الجمهورية اليمنية

جامعة إب

كلية العلوم

قسم علوم الحاسوب وتقنية المعلومات

تكليف مقرر

تنقيب بيانات - عملى

Data Mining

المحاضرة الثانية

عمل الطالب :

أسامة سعيد محمد حمود سعيد

إشراف :

أ. مالك المصنف

2024 - 2025

Numpy

```
import numpy

arr = numpy.array([1,2,3,4,5])
print (arr)
```

```
[1 2 3 4 5]
```

```
import numpy as np

arr = np.array([1,2,3,4,5])
print (arr)
```

```
[1 2 3 4 5]
```

```
print (np.__version__)
```

```
1.26.4
```

```
arr = np.array([1,2,3,4,5])
print (arr)
print (type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```
arr = np.array((1,2,3,4,5))
print (arr)
print (type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```
lst = [1,2,3,4,5,6]
arr = np.array(lst)
print (lst)
print (arr)
```

```
[1, 2, 3, 4, 5, 6]
[1 2 3 4 5 6]
```

```
arr1 = np.array(42)
print ("The Dimension of Array is : ",arr1.ndim)
print (arr1,"\n")

arr2 = np.array([1,2,3,4])
print ("The Dimension of Array is : ",arr2.ndim)
print (arr2,"\n")

arr3 = np.array([[1,2,3],[4,5,6]])
print ("The Dimension of Array is : ",arr3.ndim)
print (arr3,"\n")

arr4 = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
print ("The Dimension of Array is : ",arr4.ndim)
print (arr4,"\n")
```

```
The Dimension of Array is :  0
42

The Dimension of Array is :  1
[1 2 3 4]

The Dimension of Array is :  2
[[1 2 3]
 [4 5 6]]

The Dimension of Array is :  3
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
```

```
arr = np.array([1,2,3,4,5],ndmin=5)
print (arr)
print ("Number of Diemsions : ",arr.ndim)
```

```
[[[[[1 2 3 4 5]]]]]
Number of Diemsions :  5
```

```
lst = [1,2,3,4]
arr = np.array(lst,ndmin=5)
print (arr)
print ("Number of Dimensions : ",arr.ndim)
```

```
[[[[[1 2 3 4]]]]]
Number of Dimensions :  5
```

```
x = np.array([1,2,3])
y = np.array([[0,1,2],[3,4,5]])

print ('Return Number of array Dimensions . ')
print ('x is : ',x.ndim)
print ('y is : ',y.ndim)
```

```
Return Number of array Dimensions .
x is :  1
y is :  2
```

```
x = np.array([1,2,3])
y = np.array([[0,1,2],[3,4,5]])

print ('Return Tuple of array Dimensions . ')
print ('x is : ',x.shape)
print ('y is : ',y.shape)
```

```
Return Tuple of array Dimensions .
x is :  (3,)
y is :  (2, 3)
```

```
x = np.array([1,2,3])
y = np.array([[0,1,2],[3,4,5]])

print ('Return Number of Elements array . ')
print ('x is : ',x.size)
print ('y is : ',y.size)
```

```
Return Number of Elements array .
x is :  3
y is :  6
```

```
x = np.array([1,2,3])
y = np.array([[0,1.5,2],[3,4,5]])
z = np.array([1+3j,2,3])

print ('Return Data-Type of the Array elements .')
print ('x is : ',x.dtype)
print ('y is : ',y.dtype)
print ('z is : ',z.dtype)
```

```
Return Data-Type of the Array elements .
x is :  int32
y is :  float64
z is :  complex128
```

```
x = np.array([1,2,3])
y = np.array([[0,1.5,2],[3,4,5]])
z = np.array([1+3j,2,3])

print ('Return total bytes consumed by the elements of the array .')
print ('x is : ',x.nbytes)
print ('y is : ',y.nbytes)
print ('z is : ',z.nbytes)
```

```
Return total bytes consumed by the elements of the array .
x is :  12
y is :  48
z is :  48
```

```
print ('Create an array filled with a python list elements ')
array_arr = np.array([1,2,3])
end = '\n\n'
print (array_arr,end)
print ('Create an array filled with 0 elements')
zeros_arr = np.zeros(3)
print (zeros_arr,end)

print ('Create an array filled with 1 elements ')
ones_arr = np.ones(3)
print (ones_arr,end)

print ('Create an array filled with random elements ')
empty_arr = np.empty(4)
print (empty_arr,end)

print ('Create an array filled with a range of elements ')
arange_arr1 = np.arange(4)
arange_arr2 = np.arange(1,10,2)
print (arange_arr1,end)
print (arange_arr2,end)

print ('Create an array filled with a spaced linearly values in a specified interval ')
linspace_arr = np.linspace(0,10,num=5)
print (linspace_arr,end)

print ('Create an array filled with fill_value ')
full_arr = np.full((2,2),5)
print (full_arr,end)
```

```
Create an array filled with a python list elements
[1 2 3]


Create an array filled with 0 elements
[0. 0. 0.]


Create an array filled with 1 elements
[1. 1. 1.]


Create an array filled with random elements
[2.12199579e-314 1.07160787e-311 7.50979782e-321 6.95314361e-310]


Create an array filled with a range of elements
[0 1 2 3]


[1 3 5 7 9]


Create an array filled with a spaced linearly values in a specified interval
[ 0.   2.5  5.   7.5 10. ]


Create an array filled with fill_value
[[5 5]
 [5 5]]
```

```
x = np.array([[0,1,2],[3,4,5]])

print ('x : ',x,end)

print ('Return an array of zeros with shape and type of input .')
zeros_arr = np.zeros_like(x)
print (zeros_arr,end)

print ('Return an array of ones with shape and type of input .')
ones_arr = np.ones_like(x)
print (ones_arr,end)

print ('Return an empty array with shape and type of input ')
empty_arr = np.empty_like(x)
print (empty_arr,end)

print ('Return a new array with shape of input filled with value.')
full_arr = np.full_like(x,5)
print (full_arr,end )
```

```
x :  [[0 1 2]
 [3 4 5]]


Return an array of zeros with shape and type of input .
```

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


Return an array of ones with shape and type of input .
[[1 1 1]
 [1 1 1]]


Return an empty array with shape and type of input
[[1 1 1]
 [1 1 1]]


Return a new array with shape of input filled with value.
[[5 5 5]
 [5 5 5]]
```

```
import numpy as np
arr = np.array([2,5,7,8])

print ("The first element is : ",arr[0])
print ("The second element is : ",arr[1])
print ("Addition is : ",arr[0]+arr[1])
print ("Subtraction is : ",arr[0]-arr[1])
print ("Multiplication is : ",arr[0]*arr[1])
print ("Division is : ",arr[0]/arr[1])
print ("Remainder is : ",arr[0]%arr[1])
```

```
The first element is :  2
The second element is :  5
Addition is :  7
Subtraction is :  -3
Multiplication is :  10
Division is :  0.4
Remainder is :  2
```

```
arr = np.array([[1,2,3,4,5],[6,7,8,9,10]])
print ("2nd element on 1st dim : ",arr[0,1])
print ("5th element on 2nd dim : ",arr[1,4])
```

```
2nd element on 1st dim :  2
5th element on 2nd dim :  10
```

```
arr = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
print ("The element is : ",arr[0,1,2])
```

```
The element is :  6
```

```
arr = np.array([[1,2,3,4,5],[6,7,8,9,10]])
print ("Last element from 2nd dim : ",arr[1,-1])
```

```
Last element from 2nd dim :  10
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5])
print(arr[4:])
print(arr[:4])
print(arr[-3:-1])
print(arr[1:5:2])
print(arr[::2])
```

```
[2 3 4 5]
[5 6 7]
[1 2 3 4]
[5 6]
[2 4]
[1 3 5 7]
```

```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4], "\n")

print(arr[0:2, 2], "\n")

print(arr[0:2, 1:4], "\n")
```

```
[7 8 9]
```

```
[3 8]

[[2 3 4]
 [7 8 9]]
```

```
list_object1=[[ 0,  1,  2,  3,  4,  5,  6],
             [14, 15, 16, 17, 18, 19, 20],
             [28, 29, 30, 31, 32, 33, 34]]

array_object1= np.array(list_object1)
print('Return Values are more than 15')
print(array_object1[array_object1>15], " \n")

list_object2=[[[ 0,  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]]]

print("The original Array : ")
array_object2 = np.array(list_object2)
print(array_object2, "\n\n")

b = np.array([[True, True, False], [False, True, True]])
print('Return Values based on boolean values')
print(array_object2[b])
```

```
Return Values are more than 15
[16 17 18 19 20 28 29 30 31 32 33 34]

The original Array :
[[[ 0  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]]]


Return Values based on boolean values
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [20 21 22 23 24]
 [25 26 27 28 29]]
```

```
import numpy as np

arr1 = np.array([1, 2, 3, 4], dtype='S')
print(arr1)
print(arr1.dtype)

arr2 = np.array([1, 2, 3, 4], dtype='i4')
print(arr2)
print(arr2.dtype)
```

```
[b'1' b'2' b'3' b'4']
|S1
[1 2 3 4]
int32
```

```
arr = np.array(['a', '2', '3'], dtype='i')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[2], line 1
----> 1 arr = np.array(['a', '2', '3'], dtype='i')

ValueError: invalid literal for int() with base 10: 'a'
```

```
arr1 = np.array([1.1,2.1,3.1])

newarr1 = arr1.astype('i')
print (newarr1)
print (newarr1.dtype)

arr2 = np.array([1,0,3])
newarr2 = arr2.astype(bool)
```

```
print (newarr2)
print (newarr2.dtype)
```

```
[1 2 3]
int32
[ True False  True]
bool
```

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42

print(arr)
print(x)
```

```
[42  2  3  4  5]
[1 2 3 4 5]
```

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
x[0] = 31

print(arr)
print(x)
```

```
[31  2  3  4  5]
[31  2  3  4  5]
```

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 31

print(arr)
print(x)
```

```
[31  2  3  4  5]
[31  2  3  4  5]
```

```
arr = np.array([1, 2, 3, 4, 5])

x = arr.copy()
y = arr.view()

print(x.base)
print(y.base)
```

```
None
[1 2 3 4 5]
```

```
arr1 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print('shape of array :', arr1.shape)

arr2 = np.array([1, 2, 3, 4], ndmin=5)
print('shape of array :', arr2.shape)
```

```
shape of array : (2, 4)
shape of array : (1, 1, 1, 1, 4)
```

```
x = np.array([[2,3,4], [5,6,7]])
y= np.reshape(x, (3, 2))

print('Original Array')
print(x)

print('\n Reshaped Array')
print(y)
```

```
Original Array
[[2 3 4]
 [5 6 7]]

 Reshaped Array
[[2 3]
 [4 5]
 [6 7]]
```

```
print("Reshape From 1-D to 2-D")
arr1 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr1 = arr1.reshape(4, 3)
print(newarr1)
print("\nReshape From 1-D to 3-D")
arr2 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr2 = arr2.reshape(2, 3, 2)
print(newarr2)
```

```
Reshape From 1-D to 2-D
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]

Reshape From 1-D to 3-D
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
newarr = arr.reshape(3, 3)
print(newarr)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[15], line 2
      1 arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
----> 2 newarr = arr.reshape(3, 3)
      3 print(newarr)

ValueError: cannot reshape array of size 8 into shape (3,3)
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
newarr=arr.reshape(2, 4)
print(newarr.base)
```

```
[1 2 3 4 5 6 7 8]
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
newarr = arr.reshape(2, 2, -1)
print(newarr)
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
print(arr.reshape(2, 4).base)
```

```
[1 2 3 4 5 6 7 8]
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
newarr = arr.reshape(2, 2, -1)
print(newarr)
```

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
print (arr)
newarr = arr.reshape(-1)
print(newarr)
```

```
[[1 2 3]
 [4 5 6]]
[1 2 3 4 5 6]
```

```
arr = np.array([1, 2, 3])
for x in arr:
    print(x)
```

```
1
2
3
```

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
    print(x)
```

```
[1 2 3]
[4 5 6]
```

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
    for y in x:
        print(y)
```

```
1
2
3
4
5
6
```

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
for x in arr:
     print(x)
```

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

```
arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
for x in arr:
    for y in x:
        print(y)
```

```
[1 2]
[3 4]
[5 6]
[7 8]
```

```
arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
for x in arr:
    for y in x:
        for z in y:
            print(z)
```

```
1
2
3
4
5
6
7
8
```

```
arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
for x in arr: #np.nditer(arr):
    print(x)
```

```
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]
```

```
arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
for x in np.nditer(arr):
    print(x)
```

```
1
2
3
4
5
6
7
8
```

```
arr = np.array([1, 2, 3])
for x in np.nditer(arr, flags=['buffered'], op_dtypes=['S']):
    print(x)
```

```
b'1'
b'2'
b'3'
```

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
for x in np.nditer(arr[:, ::2]):
    print(x)
```

```
1
3
5
7
```

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
for x in np.nditer(arr[:, 1::2]):
    print(x)
```

```
2
4
6
8
```

```
print("Enumerate on following 1D arrays elements")
arr1 = np.array([1, 2, 3])
for idx, x in np.ndenumerate(arr1):
    print(idx, x)

print("Enumerate on following 2D array's elements")
arr2 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
for idx, x in np.ndenumerate(arr2):
    print(idx, x)
```

```
Enumerate on following 1D arrays elements
(0,) 1
(1,) 2
(2,) 3
Enumerate on following 2D array's elements
(0, 0) 1
(0, 1) 2
(0, 2) 3
(0, 3) 4
(1, 0) 5
(1, 1) 6
(1, 2) 7
(1, 3) 8
```

```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
print('Original array is:')
print(arr,'\n')

asc_sorted_arr=np.sort(arr)
print('Sorting array with ascending order is:')
print(asc_sorted_arr,'\n')

desc_sorted_arr=np.sort(-arr)
print('Sorting array with descending order is:')
print(desc_sorted_arr)
```

```
Original array is:
[2 1 5 3 7 4 6 8]

Sorting array with ascending order is:
[1 2 3 4 5 6 7 8]

Sorting array with descending order is:
[-8 -7 -6 -5 -4 -3 -2 -1]
```

```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
print('Original array is:')
print(arr,'\n')

asc_sorted_arr=np.sort(arr)
print('Sorting array with ascending order is:')
print(asc_sorted_arr,'\n')

desc_sorted_arr=-np.sort(arr)
print('Sorting array with descending order is:')
print(desc_sorted_arr)
```

```
Original array is:
[2 1 5 3 7 4 6 8]

Sorting array with ascending order is:
[1 2 3 4 5 6 7 8]

Sorting array with descending order is:
[-1 -2 -3 -4 -5 -6 -7 -8]
```

```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
print('Original array is:')
print(arr,'\n')

asc_sorted_arr=np.sort(arr)
print('Sorting array with ascending order is:')
print(asc_sorted_arr,'\n')

desc_sorted_arr=-np.sort(-arr)
print('Sorting array with descending order is:')
print(desc_sorted_arr)
```

```
Original array is:
[2 1 5 3 7 4 6 8]

Sorting array with ascending order is:
[1 2 3 4 5 6 7 8]

Sorting array with descending order is:
[8 7 6 5 4 3 2 1]
```

```
print("Sort the array numerically")
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))

print("\n Sort the array alphabetically")
arr = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr))

print("\n Sort a boolean array")
arr = np.array([True, False, True])
print(np.sort(arr))

print("\n Sort a 2-D array")
arr = np.array([[3, 2, 4], [5, 0, 1]])
print(np.sort(arr))
```

```
Sort the array numerically
[0 1 2 3]

 Sort the array alphabetically
['apple' 'banana' 'cherry']

 Sort a boolean array
[False  True  True]

 Sort a 2-D array
[[2 3 4]
 [0 1 5]]
```

```
print("Find the indexes where the value 7 should be inserted:")
arr = np.array([6, 7, 8, 9])
x = np.searchsorted(arr, 7)
print("Index is : " , x)
```

```
Find the indexes where the value 7 should be inserted:
Index is :  1
```

```
print("Find the indexes where the value 7 should be inserted, starting from the right")
arr = np.array([6, 7, 8, 9])
x = np.searchsorted(arr, 7, side='right')
print("Index is : " , x)
```

```
Find the indexes where the value 7 should be inserted, starting from the right
Index is :  2
```

```
print("Find the indexes where the values 2, 4, and 6 should be inserted")
arr = np.array([1, 3, 5, 7])
x = np.searchsorted(arr, [2, 4, 6])
print(x)
```

```
Find the indexes where the values 2, 4, and 6 should be inserted
[1 2 3]
```

```
print("Find the indexes where the value is 4:")
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
print(x)

print("\n Find the indexes where the values are even")
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 0)
print(x)

print("\n Find the indexes where the values are odd")
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 1)
print(x)
```

```
Find the indexes where the value is 4:
(array([3, 5, 6], dtype=int64),)

 Find the indexes where the values are even
(array([1, 3, 5, 7], dtype=int64),)

 Find the indexes where the values are odd
(array([0, 2, 4, 6], dtype=int64),)
```

```
print("Join two 1-D arrays along rows (axis=0) ")
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)

print("\nJoin two 2-D arrays along rows (axis=1)")
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])
arr = np.concatenate((arr1, arr2), axis=1)
print(arr)
```

```
Join two 1-D arrays along rows (axis=0)
[1 2 3 4 5 6]

Join two 2-D arrays along rows (axis=1)
[[1 2 5 6]
 [3 4 7 8]]
```

```
print("stack along axis")
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.stack((arr1, arr2), axis=1)
print(arr)
```

```
stack along axis
[[1 4]
 [2 5]
 [3 6]]
```

```
print("stack along rows")
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.hstack((arr1, arr2))
print(arr)
```

```
stack along rows
[1 2 3 4 5 6]
```

```
print("stack along columns")
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.vstack((arr1, arr2))
print(arr)
```

```
stack along columns
[[1 2 3]
 [4 5 6]]
```

```
print("stack along height")
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.dstack((arr1, arr2))
print(arr)
```

```
stack along height
[[[1 4]
  [2 5]
  [3 6]]]
```

```
print("Split the array in 3 parts")
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
```

```
print(newarr)

print("\nSplit the array in 4 parts")
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 4)
print(newarr)

print("\nSplit the array in 6 parts")
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 6)
print(newarr)

print("\nSplit the array in 8 parts")
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 8)
print(newarr)
```

```
Split the array in 3 parts
[array([1, 2]), array([3, 4]), array([5, 6])]

Split the array in 4 parts
[array([1, 2]), array([3, 4]), array([5]), array([6])]

Split the array in 6 parts
[array([1]), array([2]), array([3]), array([4]), array([5]), array([6])]

Split the array in 8 parts
[array([1]), array([2]), array([3]), array([4]), array([5]), array([6]), array([], dtype=int32), array([], dtype=int32)]
```

```
print("Split the array in 3 parts and Access the splitted arrays:")
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
print(newarr[0])
print(newarr[1])
print(newarr[2])
```

```
Split the array in 3 parts and Access the splitted arrays:
[1 2]
[3 4]
[5 6]
```

```
print("Split the 2-D array into three 2-D arrays")
arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])
newarr = np.array_split(arr, 3)
print(newarr)

print("\nSplit the 2-D array into three 2-D arrays")
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.array_split(arr, 3)
print(newarr)
```

```
Split the 2-D array into three 2-D arrays
[array([[1, 2],
       [3, 4]]), array([[5, 6],
       [7, 8]]), array([[ 9, 10],
       [11, 12]])]

Split the 2-D array into three 2-D arrays
[array([[1, 2, 3],
       [4, 5, 6]]), array([[ 7,  8,  9],
       [10, 11, 12]]), array([[13, 14, 15],
       [16, 17, 18]])]
```

```
print("Split the 2-D array into three 2-D arrays along rows.")
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.array_split(arr, 3, axis=1)
print(newarr)
```

```
Split the 2-D array into three 2-D arrays along rows.
[array([[ 1],
       [ 4],
       [ 7],
       [10],
       [13],
       [16]]), array([[ 2],
       [ 5],
       [ 8],
       [11],
       [14],
       [17]]), array([[ 3],
       [ 6],
       [ 9],
       [12],
       [15],
        [18]])]
```

```
print("split the 2-D array into three 2-D arrays along rows")
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.vsplit(arr, 3)
print(newarr)
```

```
split the 2-D array into three 2-D arrays along rows
[array([[1, 2, 3],
       [4, 5, 6]]), array([[ 7,  8,  9],
       [10, 11, 12]]), array([[13, 14, 15],
       [16, 17, 18]])]
```

```
print("split the 2-D array into three 2-D arrays along rows")
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.hsplit(arr, 3)
print(newarr)
```

```
split the 2-D array into three 2-D arrays along rows
[array([[ 1],
       [ 4],
       [ 7],
       [10],
       [13],
       [16]]), array([[ 2],
       [ 5],
       [ 8],
       [11],
       [14],
       [17]]), array([[ 3],
       [ 6],
       [ 9],
       [12],
       [15],
       [18]])]
```

```
print("split the 2-D array into three 2-D arrays along rows")
arr = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]]])
newarr = np.dsplit(arr, 3)
print(newarr)
```

```
split the 2-D array into three 2-D arrays along rows
[array([[[ 1],
        [ 4],
        [ 7],
        [10],
        [13],
        [16]]]), array([[[ 2],
        [ 5],
        [ 8],
        [11],
        [14],
        [17]]]), array([[[ 3],
        [ 6],
        [ 9],
        [12],
        [15],
        [18]]])]
```

```
print("Create an array from the elements on index 0 and 2")
arr = np.array([41, 42, 43, 44])
x = [True, False, True, False]
newarr = arr[x]
print(newarr)
```

```
Create an array from the elements on index 0 and 2
[41 43]
```

```
print("Create a filter array that will return only values higher than 42")
arr = np.array([41, 42, 43, 44])

filter_arr = []

for element in arr:
    if element > 42:
        filter_arr.append(True)
    else:
        filter_arr.append(False)

newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```

