

1. แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม (NumPy)

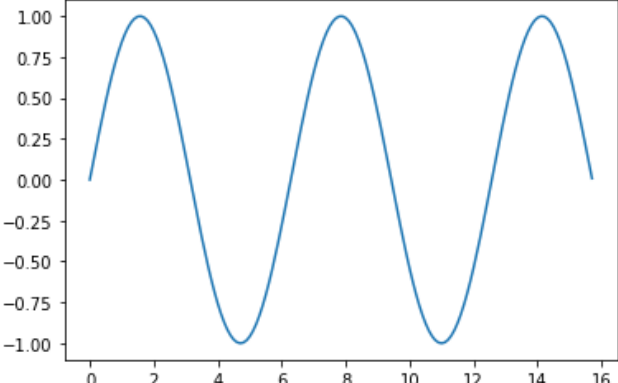
Program	Expected output
<pre>a = [10,20,30,40,50] b = [5,10,15,20,25] c = a + b print( c )</pre>	[10, 20, 30, 40, 50, 5, 10, 15, 20, 25]
<pre>def add_vector(a, b):     c = [ a[i]+b[i] for i in range(len(a)) ]     return c a = [10,20,30,40,50] b = [5,10,15,20,25] c = add_vector(a,b) print( c )</pre>	[15, 30, 45, 60, 75]
<pre>import numpy as np a = np.zeros((4,3)) b = np.identity(3)</pre>	<pre>[[0. 0. 0.]  [0. 0. 0.]  [0. 0. 0.]  [0. 0. 0.]  [1. 0. 0.]  [0. 1. 0.]  [0. 0. 1.]</pre>
<pre>import numpy as np a = np.array([10,20,30]) b = np.array([1,2,3]) c = a + b print(c) type(c)</pre>	<pre>[11 22 33] numpy.ndarray</pre>
<pre>a1 = np.array([1.0, 2.0, 3.0]) a2 = np.array([1, 2, 3], float)</pre>	<pre>[1. 2. 3.] [1. 2. 3.]</pre>

<pre>import numpy as np a = np.array( [ [1,2,3], [10,20,30] ] ) print(a) print(a.shape)</pre>	<pre>[[ 1  2  3]  [10 20 30]] (2, 3)</pre>
<pre>import numpy as np x = np.zeros((2,3)) y = np.ones((3,2)) z1 = np.arange(10) z2 = np.arange(2,10,dtype=np.float) z3 = np.arange(2, 3, 0.1)</pre>	<pre>[[0. 0. 0.]  [0. 0. 0.]  [1. 1.]  [1. 1.]  [1. 1.]  [0 1 2 3 4 5 6 7 8 9]  [2. 3. 4. 5. 6. 7. 8. 9.]  [2. 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8  2.9]</pre>
<pre>import numpy as np x = np.array([[1, 2, 3], [4, 5, 6]], float) y = np.zeros_like(x) y1 = np.ones_like(x) z = np.identity(4, float)</pre>	<pre>[[1. 2. 3.]  [4. 5. 6.]  [0. 0. 0.]  [0. 0. 0.]  [1. 1. 1.]  [1. 1. 1.]  [1. 0. 0. 0.]  [0. 1. 0. 0.]  [0. 0. 1. 0.]  [0. 0. 0. 1.]]</pre>
<pre>a1 = np.array(range(3,7)) a2 = np.array([[1,2],[3,4]]) print(a1) print(a2) print(a1.shape) print(a.size) print(a1.ndim) print(a2.shape) print(a2.size)</pre>	<pre>[3 4 5 6] [[1 2]  [3 4]] (4,) 6 1 (2, 2) 4 2</pre>

