Matplotlib

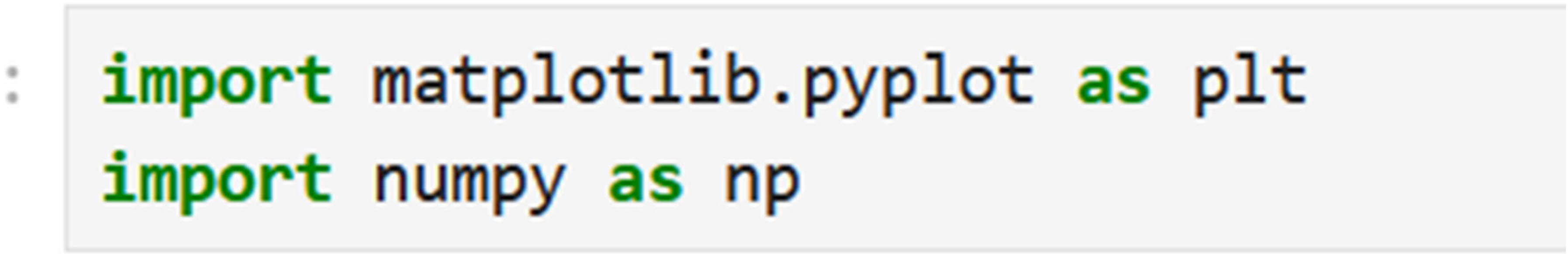
Matplotlib

Matplotlib is a low level graph plotting library in python

- provides various visualization utility (generate plots, chart and graphs)
- commonly used for exploration of datasets to improve decision-making in various stages of the AI pipeline
  - with other packages such as Seaborn and Pandas

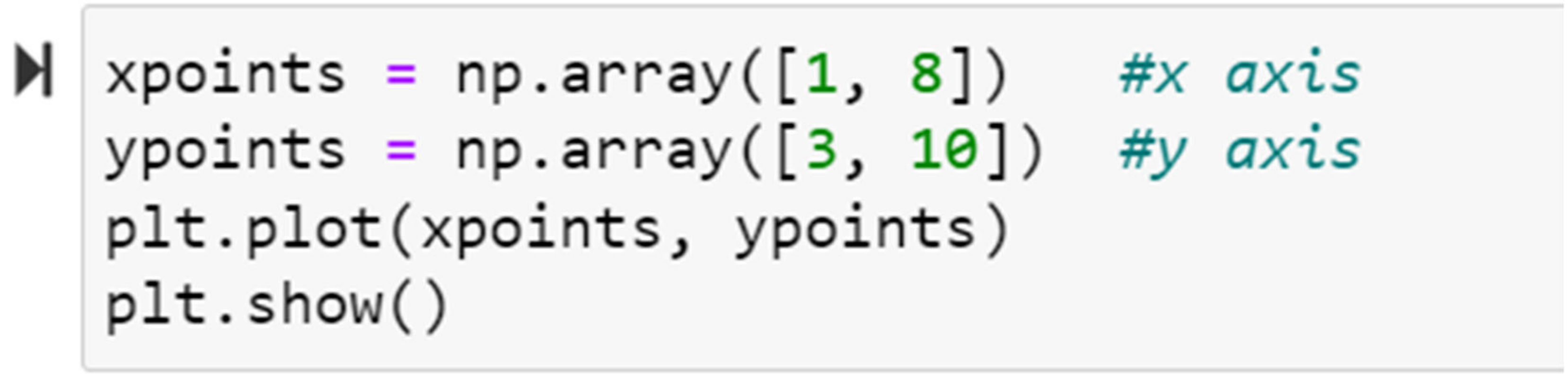
Most of the Matplotlib utilities lies under the pyplot submodule