```
Create a filter array that will return only values higher than 42
[False, False, True, True]
[43 44]
```

```
print("Create a filter array that will return only even elements from the original array")
arr = np.array([1, 2, 3, 4, 5, 6, 7])

filter_arr = []

for element in arr:
    if element % 2 == 0:
        filter_arr.append(True)
    else:
        filter_arr.append(False)

newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```

```
Create a filter array that will return only even elements from the original array
[False, True, False, True, False, True, False]
[2 4 6]
```

```
print("Create a filter array that will return only values higher than 42")
arr = np.array([41, 42, 43, 44])
filter_arr = arr > 42

newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```

```
Create a filter array that will return only values higher than 42
[False False  True  True]
[43 44]
```

```
print("Create a filter array that will return only even elements from the original array")
arr = np.array([1, 2, 3, 4, 5, 6, 7])
filter_arr = arr % 2 == 0

newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```

```
Create a filter array that will return only even elements from the original array
[False  True False  True False  True False]
[2 4 6]
```

```
arr = np.array([[0,0], [1,1], [2,2]])
a= np.insert(arr, 2, 4)
b= np.insert(arr, 2, 4, axis=1)

print("Original Array")
print(arr , "\n")

print("Return a copy of flatted array If axis is None")
print(a , "\n")

print("Return a copy of array with values inserted")
print(b , "\n")
```

```
Original Array
[[0 0]
 [1 1]
 [2 2]]

Return a copy of flatted array If axis is None
[0 0 4 1 1 2 2]

Return a copy of array with values inserted
[[0 0 4]
 [1 1 4]
 [2 2 4]]
```

```
import numpy as np

arr = np.array([[0,0], [1,1], [2,2]])
a= np.insert(arr, [1], [[3], [4], [5]], axis=0)
b= np.insert(arr, [1], [[3], [4], [5]], axis=1)

print("Original Array")
print(arr , "\n")

print("Return a copy of array with values inserted along axis = 0")
print(a , "\n")

print("Return a copy of array with values inserted along axis = 1")
print(b , "\n")
```

```
Original Array
[[0 0]
 [1 1]
 [2 2]]

Return a copy of array with values inserted along axis = 0
[[0 0]
 [3 3]
 [4 4]
 [5 5]
 [1 1]
 [2 2]]

Return a copy of array with values inserted along axis = 1
[[0 3 0]
 [1 4 1]
 [2 5 2]]
```

```
arr = np.array([[0,0], [1,1], [2,2]])
a= np.insert(arr, 1, [3, 4, 5], axis=0)
b= np.insert(arr, 1, [3, 4, 5], axis=1)

print("Original Array")
print(arr , "\n")

print("Return a copy of array with values inserted along axis = 0")
print(a , "\n")

print("Return a copy of array with values inserted along axis = 1")
print(b , "\n")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[3], line 2
      1 arr = np.array([[0,0], [1,1], [2,2]])
----> 2 a= np.insert(arr, 1, [3, 4, 5], axis=0)
      3 b= np.insert(arr, 1, [3, 4, 5], axis=1)
      5 print("Original Array")

File C:\ProgramData\anaconda3\Lib\site-packages\numpy\lib\function_base.py:5525, in insert(arr, obj, values, axis)
   5523 new[tuple(slobj)] = arr[tuple(slobj)]
   5524 slobj[axis] = slice(index, index+numnew)
-> 5525 new[tuple(slobj)] = values
   5526 slobj[axis] = slice(index+numnew, None)
   5527 slobj2 = [slice(None)] * ndim

ValueError: could not broadcast input array from shape (1,3) into shape (1,2)
```

```
arr = np.arange(12).reshape(3,4)
id_arr = (1, 3)
x = np.insert(arr, id_arr, 777, axis=1)

print("Original Array")
print(arr , "\n")

print("Return a copy of array with values inserted along axis = 1")
print(x , "\n")
```

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

Return a copy of array with values inserted along axis = 1
[[  0 777   1   2 777   3]
 [  4 777   5   6 777   7]
 [  8 777   9  10 777  11]]
```

```
x =  np.array([0, 1, 2])
y = np.array([[3, 4, 5], [6, 7, 8]])
z = np.append(x, y )

print("The First Array")
print(x , "\n")

print("The Second Array")
print(y , "\n")

print("Return appening Arrays")
print(z , "\n")
```

```
The First Array
[0 1 2]

The Second Array
[[3 4 5]
 [6 7 8]]

Return appening Arrays
[0 1 2 3 4 5 6 7 8]
```

```
z = np.append ([0, 1, 2], [[3, 4, 5], [6, 7, 8]])

print("Return appening Arrays")
print(z , "\n")
```

```
Return appening Arrays
[0 1 2 3 4 5 6 7 8]
```

```
z = np.append([[0, 1, 2], [3, 4, 5]],[[6, 7, 8]], axis=0)

print("Return appening Arrays")
print(z , "\n")
```

```
Return appening Arrays
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

```
arr = np.array([[0,1,2], [4,5,6], [7,8,9]])
x=np.delete(arr, 1, 0)

print("The Original Array")
print(arr , "\n")

print("Return a copy of arr with the elements specified by obj removed ")
print(x , "\n")
```

```
The Original Array
[[0 1 2]
 [4 5 6]
 [7 8 9]]

Return a copy of arr with the elements specified by obj removed
[[0 1 2]
 [7 8 9]]
```

```
a = np.array([1.0, 2.0, 3.0])
b = np.array([2.0, 2.0, 2.0])

print(a*b)
```

```
[2. 4. 6.]
```

```
a=np.array([1, 2, 3, 4])
b=np.array([5, 6, 7, 8])

print("Addition of numbers in Arrays by broadcasting","\n" , a+b, "\n")
print("Subtraction of numbers in Arrays by broadcasting","\n" , a-b, "\n")
print("Multiplication  of numbers in Arrays by broadcasting","\n" , a*b, "\n")
print("Division of numbers in Arrays by broadcasting","\n" , a/b, "\n")
```

```
Addition of numbers in Arrays by broadcasting
 [ 6  8 10 12]

Subtraction of numbers in Arrays by broadcasting
 [-4 -4 -4 -4]

Multiplication  of numbers in Arrays by broadcasting
 [ 5 12 21 32]
```

```
Division of numbers in Arrays by broadcasting
 [0.2        0.33333333 0.42857143 0.5       ]
```

```
data=np.array([1, 2, 3, 4])

print("Addition of numbers in Arrays by broadcasting","\n" , data+5, "\n")
print("Subtraction of numbers in Arrays by broadcasting","\n" , data-5, "\n")
print("Multiplication  of numbers in Arrays by broadcasting","\n" , data*5, "\n")
print("Division of numbers in Arrays by broadcasting","\n" , data/5, "\n")
```

```
Addition of numbers in Arrays by broadcasting
 [6 7 8 9]

Subtraction of numbers in Arrays by broadcasting
 [-4 -3 -2 -1]

Multiplication  of numbers in Arrays by broadcasting
 [ 5 10 15 20]

Division of numbers in Arrays by broadcasting
 [0.2 0.4 0.6 0.8]
```

```
data=np.array([1, 2, 3, 4])

print("the max of elements is : ", data.max(), "\n")

print("the min of elements is : ", data.min(), "\n")

print("the sum of elements is : ", data.sum(), "\n")

print("the mean of elements is : ", data.mean(), "\n")

print("the standard deviation of elements is : ", data.std(), "\n")
```

```
the max of elements is :  4

the min of elements is :  1

the sum of elements is :  10

the mean of elements is :  2.5

the standard deviation of elements is :  1.118033988749895
```

```
from numpy import random
x = random.randint(100)
print(x)
```

```
60
```

```
x = random.rand()
print(x)
```

```
0.49353558522603025
```

```
print("Generate a 1-D array containing 5 random integers from 0 to 100:")
int_x_1D=random.randint(100, size=(5))
print(int_x_1D)
print("\n")

print("Generate a 2D array with 3 rows, each row is 5 random integers from 0 to 100")
int_x_2D = random.randint(100, size=(3, 5))
print(int_x_2D)
print("\n")

print("Generate a 1-D array containing 5 random floats:")
float_x_1D = random.rand(5)
print(float_x_1D)
print("\n")

print("Generate a 2-D array with 3 rows, each row containing 5 random numbers:")
float_x_2D = random.rand(3, 5)
print(float_x_2D)
```

```
Generate a 1-D array containing 5 random integers from 0 to 100:
[52 24 62 55 22]


Generate a 2D array with 3 rows, each row is 5 random integers from 0 to 100
```

```
[[11 95 10 52  4]
 [50 48 91 17 88]
 [96 55 57 30 24]]


Generate a 1-D array containing 5 random floats:
[0.95470596 0.06675996 0.81909604 0.10506703 0.55365115]


Generate a 2-D array with 3 rows, each row containing 5 random numbers:
[[0.06166986 0.25744071 0.40975643 0.07581912 0.98564058]
 [0.65546071 0.29138793 0.01494116 0.54947621 0.73112972]
 [0.76074566 0.06281352 0.87437252 0.8655471  0.77704347]]
```

```
print("Return one of the values in an array:")
x = random.choice([3, 5, 7, 9])
print(x)
print("\n")

print("Generate a 2D arrayconsists of the values in the array parameter (3,5,7,and 9):")
y = random.choice([3, 5, 7, 9], size=(3, 5))
print(y)
```

```
Return one of the values in an array:
7


Generate a 2D arrayconsists of the values in the array parameter (3,5,7,and 9):
[[5 3 5 3 3]
 [7 9 7 5 9]
 [3 9 3 3 5]]
```

```
print("Generate a 1D array has 100 values, where each value has to be 3, 5, 7 or 9.")
x = random.choice([3, 5, 7, 9], p=[0.1, 0.3, 0.6, 0.0], size=(100))
print(x)
```

```
Generate a 1D array has 100 values, where each value has to be 3, 5, 7 or 9.
[5 7 7 7 7 7 7 5 7 7 3 7 7 7 7 7 7 7 7 3 7 7 5 5 7 7 7 7 7 7 7 7 3 7 5 7 7
 7 5 7 7 7 7 7 3 5 3 7 7 3 7 7 5 7 7 7 5 7 7 7 7 7 7 3 7 7 7 7 7 7 7 7 5 7
 5 5 3 7 7 7 7 7 5 3 5 5 5 7 7 7 7 7 3 7 7 7 7 7 7 7]
```

```
print("Same example as above, but return a 2D array with 3 rows, each row has 5 values.")
x = random.choice([3, 5, 7, 9], p=[0.1, 0.3, 0.6, 0.0], size=(3, 5))
print(x)
```

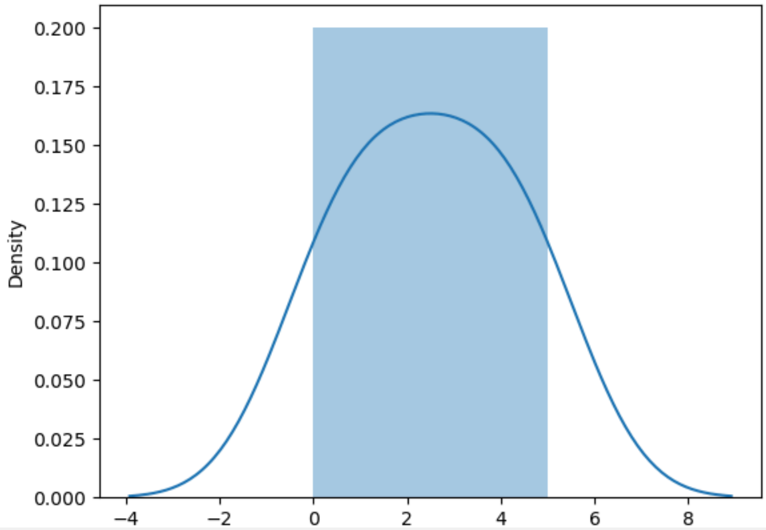
```
Same example as above, but return a 2D array with 3 rows, each row has 5 values.
[[7 7 7 7 5]
 [7 5 7 7 7]
 [7 3 5 7 5]]
```

```
print("Randomly shuffle elements of following array:")
arr1 = np.array([1, 2, 3, 4, 5])
arrshuf=random.shuffle(arr1)
print("shuffle array:", arrshuf)
print("original array:", arr1 , "\n")

print("Randomly shuffle elements of following array:")
arr2 = np.array([1, 2, 3, 4, 5])
print("permutation array:", random.permutation(arr2))
print("original array:", arr2)
```

```
Randomly shuffle elements of following array:
shuffle array: None
original array: [3 4 5 2 1]

Randomly shuffle elements of following array:
permutation array: [1 4 3 5 2]
original array: [1 2 3 4 5]
```

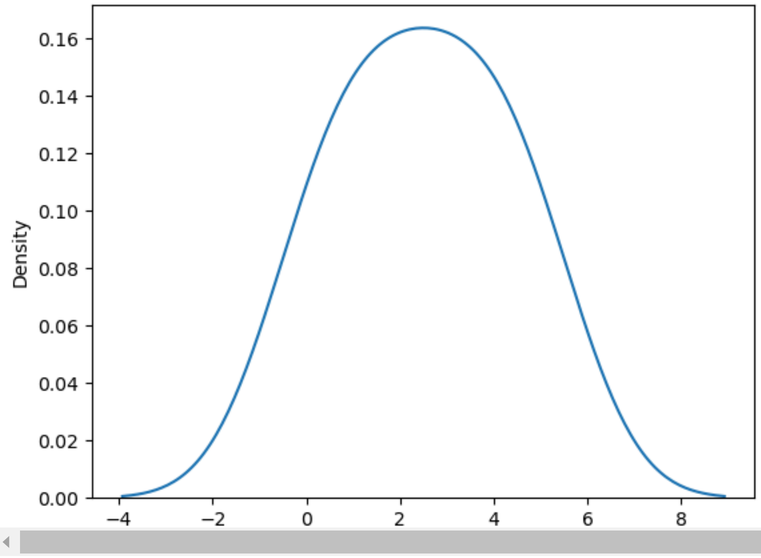
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot([0, 1, 2, 3, 4, 5], hist=True)
plt.show()
```

```
C:\Users\hpp\AppData\Local\Temp\ipykernel_6068\961196836.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot([0, 1, 2, 3, 4, 5], hist=True)
```



```
sns.distplot([0, 1, 2, 3, 4, 5], hist=False)
plt.show()
```

```
C:\Users\hpp\AppData\Local\Temp\ipykernel_6068\4080913807.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot([0, 1, 2, 3, 4, 5], hist=False)
```



```
arr = np.array([1, 2, 3, 4, 5, 6])

np.save('filename', arr)


x= np.load('filename.npy')

print(x)
```

```
[1 2 3 4 5 6]
```

```
csv_arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
np.savetxt('new_file.csv', csv_arr)
```