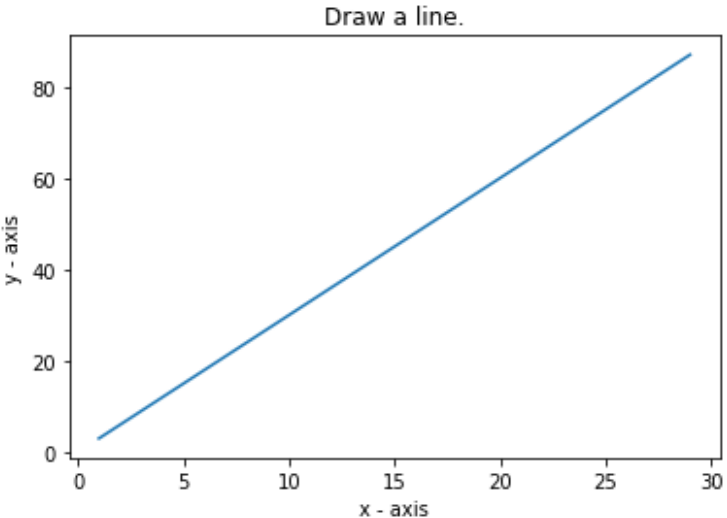
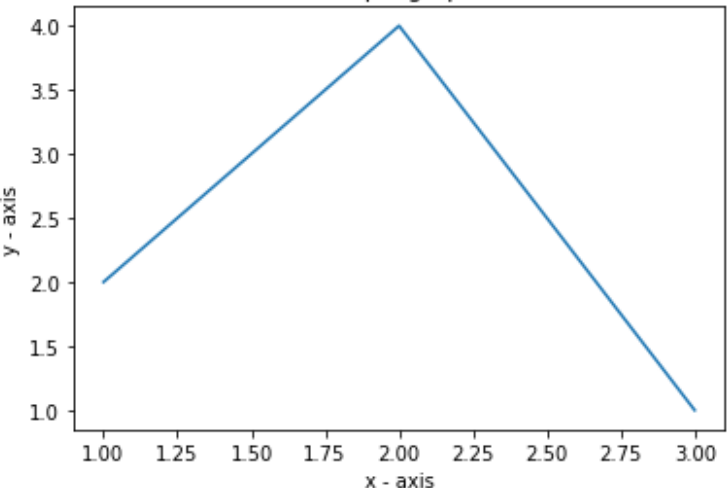
<code>print(a2.ndim)</code>	
<pre> a3 = np.array([[[1,2],[3,4]],[[5,6],[7,8]]] ) a4 = np.array([[1,2],[3,4,5]]) print(a3.dtype) print(a4.dtype) </pre>	<pre> int64 object </pre>
<pre> a5 = np.array([1,2,3,4],dtype='int16') a6 = np.array([1,2,3,4],dtype='float32') print(a6) </pre>	<pre> [1. 2. 3. 4.] </pre>
<pre> a7 = np.array([[1,2],[3.,'4']]) print(a7) print(a7.dtype) </pre>	<pre> [['1' '2']  ['3.0' '4']] &lt;U32 </pre>
<pre> array = np.array([[1,2,3],[4,5,6]]) print(array[0][1]) print(array[1][2]) print(array[0,2]) print(array[1,1]) print(array[1][:]) print(array[1,:]) </pre>	<pre> 2 6 3 5 [4 5 6] [4 5 6] </pre>
<pre> m=np.array([[1, 2, 3], [3, 6, 9], [2, 4, 6]]) print(m) print(m[1, 2]) print(m[1]) print(m[:,1]) print(m[1, 1:3]) </pre>	<pre> [[1 2 3]  [3 6 9]  [2 4 6]] 9 [3 6 9] [2 6 4] [6 9] [[1 3] </pre>

<pre>print(m[:,2,::2]) m[:, 0] = [0, 9, 8] print(m)</pre>	<pre>[2 6] [[0 2 3]  [9 6 9]  [8 4 6]]</pre>
<pre>a2 = np.array([[13,14,15,16],  [17,18,19,20],[21,22,23,24]]) print(a2[1:2,2:3]) print(a2[0:2,1:3]) print(a2[0,1:3]) print(a2[:,2,2]) print(a2[:,:-1,::-1])</pre>	<pre>[[19]] [[14 15]  [18 19]] [14 15] [15 23] [[24 23 22 21]  [20 19 18 17]  [16 15 14 13]]</pre>
<pre>import numpy as np x = np.array([[1,2],[3,4]]) y = np.array([[5,6],[7,8]]) z = x+y z = np.add(x,y) z = x-y z = np.subtract(x,y) z = x*y z = np.multiply(x,y) z = x/y z = np.divide(x,y) z = np.sqrt(x)</pre>	<pre>[[1 2]  [3 4]]  [[5 6]  [7 8]]  [[1.      1.41421356]  [1.73205081 2.      ]]</pre>
<pre>x = np.array([[1,2],[3,4],[5,6]]) u = np.array([2]) + x w = np.array([10,20]) + x v = np.array([[10],[20],[30]]) + x</pre>	<pre>[[1 2]  [3 4]  [5 6]] [[3 4]  [5 6]  [7 8]] [[11 22]  [13 24]  [15 26]] [[11 12]]</pre>