- normally import it as plt

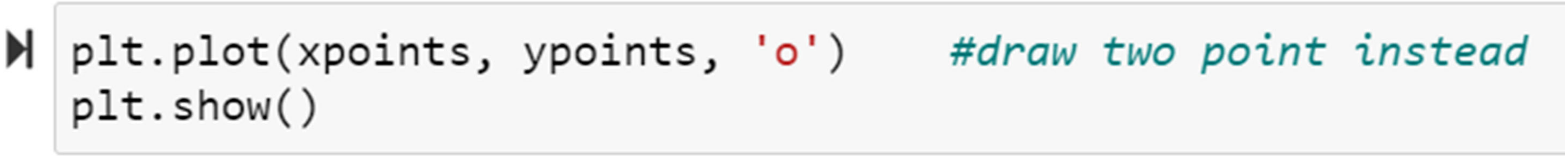


```
import matplotlib.pyplot as plt
import numpy as np
```

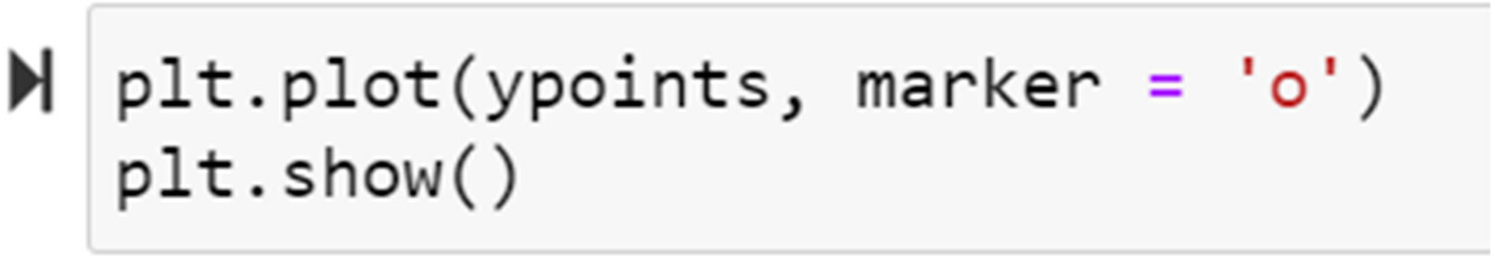
Finger Exercises



```
xpoints = np.array([1, 8])   #x axis
ypoints = np.array([3, 10])  #y axis
plt.plot(xpoints, ypoints)
plt.show()
```



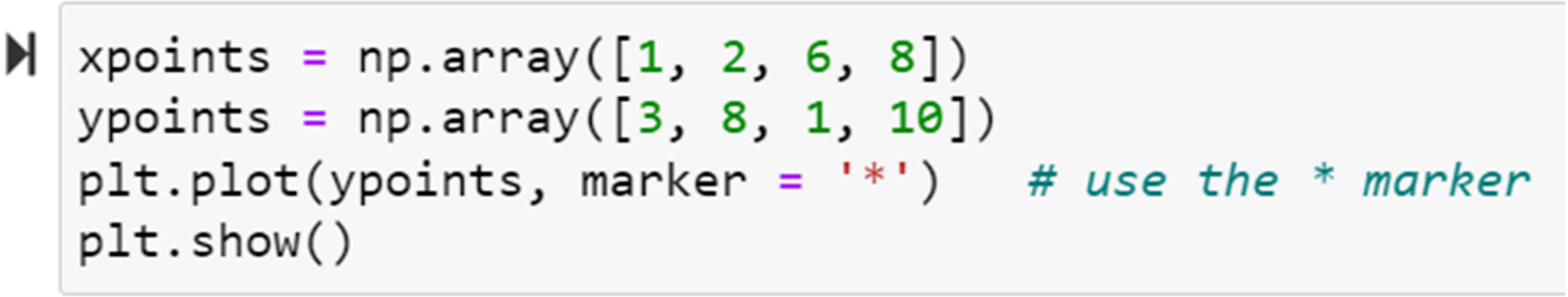
```
plt.plot(xpoints, ypoints, 'o')    #draw two point instead
plt.show()
```



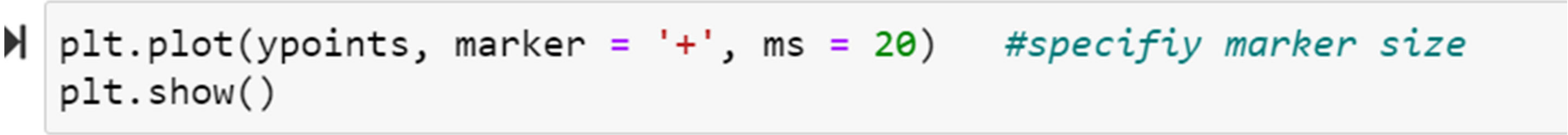
```
plt.plot(ypoints, marker = 'o')
plt.show()
```