```
x=np.loadtxt('new_file.csv')
print(x)
```

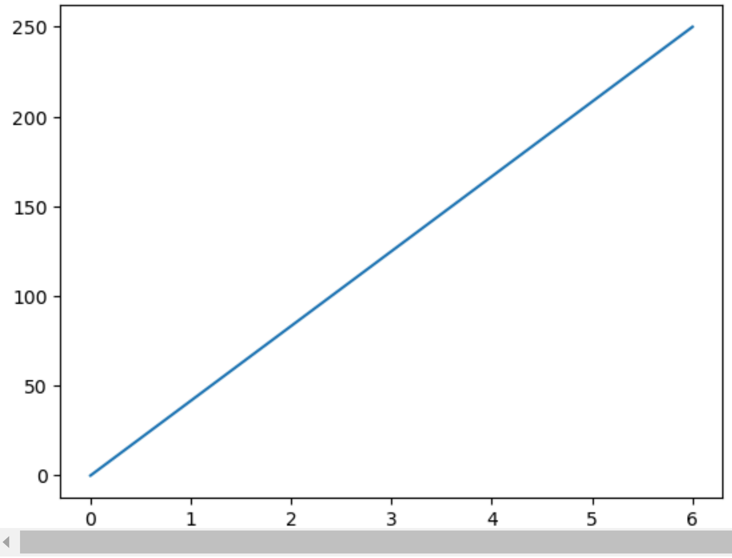
```
[1. 2. 3. 4. 5. 6. 7. 8.]
```

Matplotlib

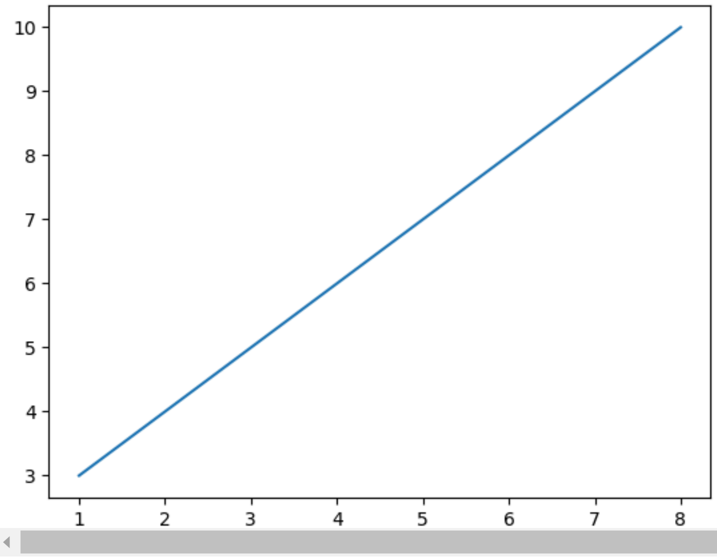
```
import matplotlib

print (matplotlib.__version__)
```

```
3.8.4
```

```
import matplotlib.pyplot as plt
import numpy as np

xpoint = np.array([0,6])
ypoint = np.array([0,250])

plt.plot(xpoint,ypoint)
plt.show()
```

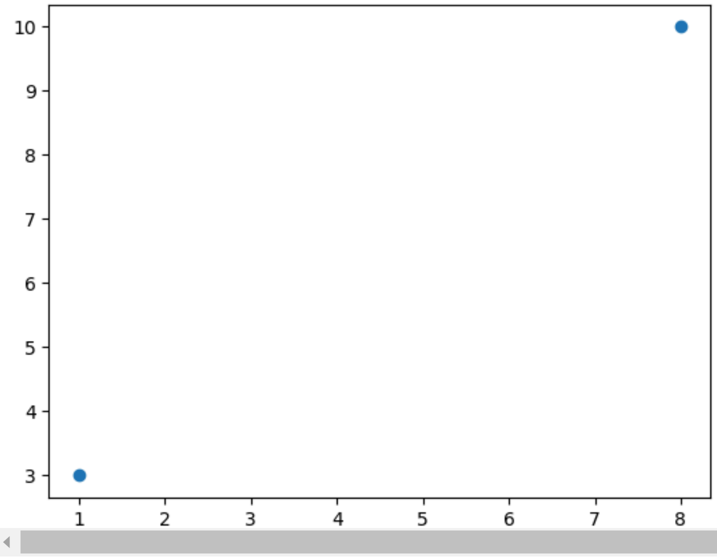


```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

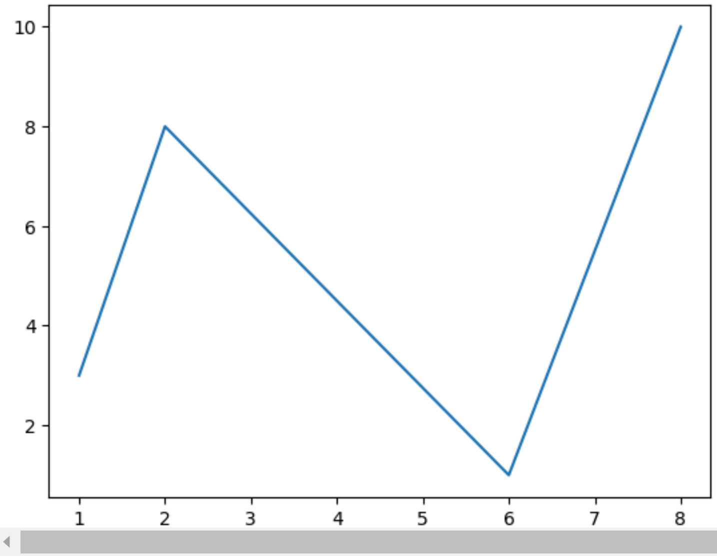


```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
```

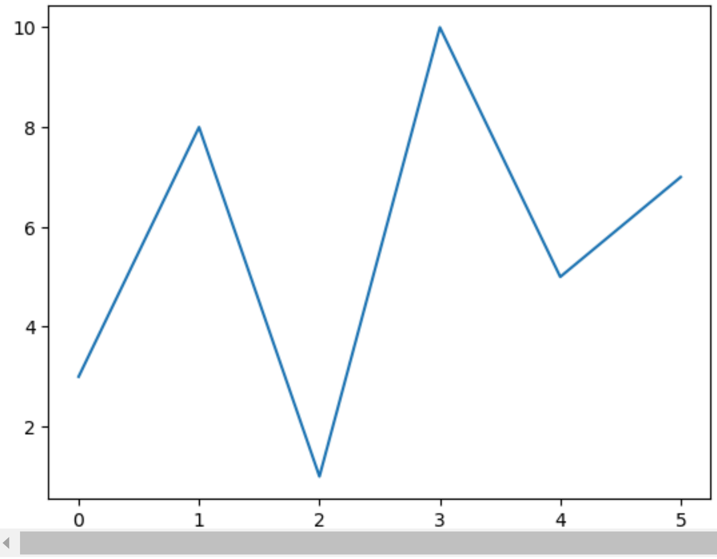
```
plt.plot(xpoints, ypoints, 'o')
plt.show()
```



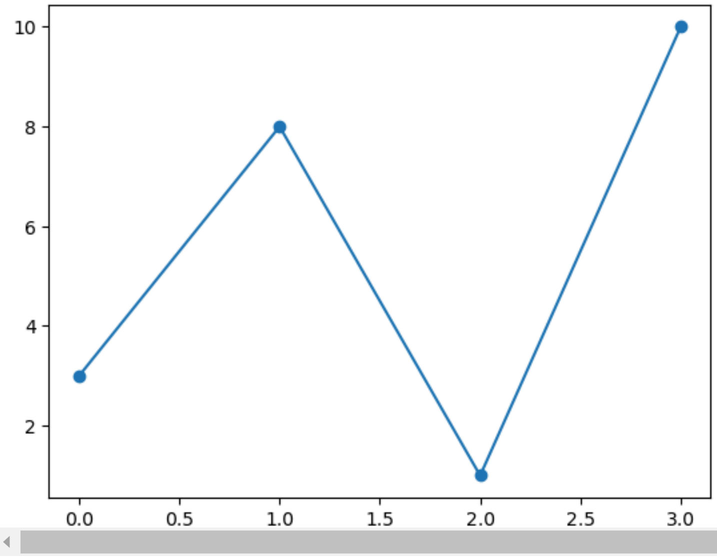
```
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```



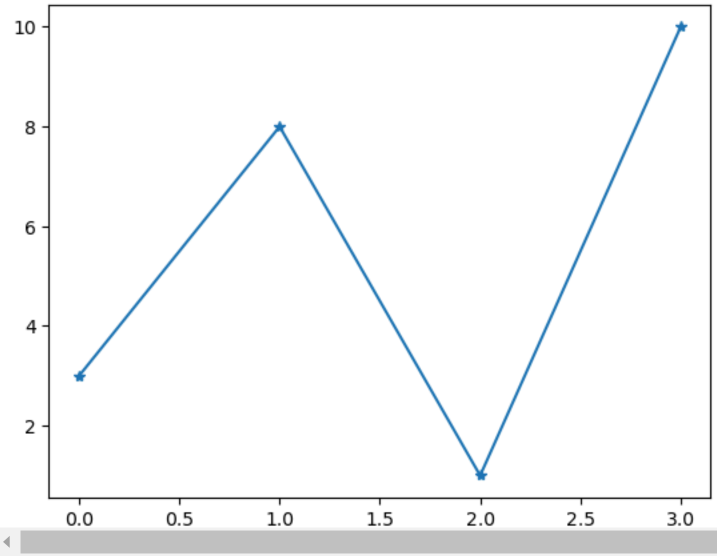
```
ypoints = np.array([3, 8, 1, 10, 5, 7])
plt.plot(ypoints)
plt.show()
```



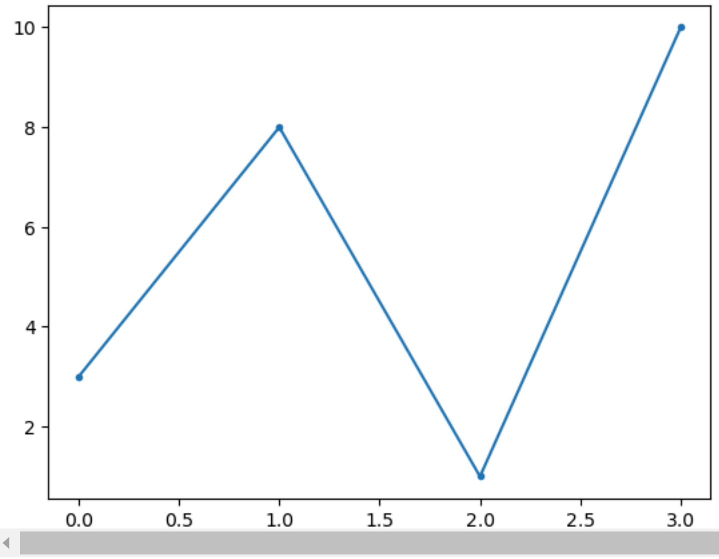
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'o')
plt.show()
```



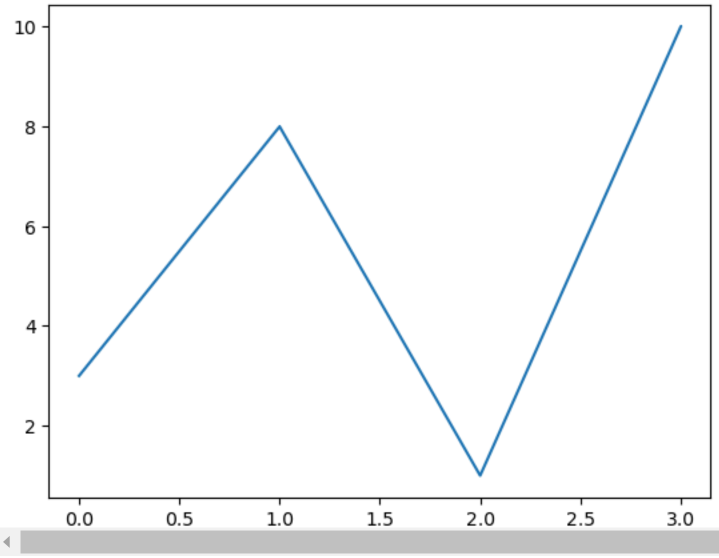
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '*')
plt.show()
```



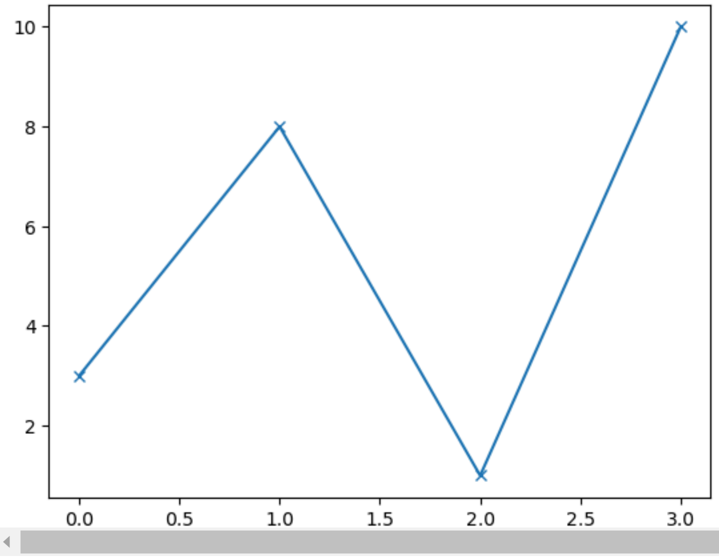
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '.')
plt.show()
```



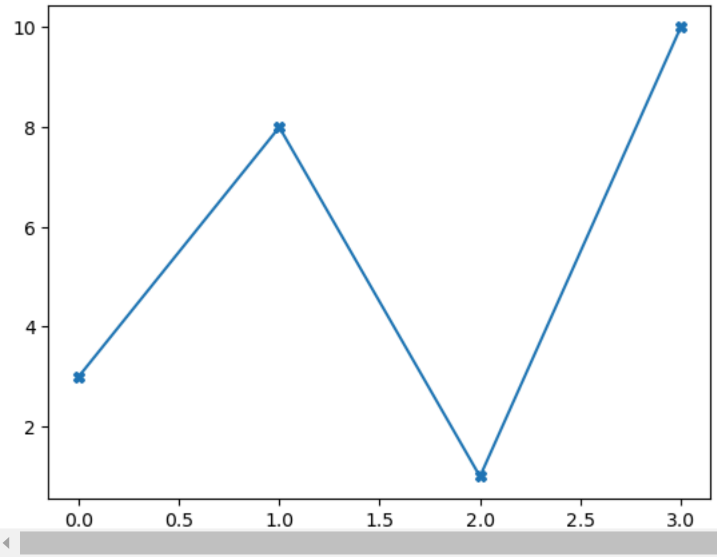
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = ',')
plt.show()
```



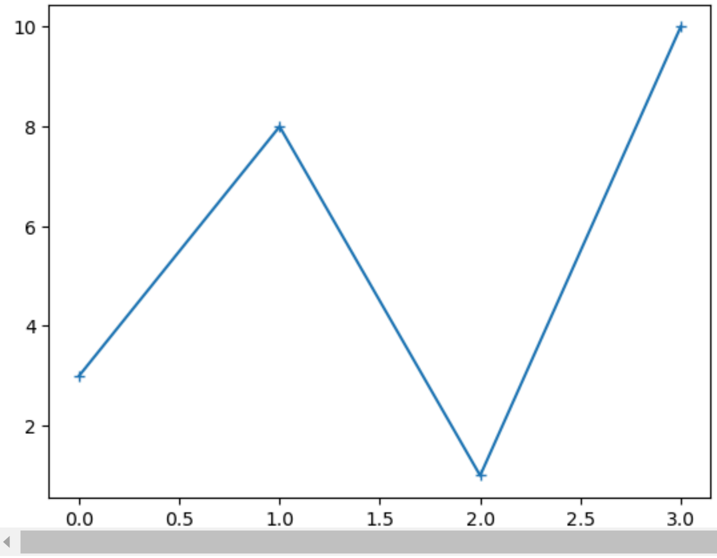
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'x')
plt.show()
```



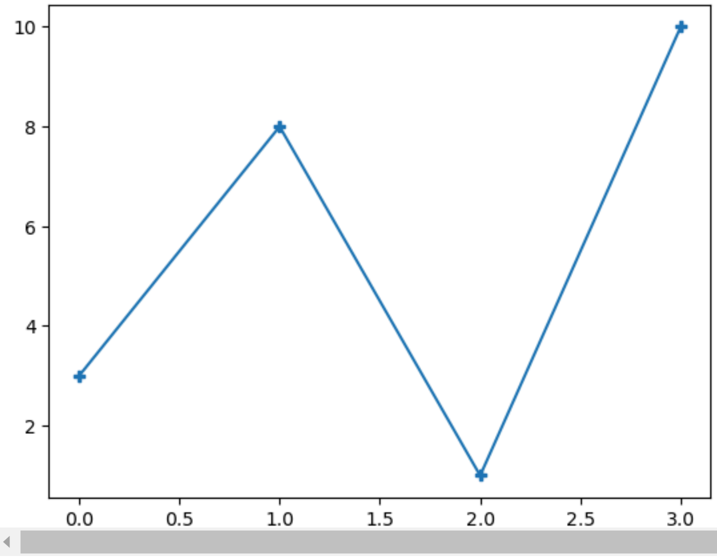
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'X')
plt.show()
```



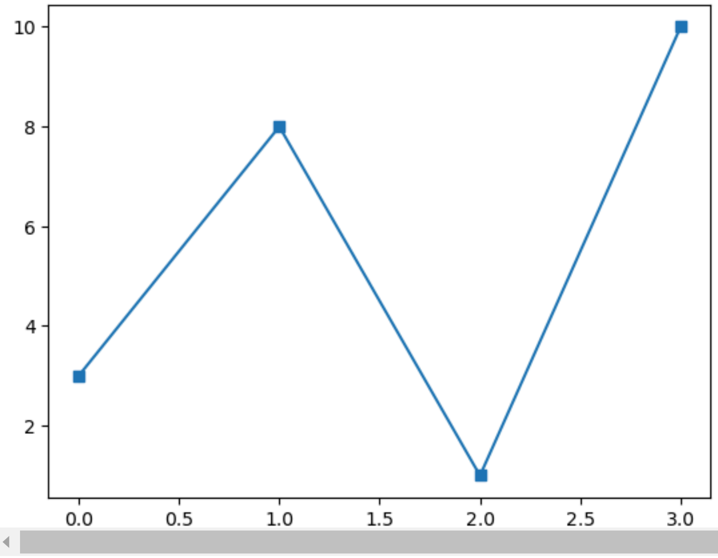
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '+')
plt.show()
```



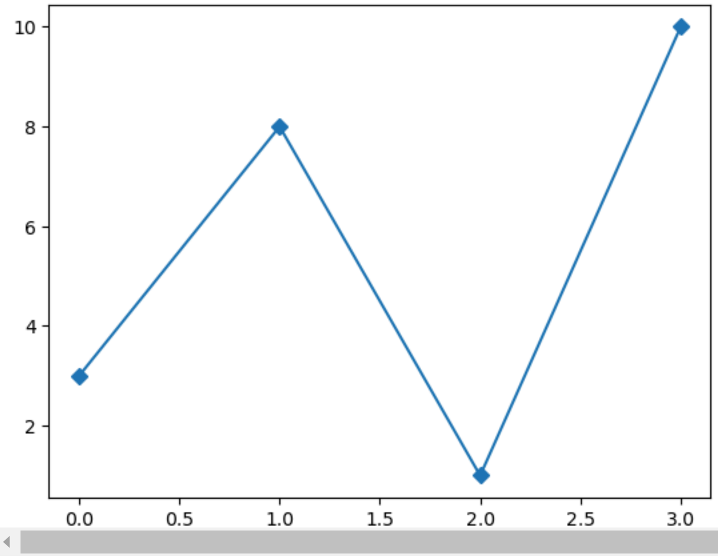
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'P')
plt.show()
```



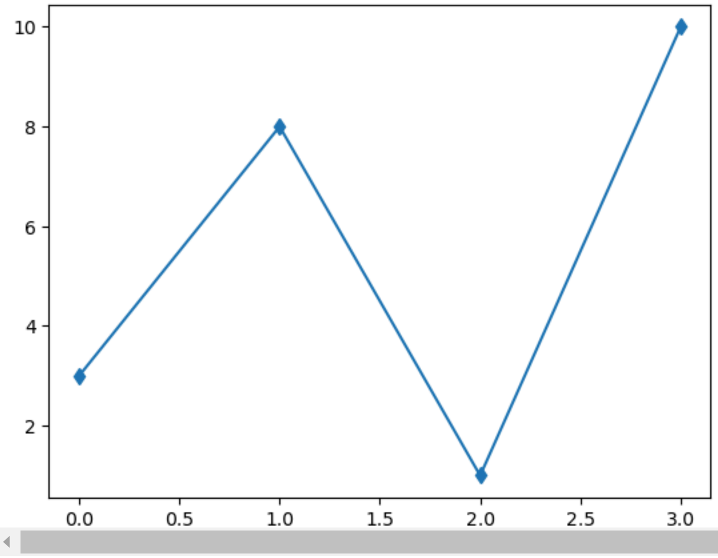
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 's')
plt.show()
```