	<pre>[23 24] [35 36]]</pre>
<pre>import numpy as np x = np.array([1,2,3,4]) print( x + 2 ) print( 3 * x ) print( x**2 ) print( x + [2] ) print( x + [1,2,3,4] )</pre>	<pre>[3 4 5 6] [ 3  6  9 12] [ 1  4  9 16] [3 4 5 6] [2 4 6 8]</pre>
<pre>import numpy as np def translation2D(m,dx,dy):     return m + np.array([dx,dy]) m1 = np.array([ [-7,2],[-5,7],[-1,0] ]) m2 = translation2D(m1, 7, -3) print(m2)</pre>	<pre>[[ 0 -1] [ 2  4] [ 6 -3]]</pre>
<pre>from numpy import array a = array([1, 2, 3]) print(a) b = 2 print(b) c = a + b print(c)</pre>	<pre>[1 2 3] 2 [3 4 5]</pre>
<pre>from numpy import array A = array([[1, 2, 3], [1, 2, 3]]) print(A) b = array([1, 2, 3]) print(b) C = A + b Print(C)</pre>	<pre>[[1 2 3] [1 2 3]] [1 2 3] [[2 4 6] [2 4 6]]</pre>

<pre> from numpy import array A = array([[1, 2, 3], [1, 2, 3]]) print(A.shape) b = array([1, 2]) print(b.shape) C = A + b print(C) </pre>	<p>ValueError: operands could not be broadcast together with shapes (2,3) (2,)</p>
<pre> import numpy as np import matplotlib.pyplot as plt x = np.arange(0.0, 5*np.pi, 0.1) y = np.sin(x) plt.plot(x,y) plt.show() </pre>	
<pre> import numpy as np x = np.array([1,2,3]) y = np.array([4,5,6]) z = x.dot(y) z = np.dot(x,y) print(z) </pre>	<p>32</p>
<pre> import numpy as np x = np.array([[1,2,3],[4,5,6]]) y = np.array([[7,8],[9,10],[11,12]]) z = x.dot(y) z = np.dot(x,y) print(z) </pre>	<pre> [[ 58  64]  [139 154]] </pre>

## 2. matplotlib

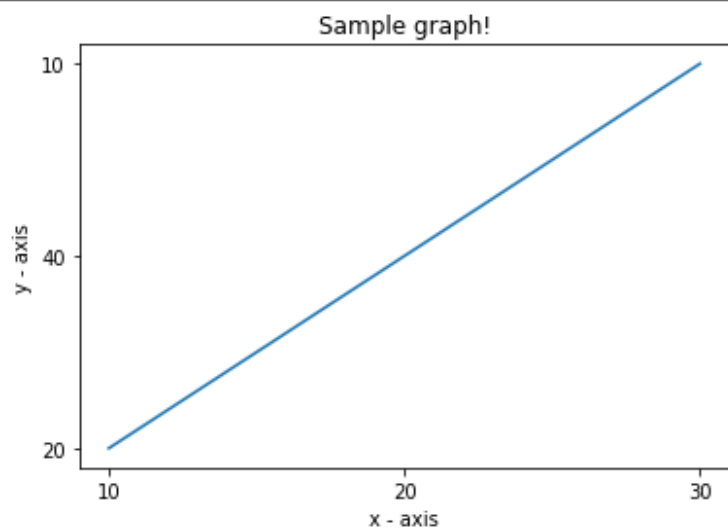
Program	Expected output
<pre>import matplotlib.pyplot as plt X = range(1, 30) Y = [value * 3 for value in X] print("Values of X:") print(*range(1,30)) print("Values of Y :") print(Y) plt.plot(X, Y) plt.xlabel('x - axis') plt.ylabel('y - axis') plt.title('Draw a line.') plt.show()</pre>	<p>Values of X:  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  18 19 20 21 22 23 24 25 26 27 28 29</p> <p>Values of Y :  [3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36,  39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69,  72, 75, 78, 81, 84, 87]</p> 
<pre>import matplotlib.pyplot as plt x = [1,2,3] y = [2,4,1] plt.plot(x, y) plt.xlabel('x - axis') plt.ylabel('y - axis') plt.title('Sample graph!') plt.show()</pre>	

```

import matplotlib.pyplot as plt
with
open("D:/data/test.txt") as f:
    data1 = f.read()
data1 = data1.split('\n')
x = [row.split(' ')[0] for row
in data1]
y = [row.split(' ')[1] for row
in data1]
plt.plot(x, y)
plt.xlabel('x -
axis')
plt.ylabel('y -
axis')
plt.title('Sample graph!')
plt.show()

```

test.txt
10 20
20 40
30 10



```

import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2, 100)
plt.plot(x, x, label='linear')
plt.plot(x, x**2,
label='quadratic')
plt.plot(x, x**3,
label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
plt.show()

```