Matplotlib - Markers

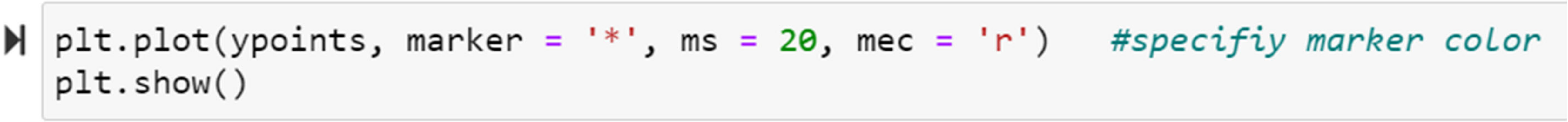
Marker choices



```
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = '*')   # use the * marker
plt.show()
```



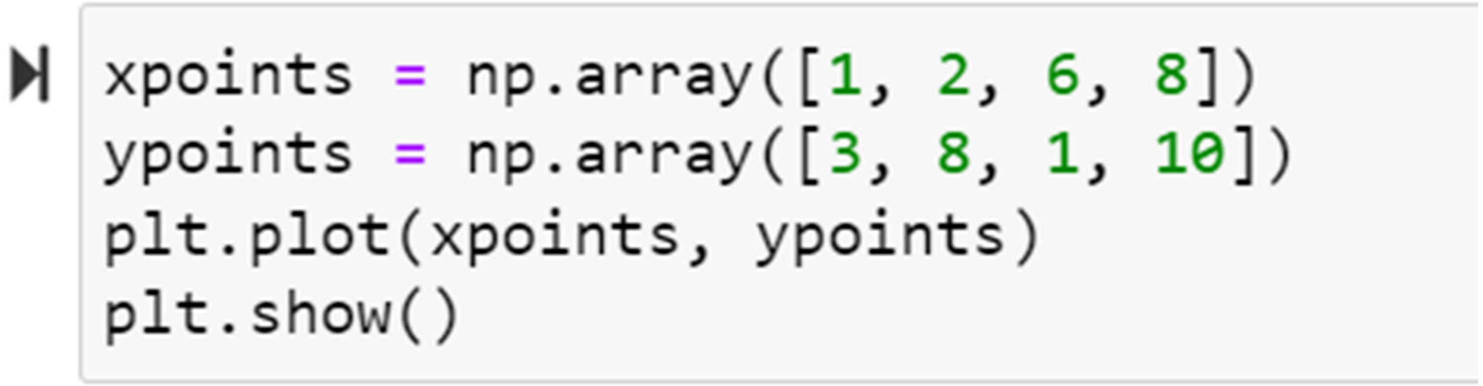
```
plt.plot(ypoints, marker = '+', ms = 20)   #specifiy marker size
plt.show()
```



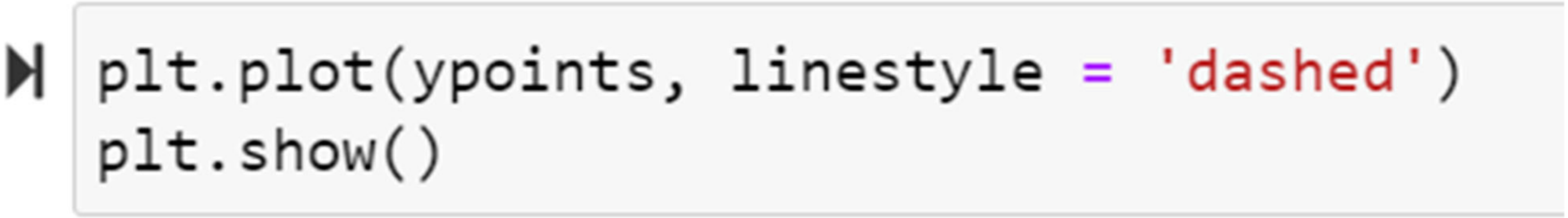
```
plt.plot(ypoints, marker = '*', ms = 20, mec = 'r')   #specifiy marker color
plt.show()
```