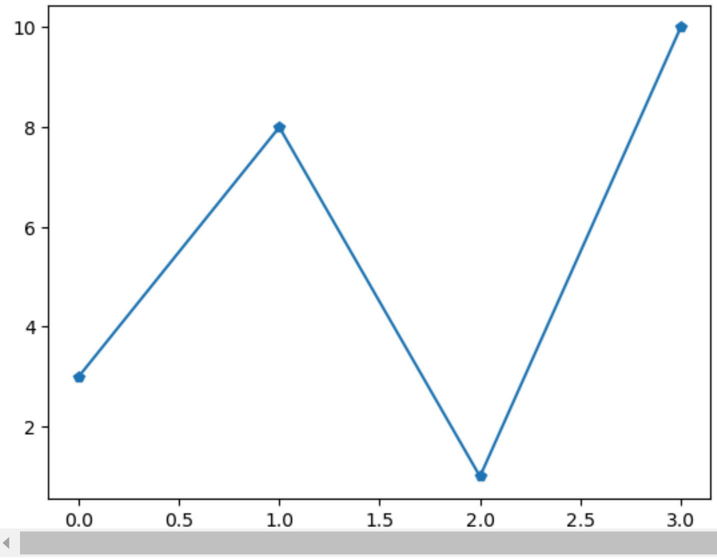
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'D')
plt.show()
```



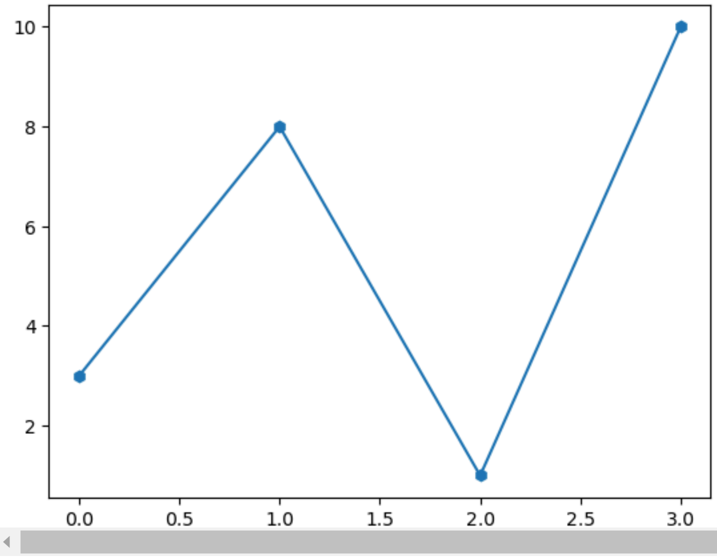
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'd')
plt.show()
```



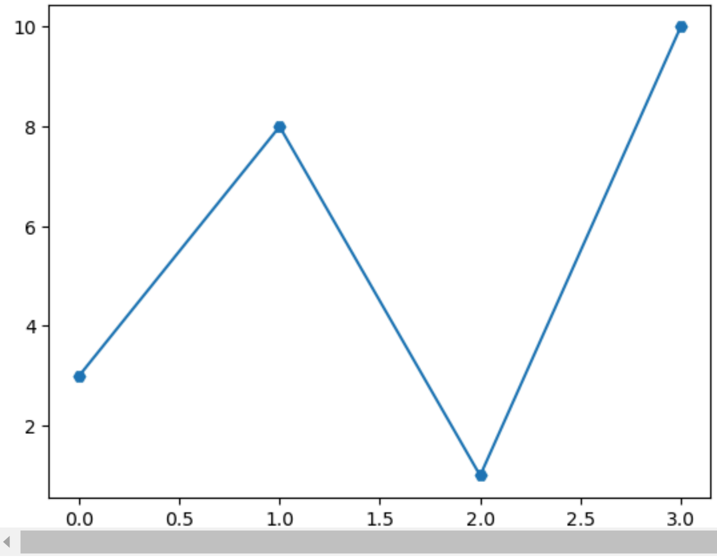
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'p')
plt.show()
```



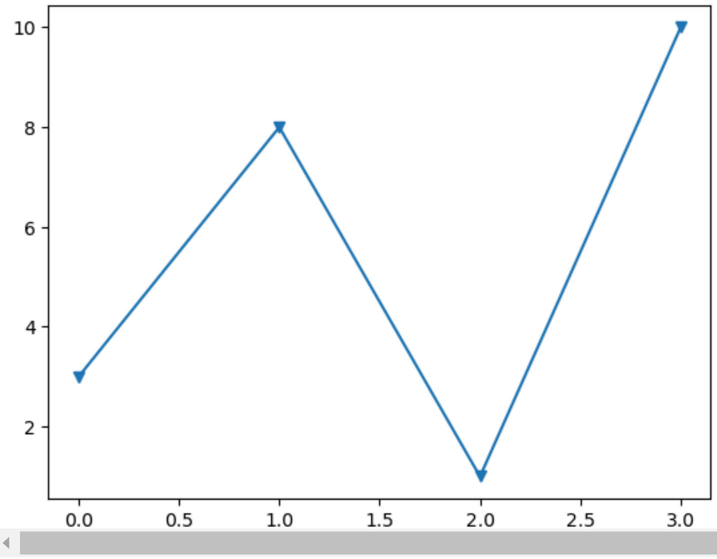
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'h')
plt.show()
```



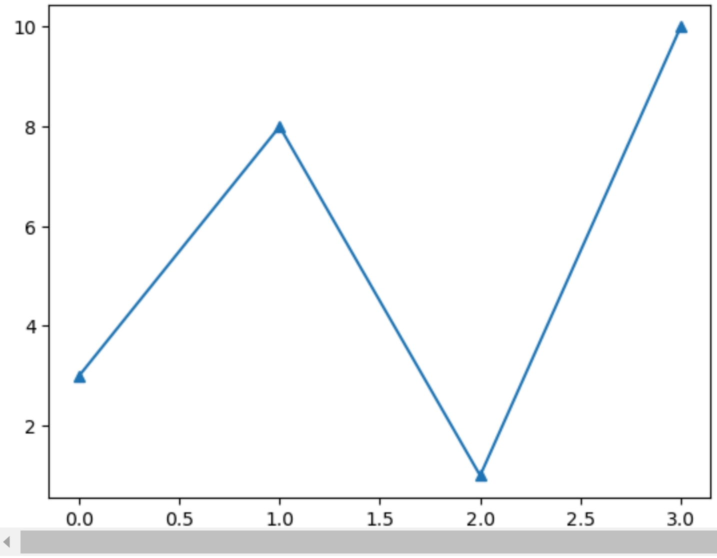
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'H')
plt.show()
```



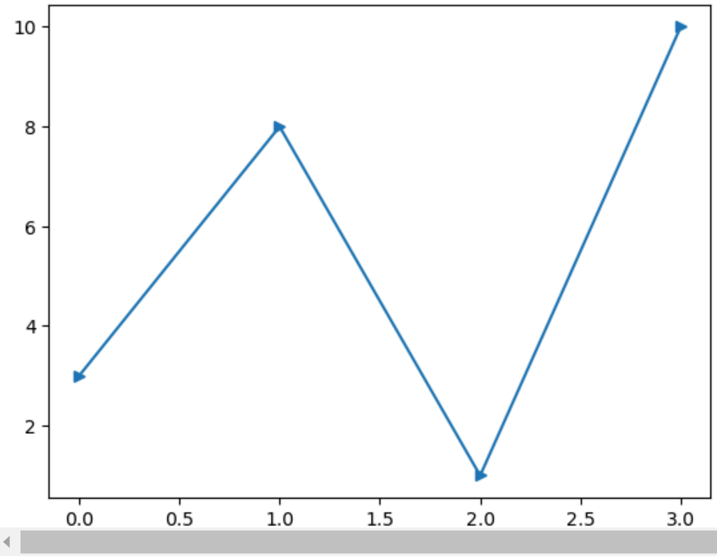
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'v')
plt.show()
```



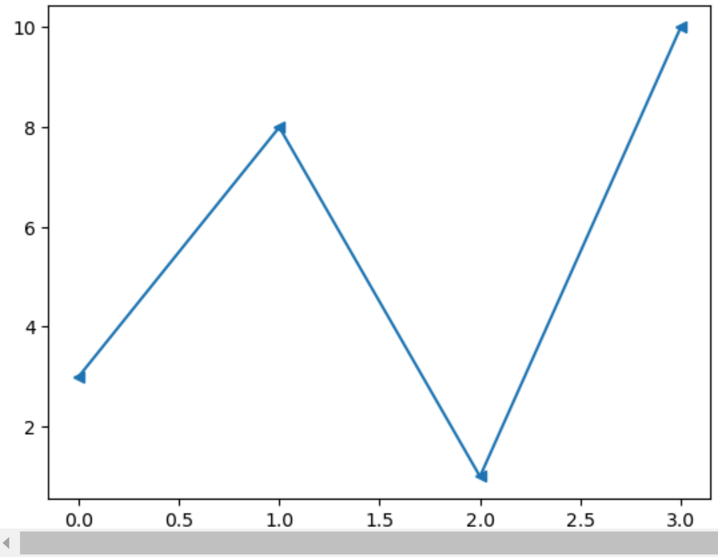
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '^')
plt.show()
```



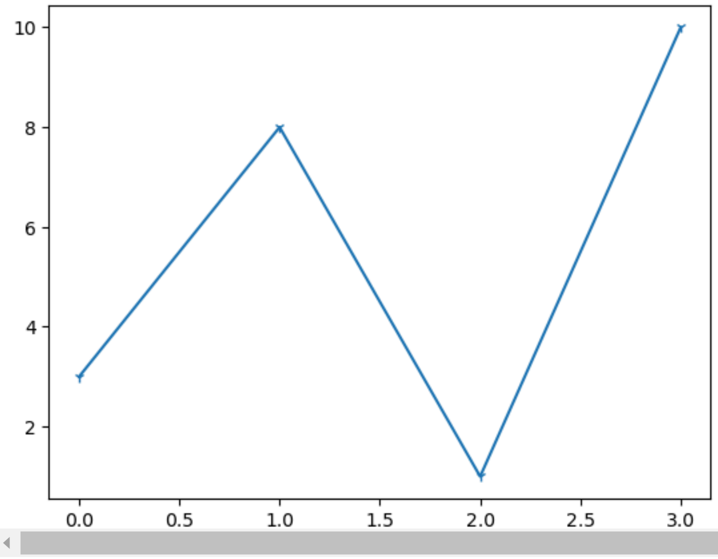
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '>')
plt.show()
```



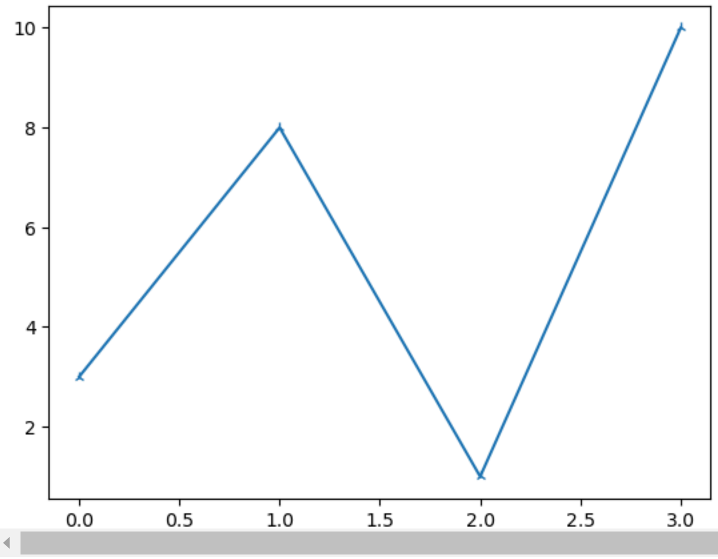
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '<')
plt.show()
```



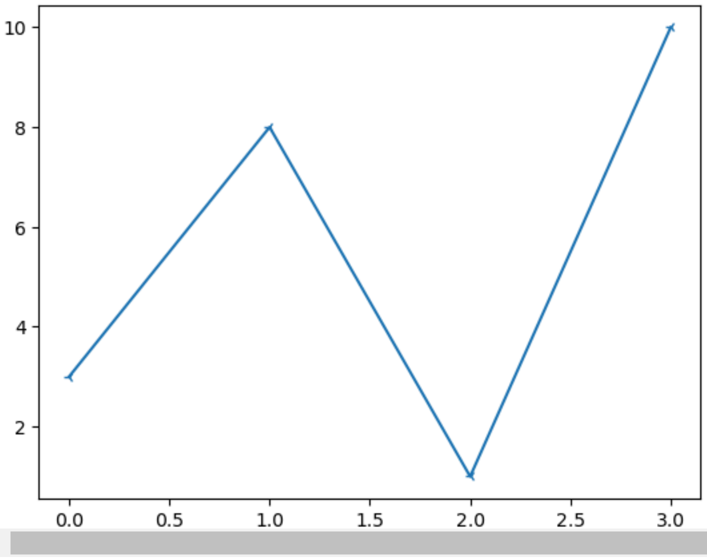
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '1')
plt.show()
```



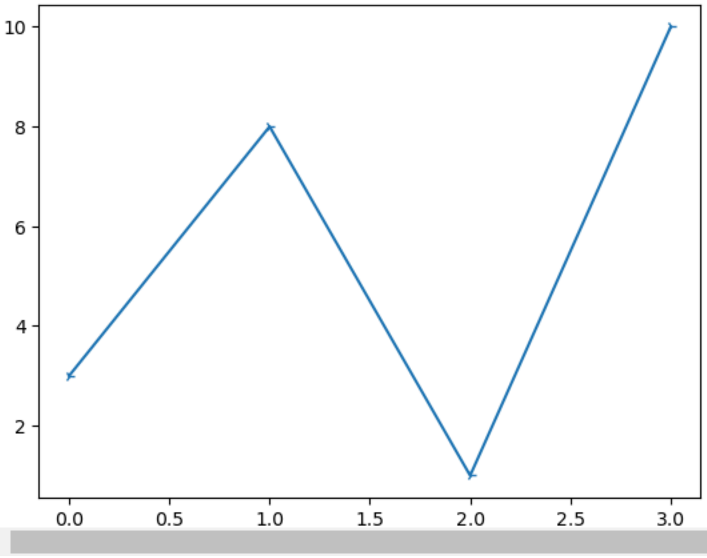
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '2')
plt.show()
```



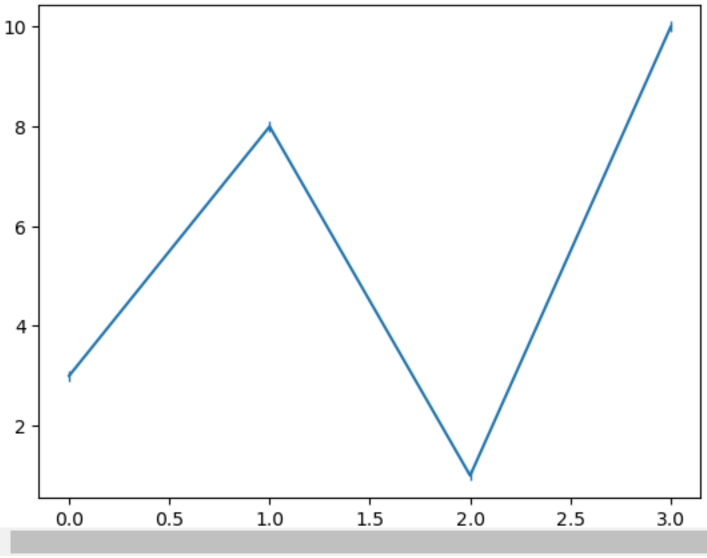
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '3')
plt.show()
```



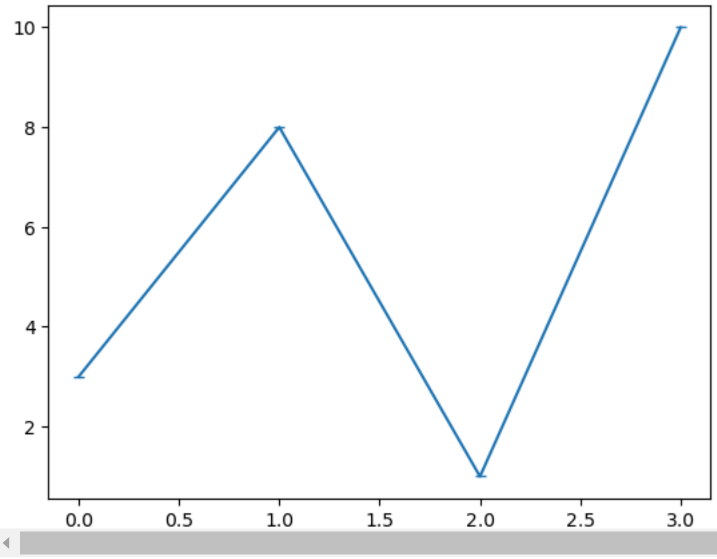
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '4')
plt.show()
```



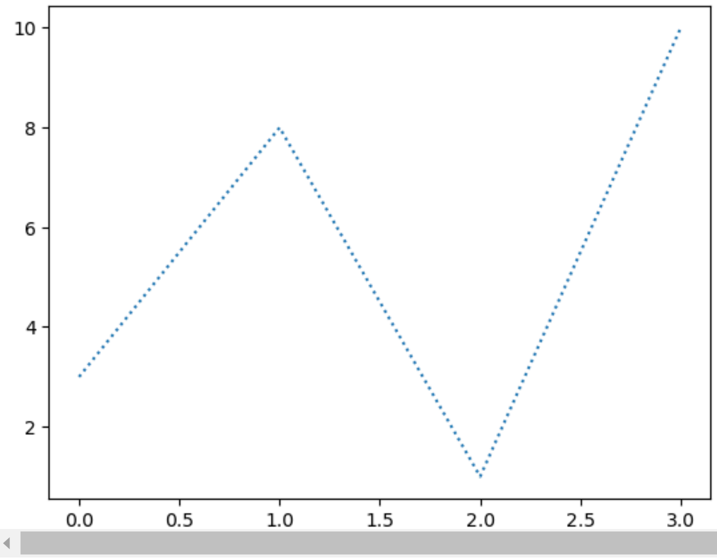
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '|')
plt.show()
```



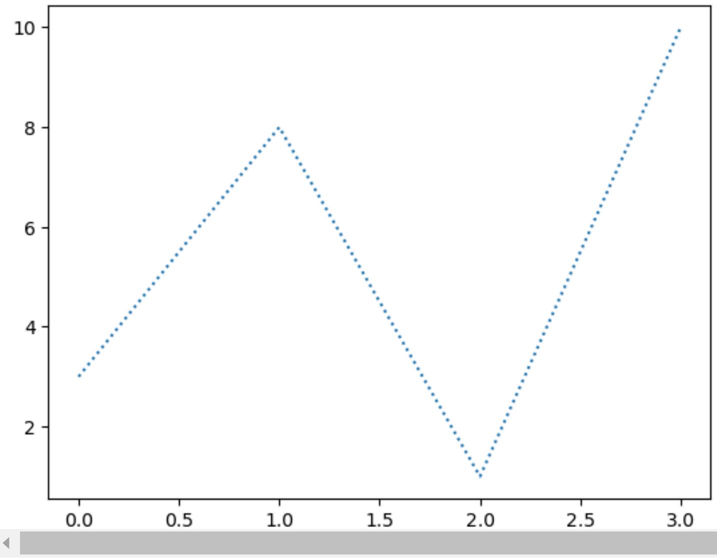
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '_')
plt.show()
```



