 44. 50.]
```