```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

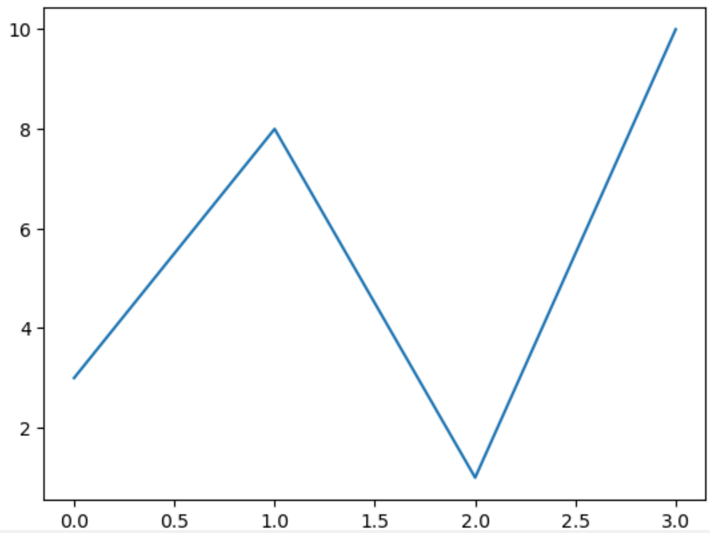


```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, ls = ':')
plt.show()
```

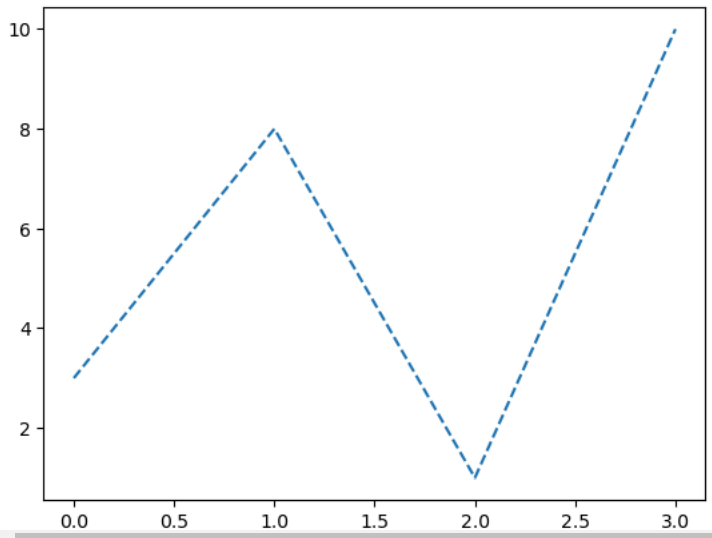


```
ypoints = np.array([3, 8, 1, 10])
```

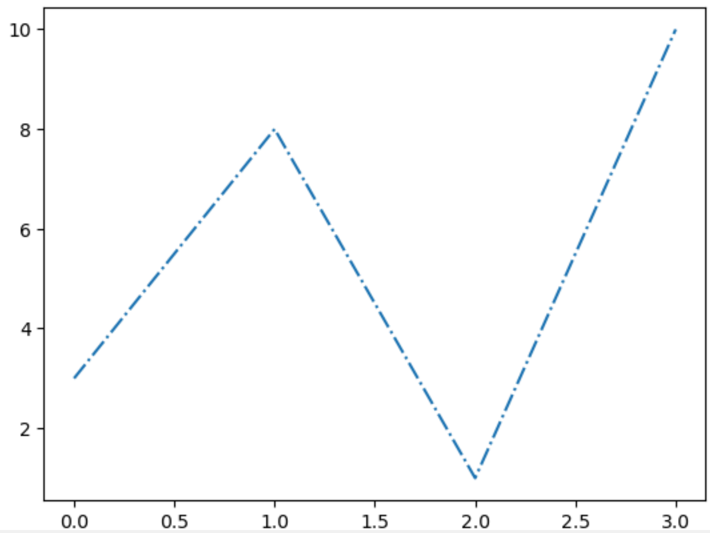
```
plt.plot(ypoints, linestyle = 'solid')
plt.show()
```



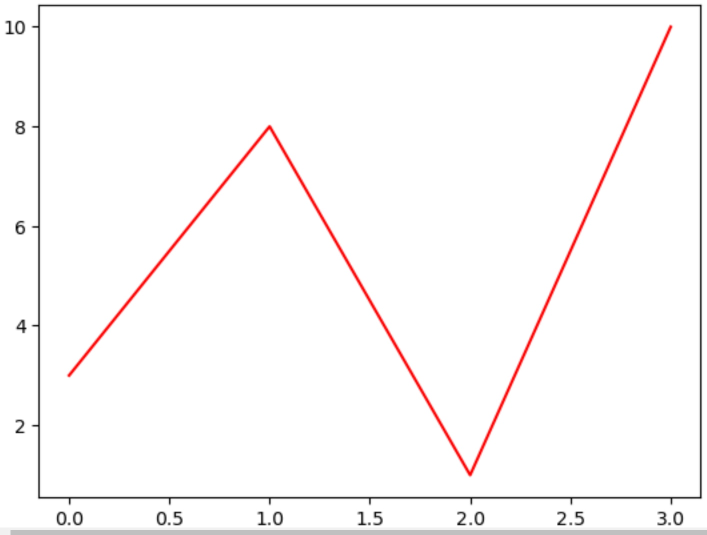
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dashed')
plt.show()
```



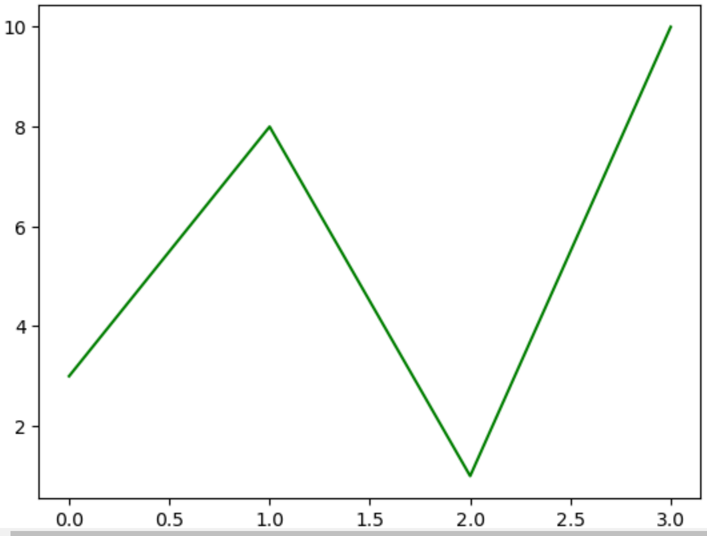
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, ls = '-.')
plt.show()
```



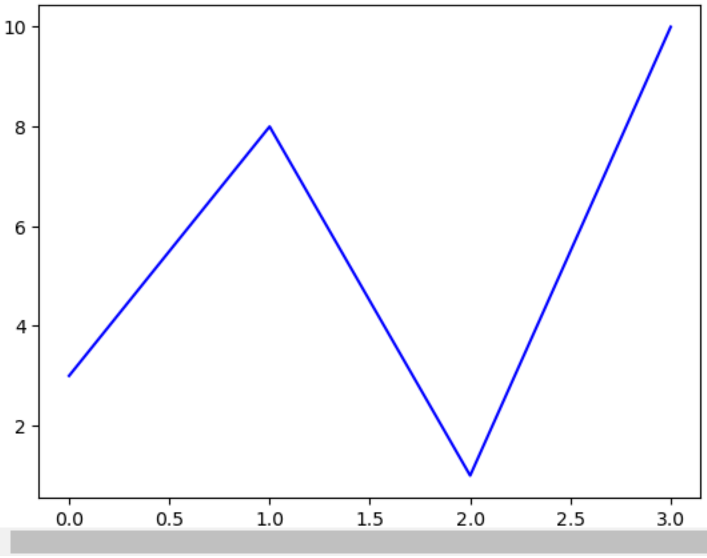
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'r')
plt.show()
```



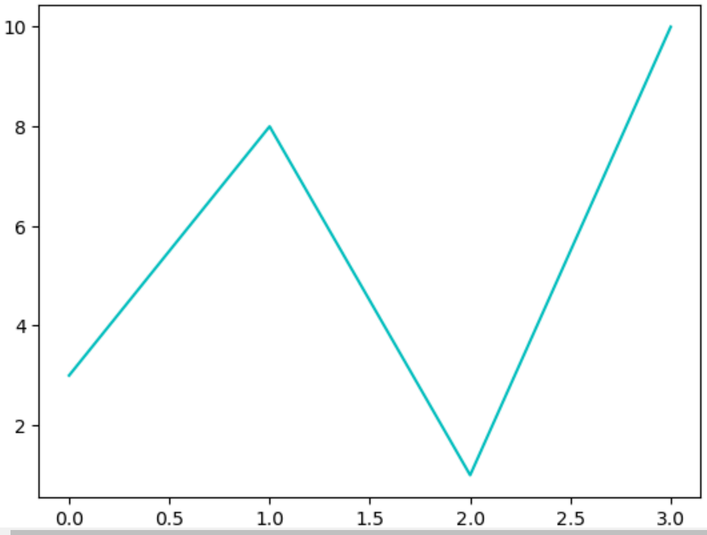
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'g')
plt.show()
```



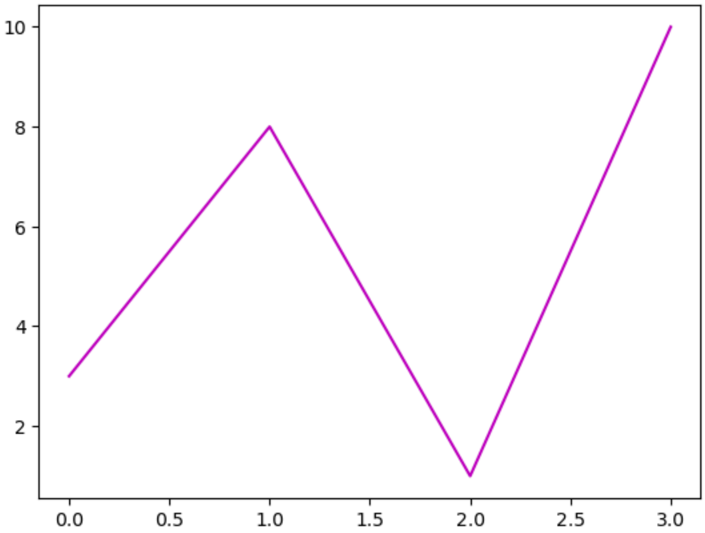
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'b')
plt.show()
```



```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'c')
plt.show()
```

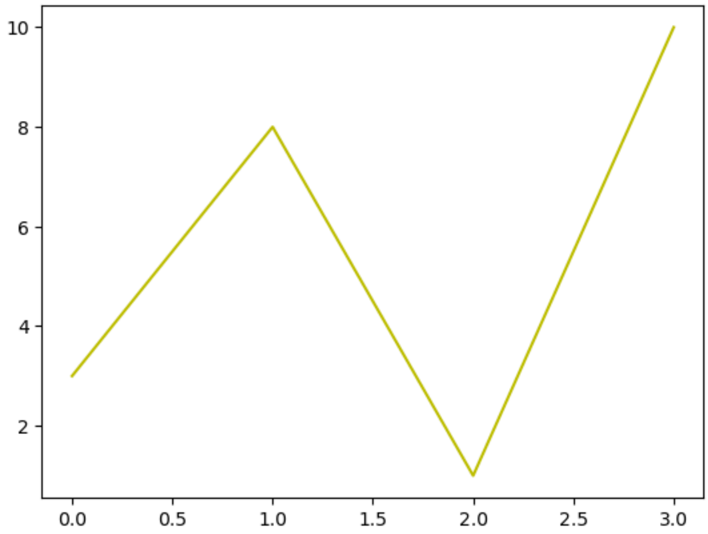


```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'm')
plt.show()
```

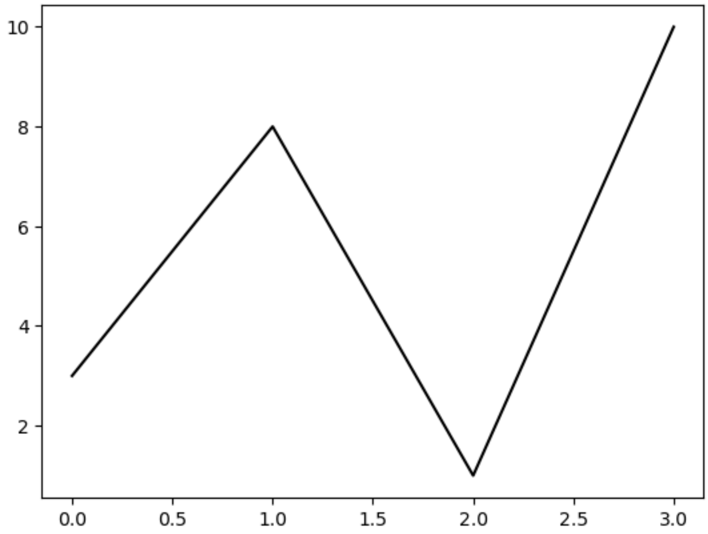


```
ypoints = np.array([3, 8, 1, 10])
```

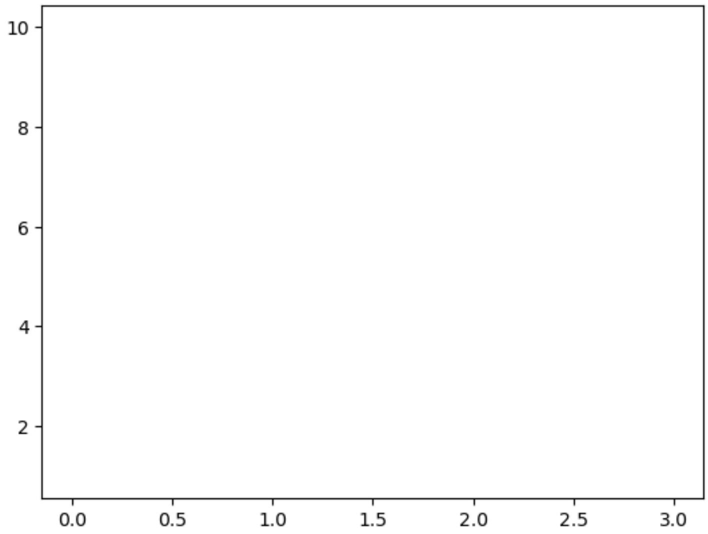
```
plt.plot(ypoints, color = 'y')
plt.show()
```



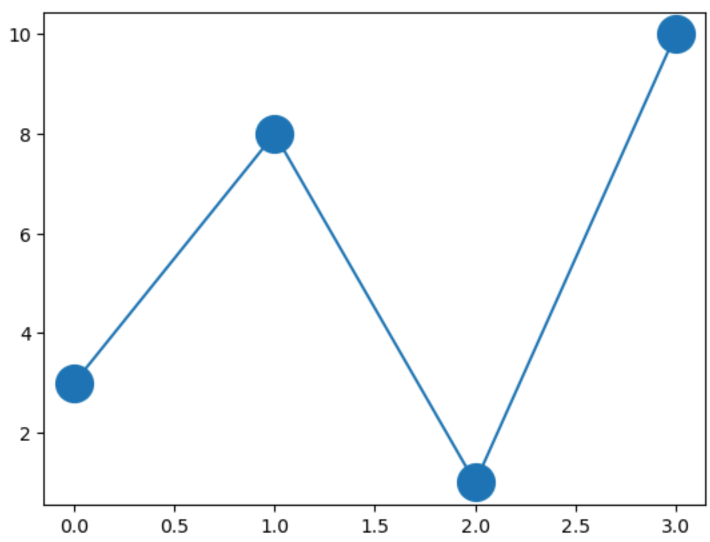
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'k')
plt.show()
```



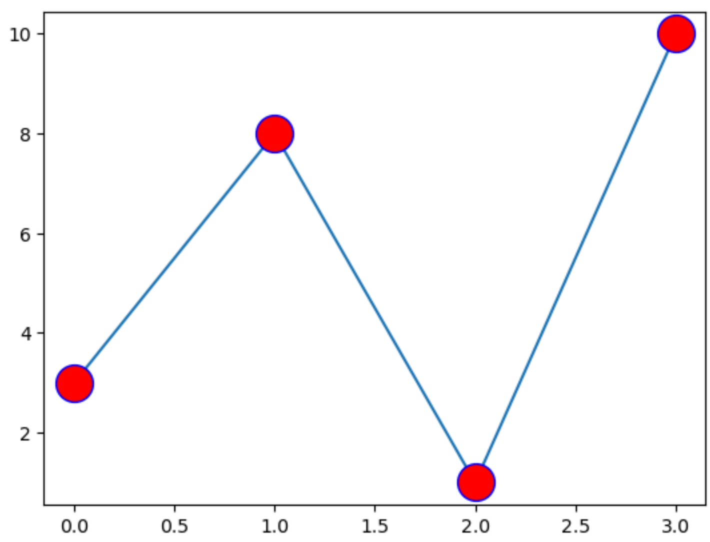
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'w')
plt.show()
```



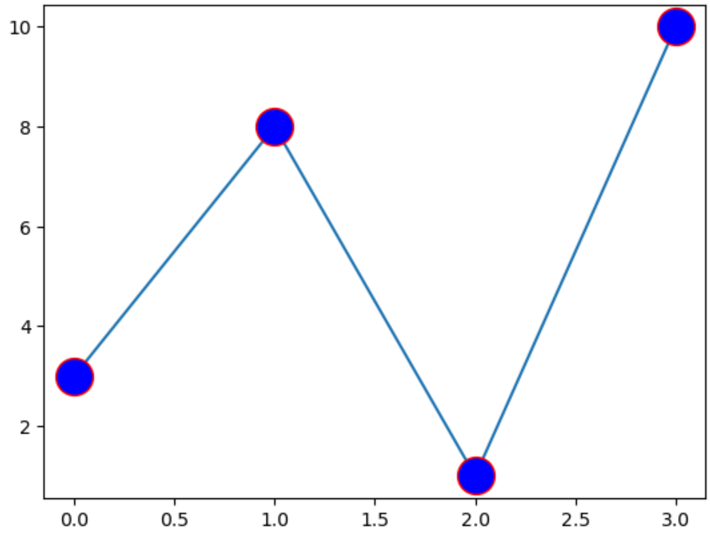
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20)
plt.show()
```



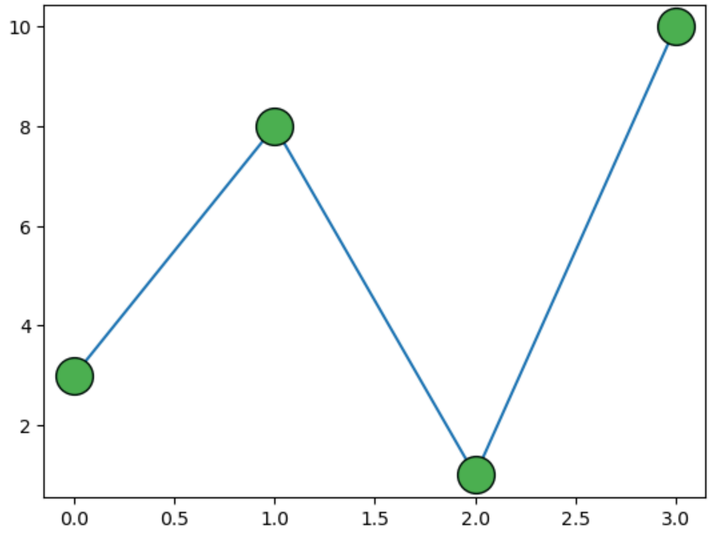
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec='blue', mfc = 'red')
plt.show()
```



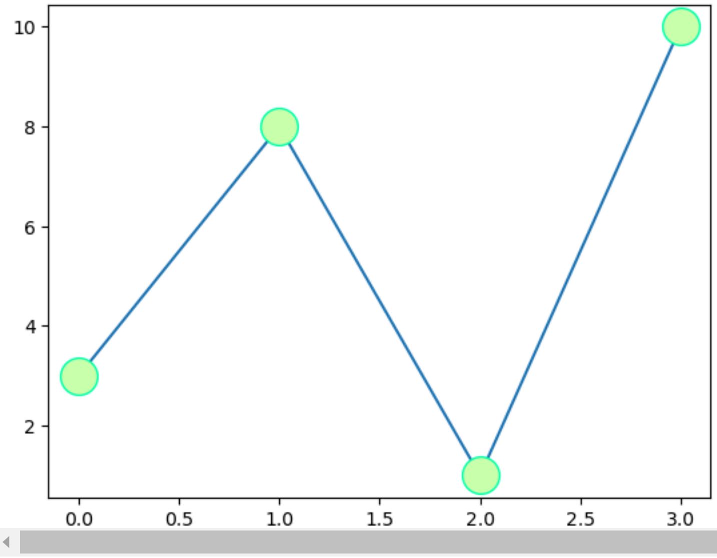
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec='red', mfc = 'blue')
plt.show()
```



```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = '#000F0A', mfc = '#4CAF50')
plt.show()
```

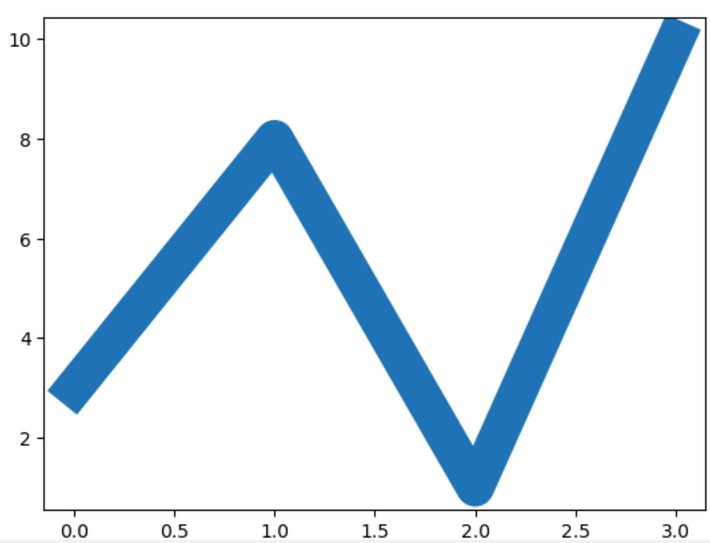


```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = '#0FA', mfc = '#CFA')
plt.show()
```

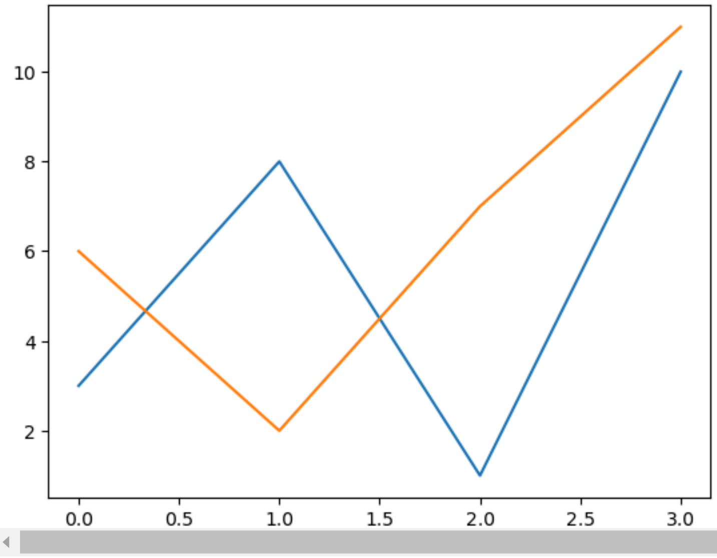


```
ypoints = np.array([3, 8, 1, 10])
```

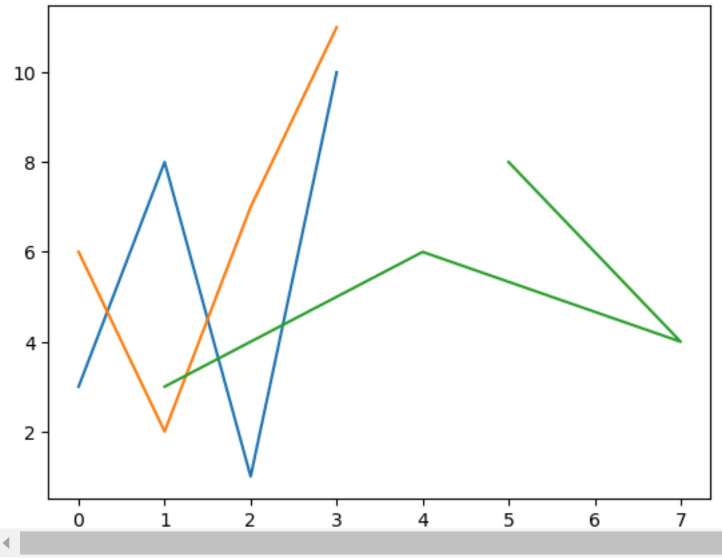
```
plt.plot(ypoints, linewidth = '20.5')
plt.show()
```



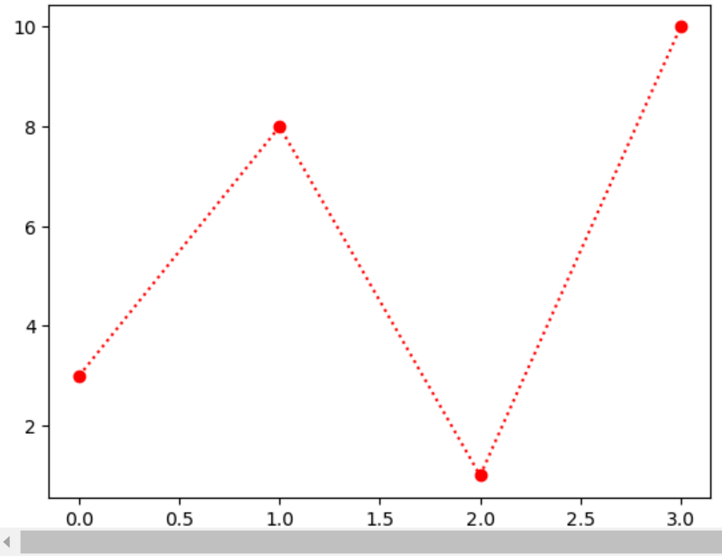
```
y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])

plt.plot(y1)
plt.plot(y2)

plt.show()
```



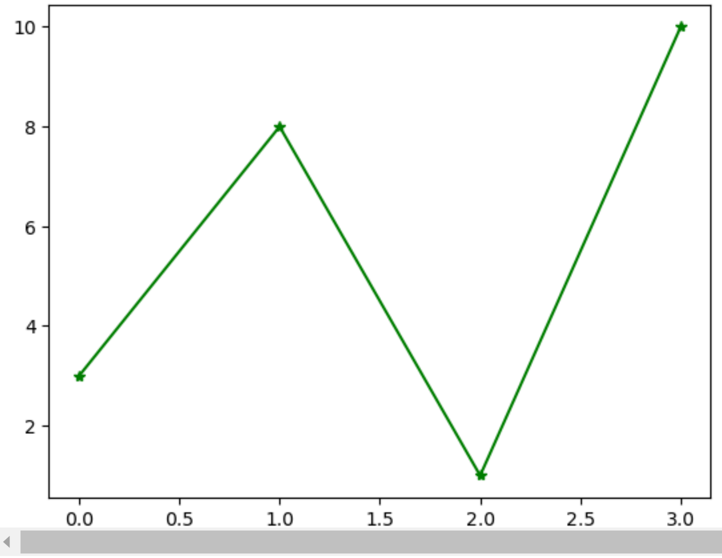
```
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])

x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

x3 = np.array([1, 4, 7, 5])
y3 = np.array([3, 6, 4, 8])

plt.plot(x1, y1,  x2, y2,  x3, y3)
plt.show()
```



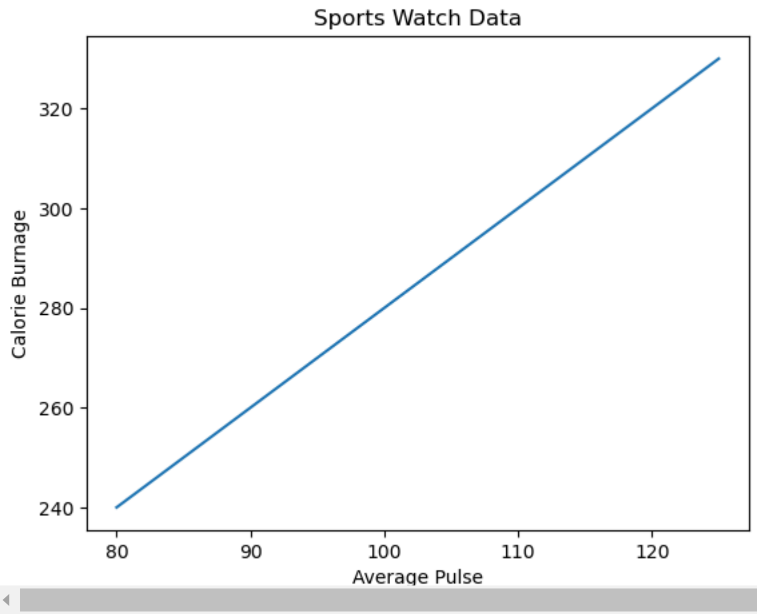
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, 'o:r')
plt.show()
```



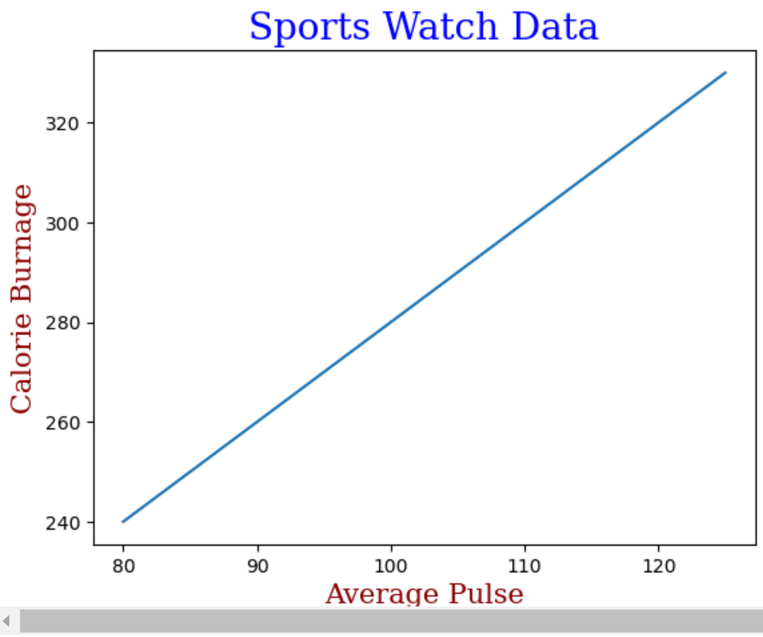
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, '*-g')
plt.show()
```



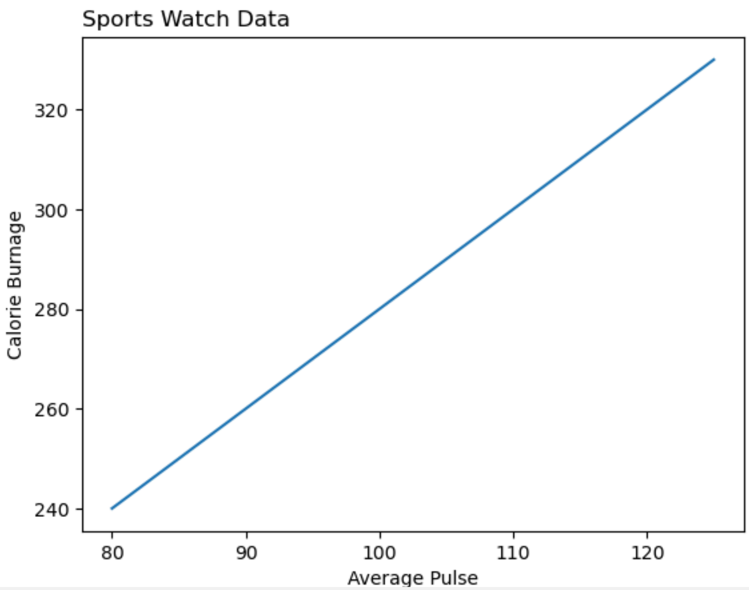
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```



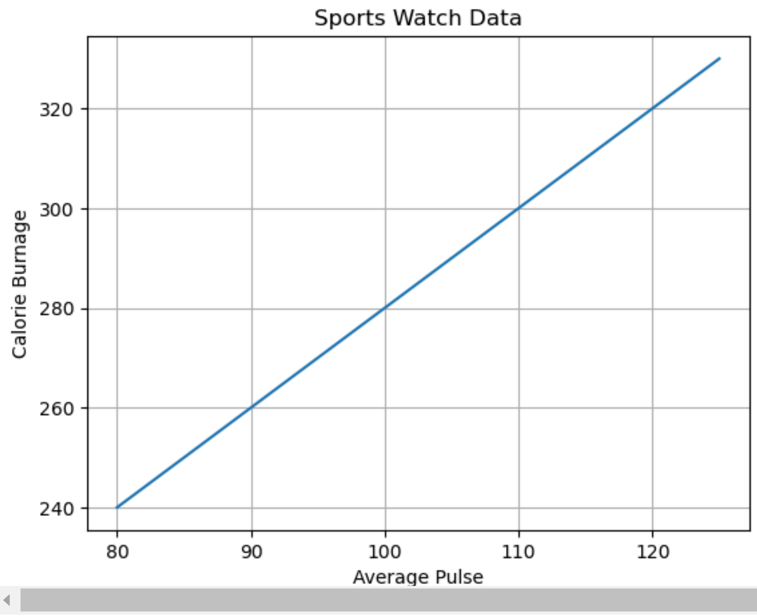
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)

plt.plot(x, y)
plt.show()
```



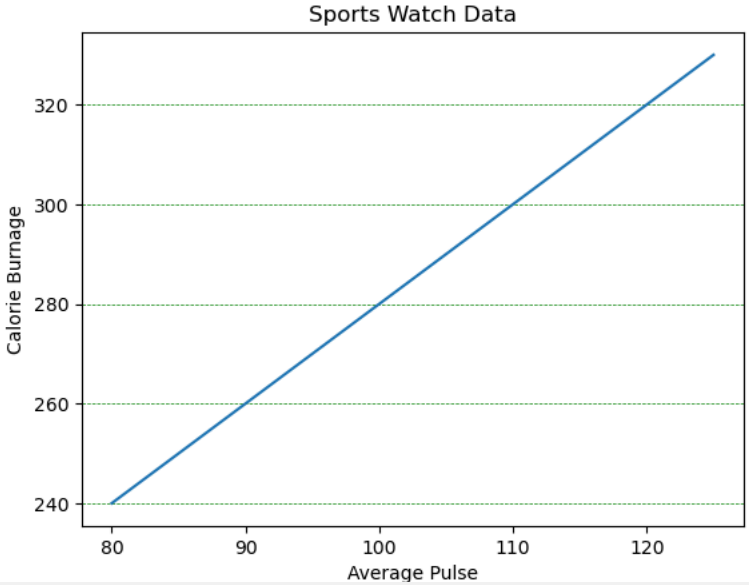
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.title("Sports Watch Data", loc = 'left')
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)
plt.show()
```



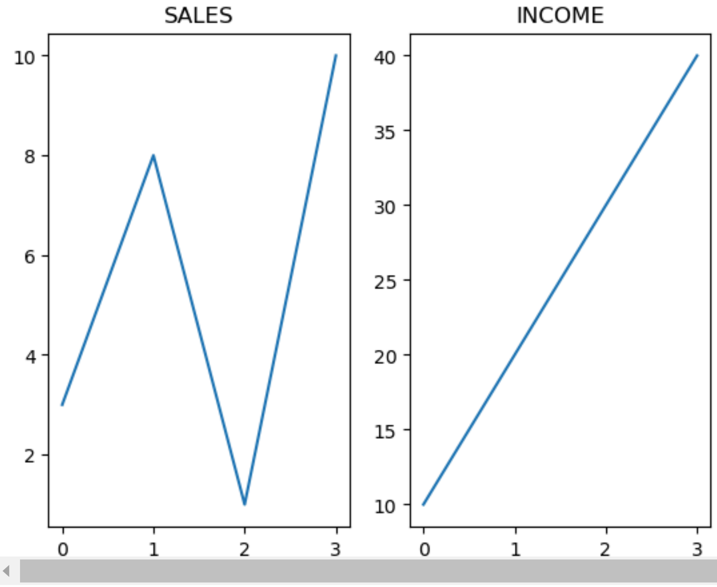
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid()

plt.show()
```



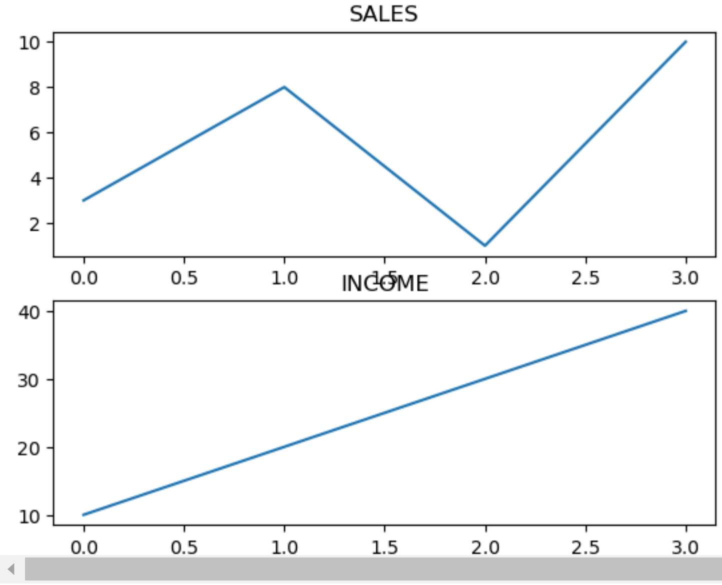
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid(axis = 'y', color = 'green', linestyle = '--', linewidth = 0.5)

plt.show()
```



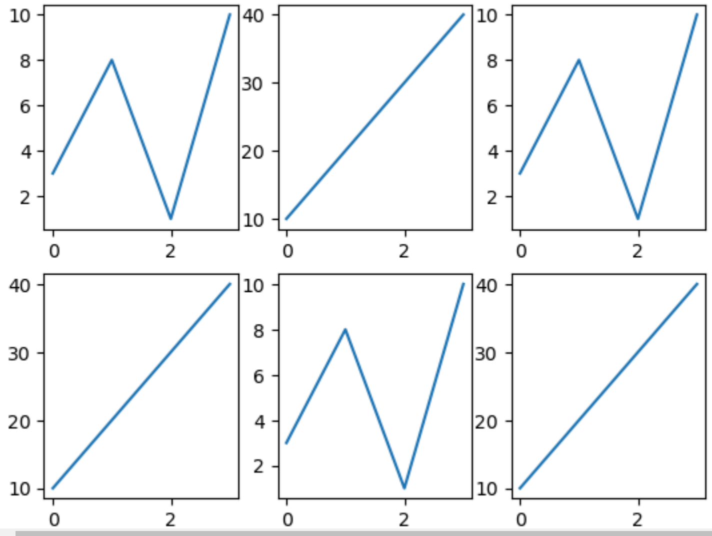
```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("SALES")

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")

plt.show()
```



```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 1, 1)
plt.plot(x,y)
plt.title("SALES")

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 1, 2)
plt.plot(x,y)
plt.title("INCOME")

plt.show()
```

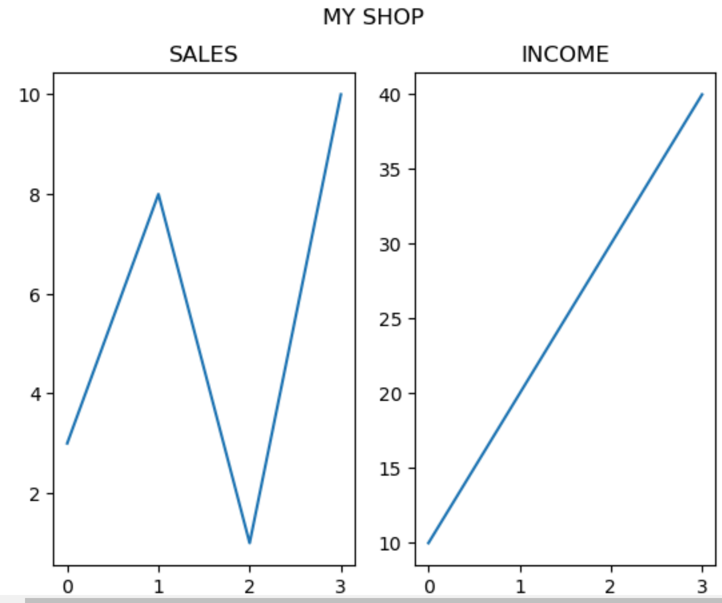


```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 1)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 2)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 3)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 4)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 5)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 6)
plt.plot(x,y)

plt.show()
```

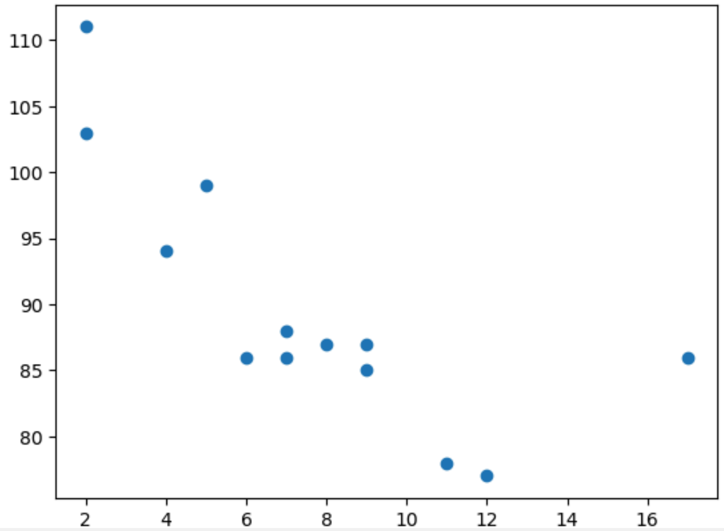


```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
```

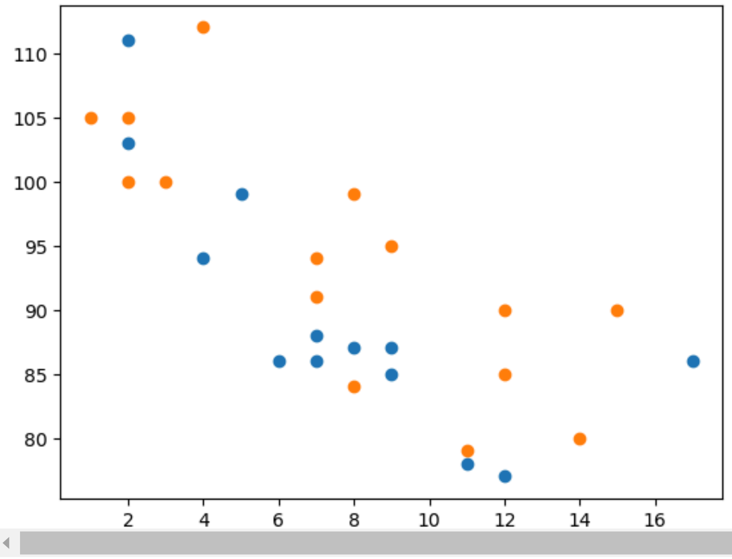
```
plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("SALES")

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")

plt.suptitle("MY SHOP")
plt.show()
```



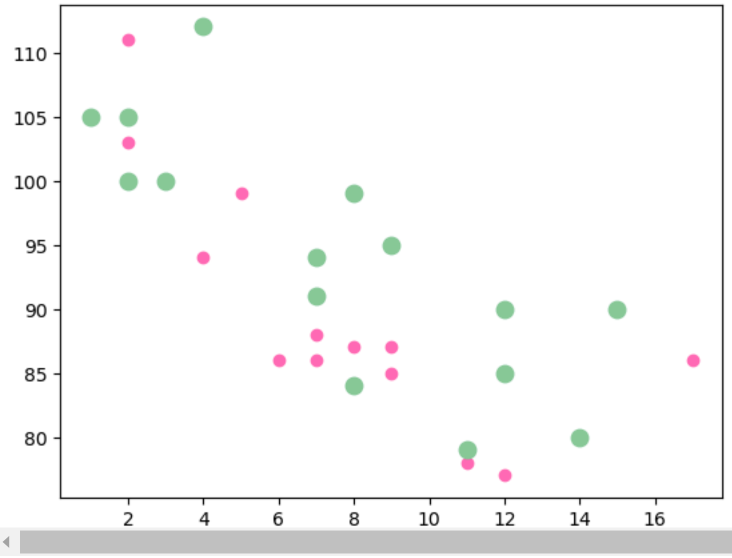
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```



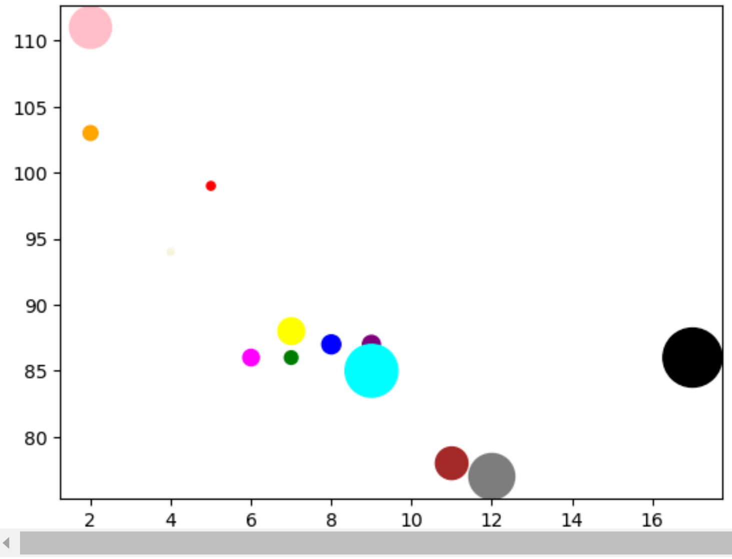
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y)

plt.show()
```



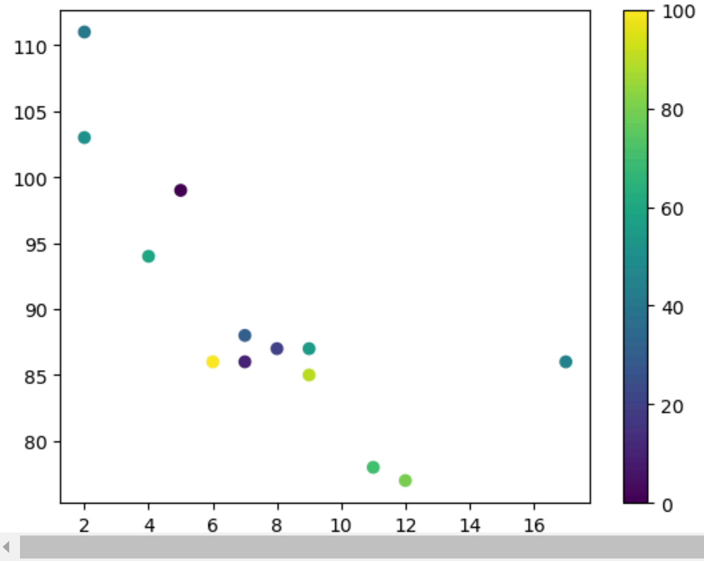
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y, color = 'hotpink')

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y, color = '#88c999' , s=80)

plt.show()
```



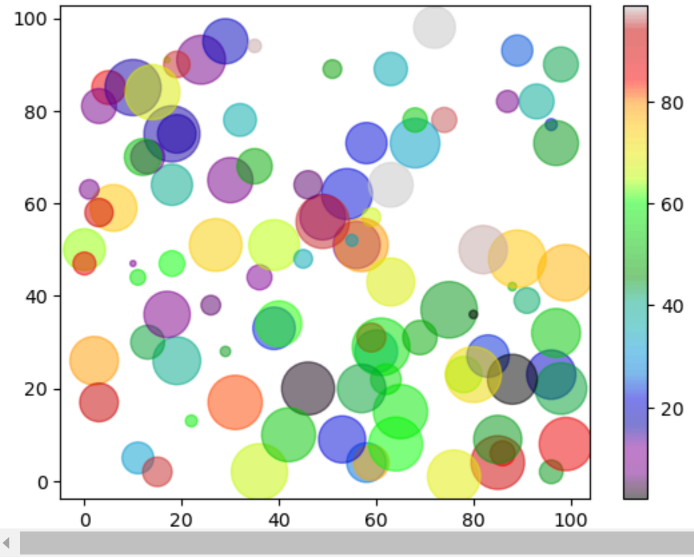
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

colors = np.array(["red","green","blue","yellow","pink","black","orange","purple","beige"
 ,"brown","gray","cyan","magenta"])

sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, c=colors, s=sizes)

plt.show()
```



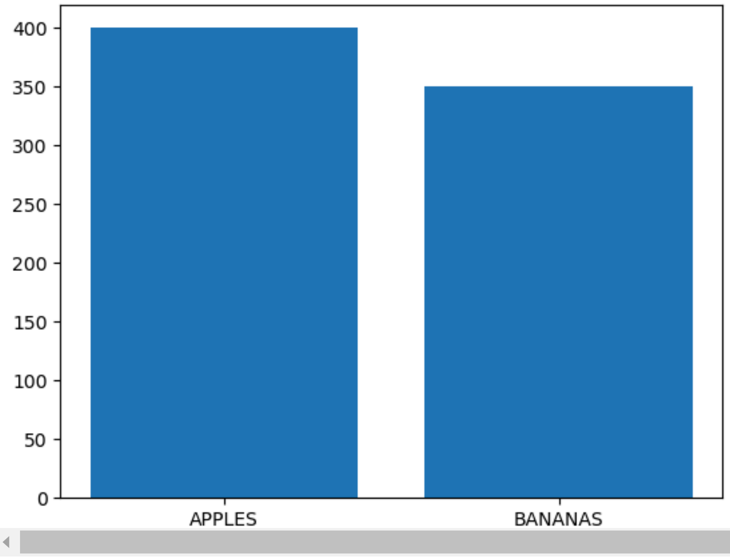
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='viridis')
plt.colorbar()
plt.show()
```



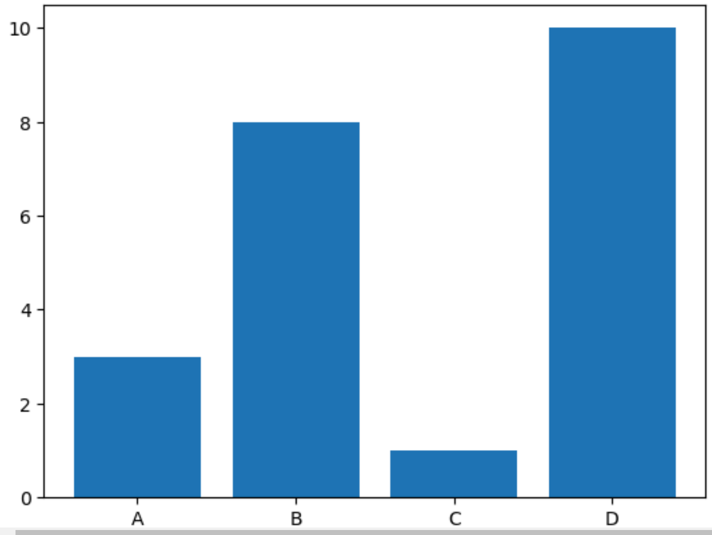
```
x = np.random.randint(100, size=(100))
y = np.random.randint(100, size=(100))

colors = np.random.randint(100, size=(100))
sizes = 10 * np.random.randint(100, size=(100))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')

plt.colorbar()

plt.show()
```



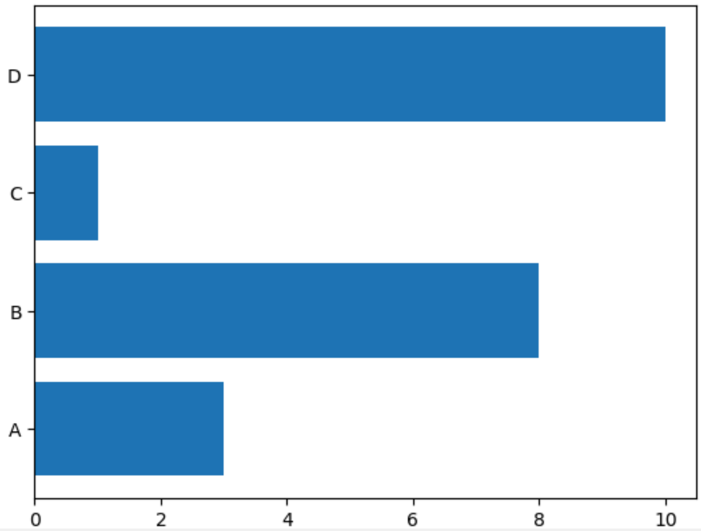
```
x = ["APPLES", "BANANAS"]
y = [400, 350]

plt.bar(x, y)
plt.show()
```



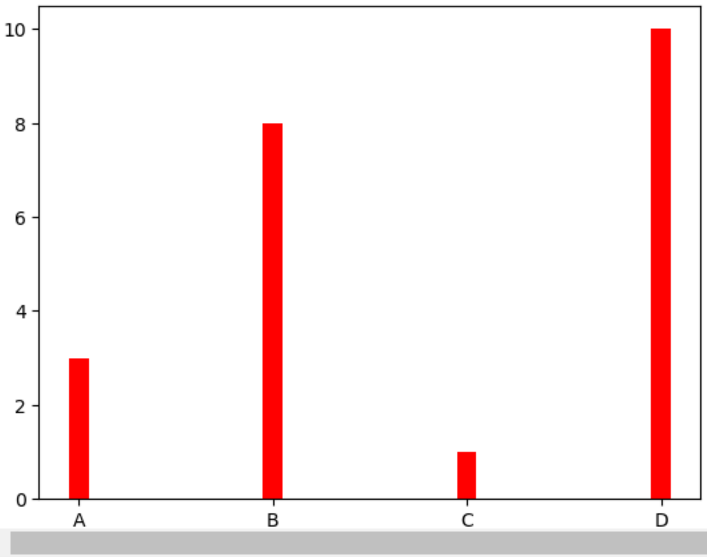
```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```



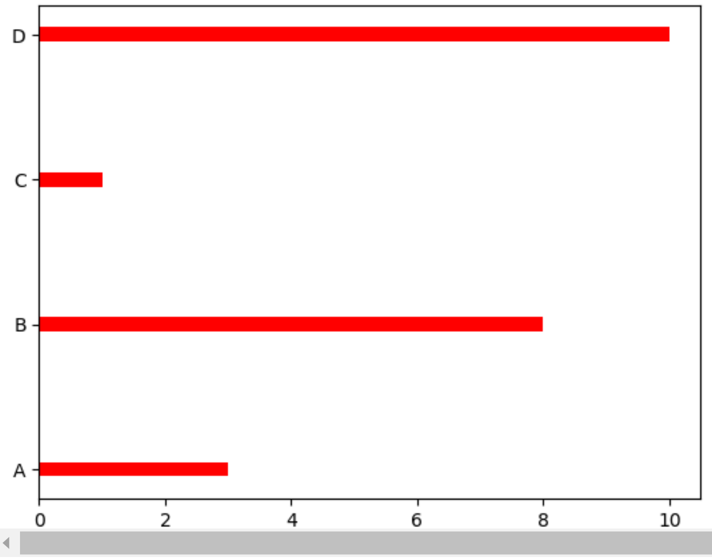
```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x, y)
plt.show()
```



```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, color='red', width = 0.1)
plt.show()
```



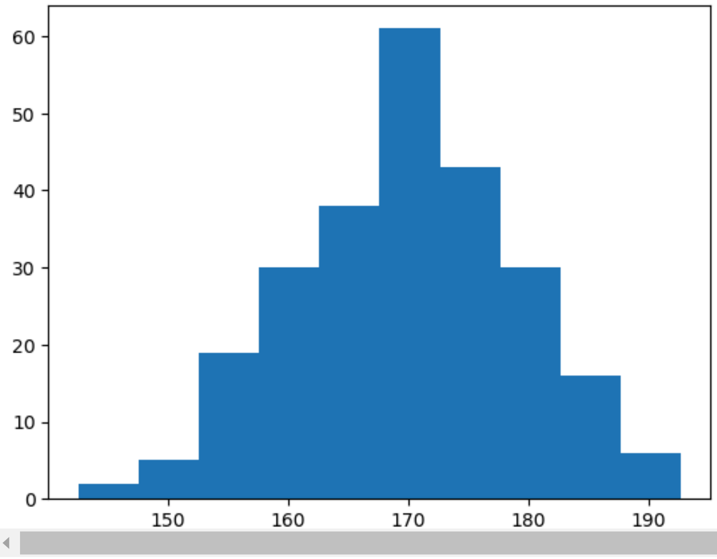
```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x, y, color='red', height = 0.1)
plt.show()
```



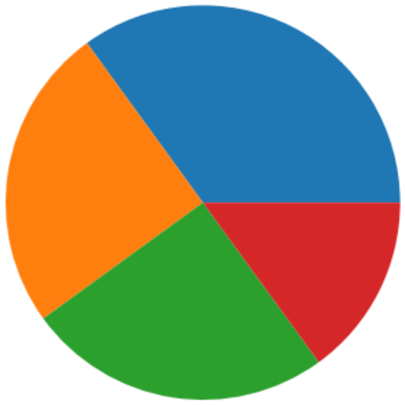
```
lengths = np.random.normal(170, 10, 250)
print(lengths)
```

```
[178.14039447 174.82552376 180.71632959 159.47972867 184.25200067
 173.63041021 173.57934696 178.89154075 166.44235856 177.0235705
 165.66903821 174.22254324 173.02488614 168.09411297 164.18638806
 164.07011038 162.08031186 170.94043784 173.08755602 176.65790949
 174.27565992 193.25161424 163.05277315 167.85377604 169.26019691
 171.48408007 158.45028596 154.72714166 162.40570043 175.0321693
 173.66641043 163.65211868 168.7453893  167.20882339 150.37151781
 164.74709516 191.3920075  157.19954625 168.15991136 175.4938279
 163.3345563  152.74778872 177.94860738 168.41551219 182.02032836
 163.27199057 171.38901594 160.09101063 180.22833372 181.94042341
 180.10808796 167.88491851 151.99791213 168.06128397 173.2137529
 165.68263538 164.82671044 181.48178662 187.63341356 164.68189279
 168.94712029 178.31213566 179.81383855 156.16378123 166.7144014
 171.33959825 171.13043647 159.86495224 155.87768935 168.33437177
 173.37431472 160.45780196 165.98465633 168.38596928 171.18065281
 160.8441479  163.72898309 166.83869996 173.36566995 164.57828861
 146.57164577 164.22822344 191.41370294 177.01431305 165.60030575
 174.72426826 158.22907121 171.70114206 180.77867812 151.06293074
 169.9206339  169.81847313 190.88824917 163.07928025 158.14102272
 160.27096171 169.13435623 180.84644545 180.60914882 169.6347308
 171.65554806 166.05179576 173.36375166 174.46612022 159.3161497
 154.50730765 171.542314   189.13426141 169.43349092 160.12784691
 178.86850895 172.64571708 163.59453086 162.92286773 168.53997429
 171.83393208 161.2592814  165.21037795 166.43683224 173.00922554
 169.06252667 169.47965461 168.57821156 172.80506928 157.28813589
 161.38937056 162.9679014  174.97612253 181.3996531  182.63681525
 178.29677774 166.39461978 184.28661576 174.61658411 157.93362121
 173.77924058 174.97884611 192.52405463 183.59733593 175.46756628
 175.45037588 185.47595173 172.41740636 178.02169384 172.23033108
 181.33764605 135.82510228 172.02955538 161.16688177 170.94378856
 196.29497981 172.48965855 157.06548158 168.85940166 169.62555378
 176.14143302 181.55879608 146.27944451 173.2582173  174.92491862
 169.56044931 158.82570181 174.7971281  158.64117358 162.38657959
 166.31457489 161.43052489 177.18401099 184.47533279 185.96896818
 175.45566486 171.00926956 172.36484517 177.84077538 178.76382417
 162.31412206 168.70906645 166.82208513 171.98358955 150.42928568
 172.18957345 170.60562974 177.76250479 167.83191613 165.73877519
 174.30495632 166.92488764 181.4809027  172.35552024 166.05987674
 175.44628992 164.39586126 164.37993247 168.19004987 188.80111847
 155.11239378 176.46229474 176.62856289 153.28992896 182.01199649
 162.06166382 161.72282844 159.69868959 167.26158096 157.93012333
 168.03621172 165.0380764  179.15805164 176.00547458 188.00965716
 171.06660908 162.72533077 139.89245935 171.32984537 167.63779995
 182.65814644 172.71889556 182.68430823 160.89309879 181.63700188
 180.72966834 182.95506754 187.74953773 175.89745865 176.98425015
 166.40481969 149.5189683  172.63568116 147.63456847 179.01292125
 172.14494801 181.42204388 181.31079723 183.70352816 180.16726355
 184.05062035 168.14964609 160.15375314 171.56129431 139.50375783
 164.11746183 206.76069181 184.51858482 190.20354786 165.23961198
 164.48201447 160.44945371 163.57733476 179.1640058  176.13100896]
```

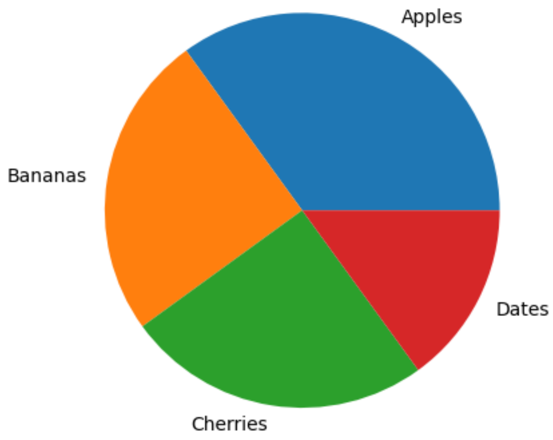
```
lengths = np.random.normal(170, 10, 250)

plt.hist(lengths)
plt.show()
```



```
y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

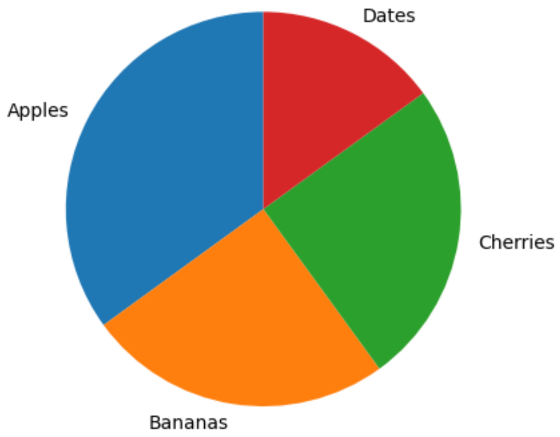


```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```

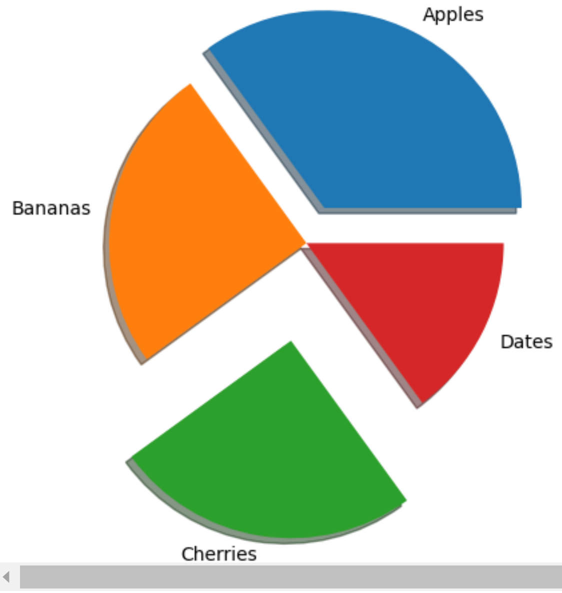


```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
```

```
plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```

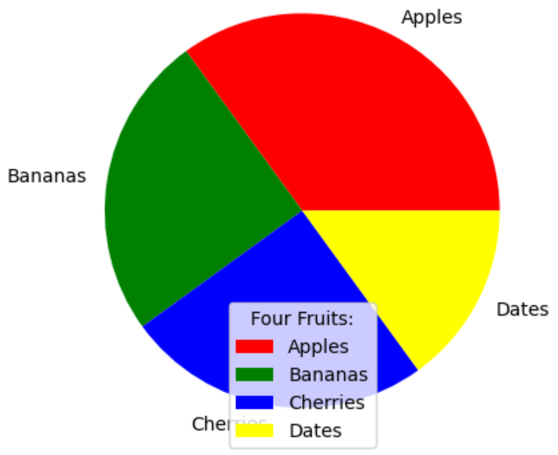


```
y=np.array([35,25,25,15])
mylabels = ["Apples","Bananas","Cherries","Dates"]
myexplode = [0.2,0,0.5,0]

plt.pie(y,labels = mylabels , explode = myexplode , shadow = True)
plt.show()
```



```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
mycolors = ['red', 'green', 'blue' , 'yellow']

plt.pie(y, colors = mycolors ,labels = mylabels, shadow=False)

plt.legend(title = "Four Fruits:")
```

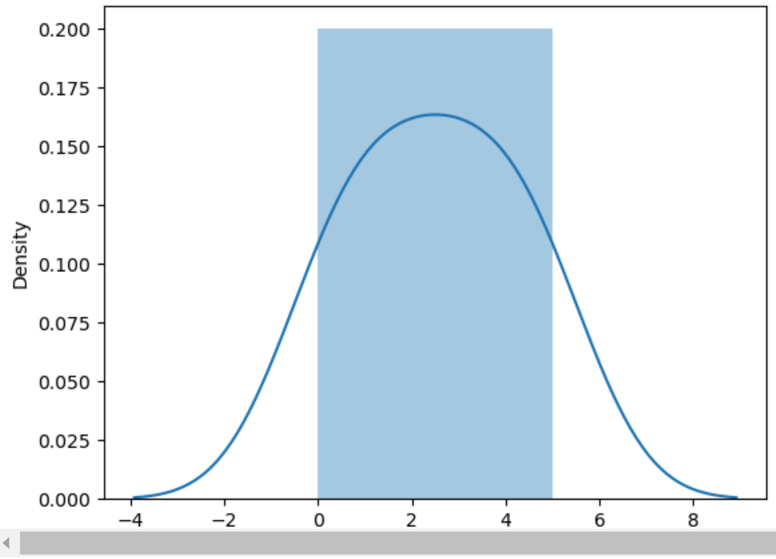
```
<matplotlib.legend.Legend at 0x18de2e60cb0>
```



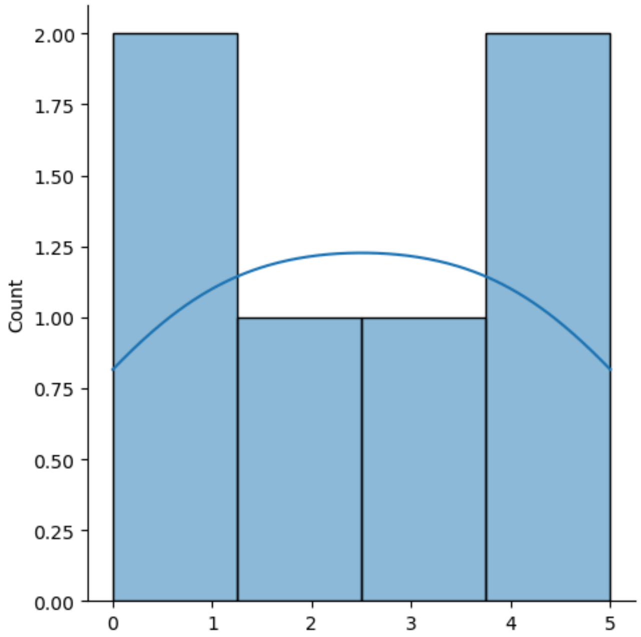
Seaborn

```
import seaborn as sns

sns.distplot([0, 1, 2, 3, 4, 5])

plt.show()
```

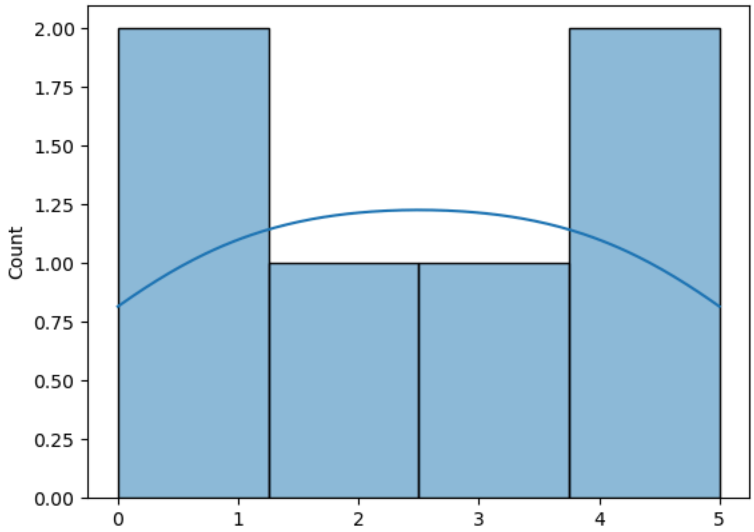
```
C:\Users\hpp\AppData\Local\Temp\ipykernel_2440\1132304403.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot([0, 1, 2, 3, 4, 5])
```



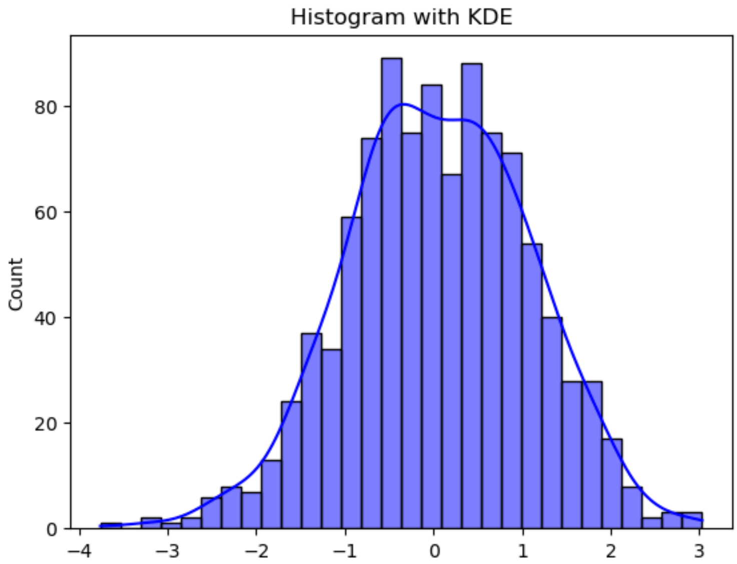
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.displot([0, 1, 2, 3, 4, 5],kde=True)

plt.show()
```



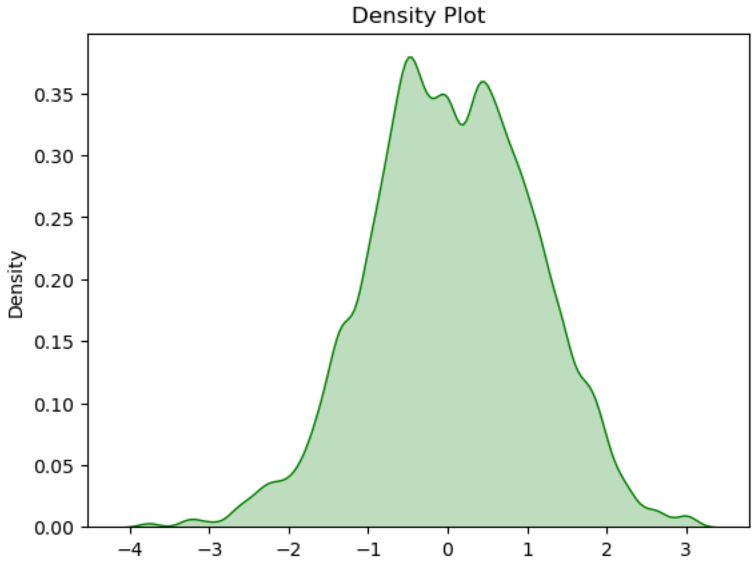
```
sns.histplot([0, 1, 2, 3, 4, 5] ,kde=True )

plt.show()
```



```
data = np.random.randn(1000)

sns.histplot(data, bins=30, kde=True, color="blue")

plt.title("Histogram with KDE")
plt.show()
```

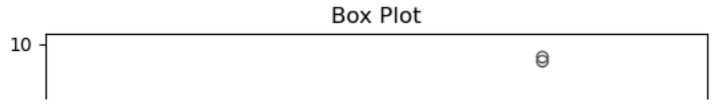


```
sns.kdeplot(data, fill=True, color="green", bw_adjust=0.5)
plt.title("Density Plot")
plt.show()
```



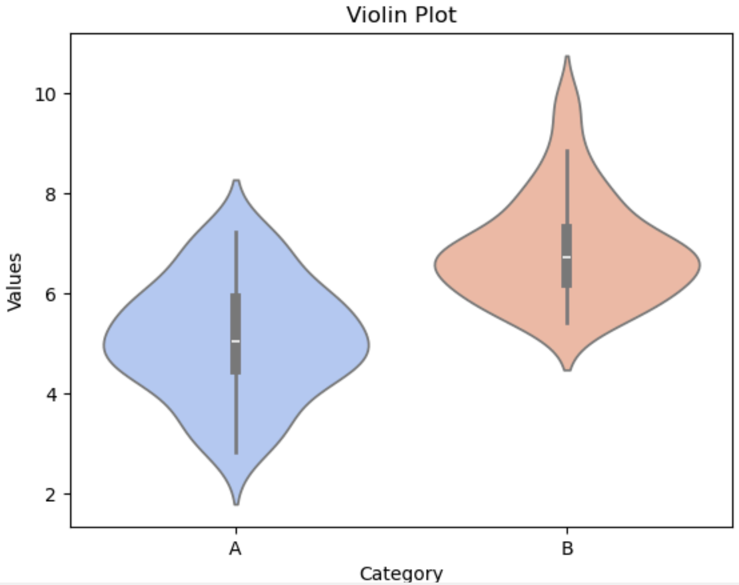
```
import pandas as pd

df = pd.DataFrame({"Category": ["A"] * 50 + ["B"] * 50, "Values": np.concatenate([np.random.randn(50) + 5, np.random.randn(50) + 7])})

sns.boxplot(x="Category", y="Values", data=df, palette="Set2")

plt.title("Box Plot")
plt.show()
```

```
C:\Users\hpp\AppData\Local\Temp\ipykernel_2440\2754221814.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.boxplot(x="Category", y="Values", data=df, palette="Set2")
```

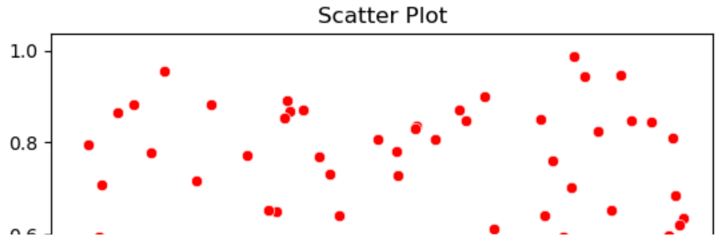


```
sns.violinplot(x="Category", y="Values", data=df, palette="coolwarm")
plt.title("Violin Plot")
plt.show()
```

```
C:\Users\hpp\AppData\Local\Temp\ipykernel_2440\3154011866.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.violinplot(x="Category", y="Values", data=df, palette="coolwarm")
```



```
df = pd.DataFrame({"X": np.random.rand(100), "Y": np.random.rand(100)})

sns.scatterplot(x="X", y="Y", data=df, color="red")

plt.title("Scatter Plot")
plt.show()
```


Scatter Plot