Matplotlib – Line types

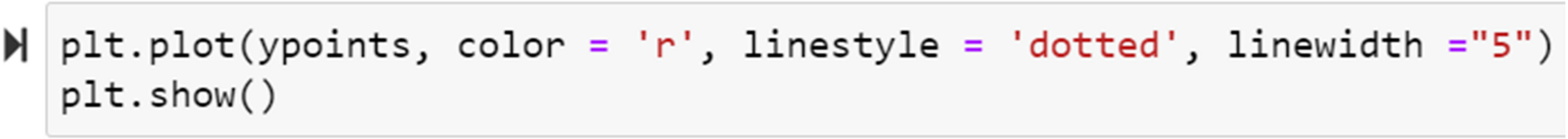
Line types



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



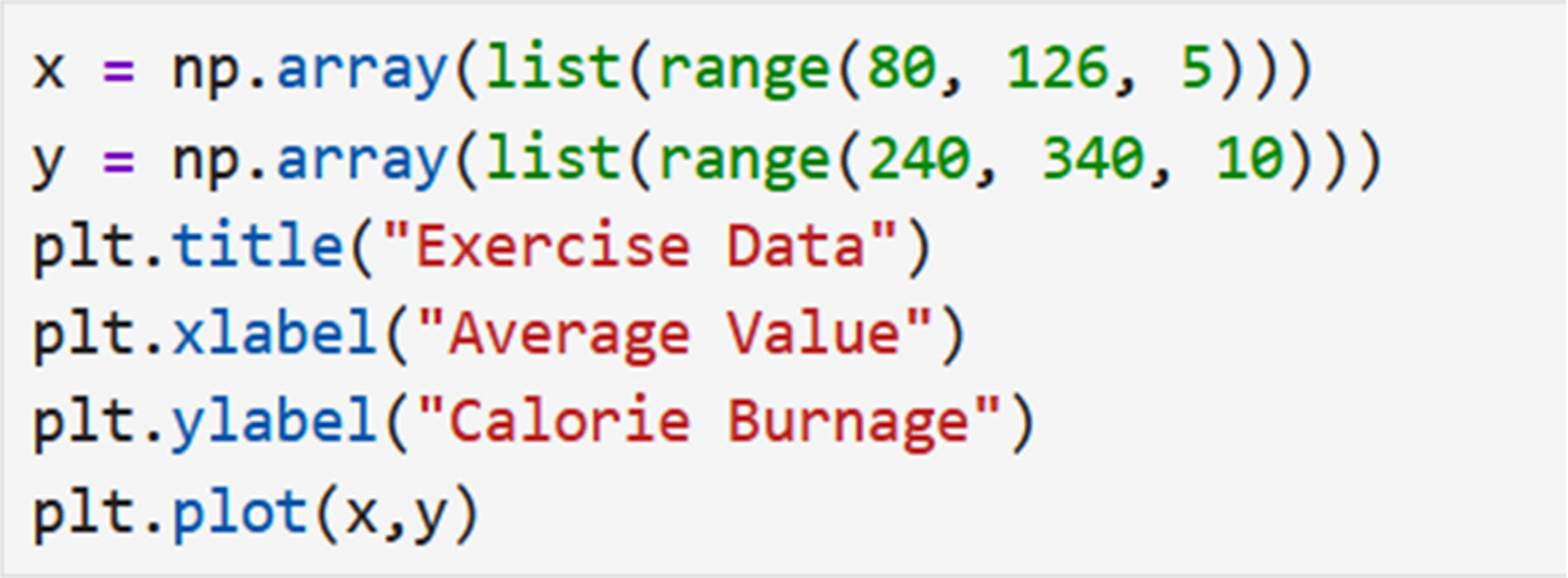
```
plt.plot(ypoints, linestyle = 'dashed')
plt.show()
```



```
plt.plot(ypoints, color = 'r', linestyle = 'dotted', linewidth ="5")
plt.show()
```

Matplotlib – Label

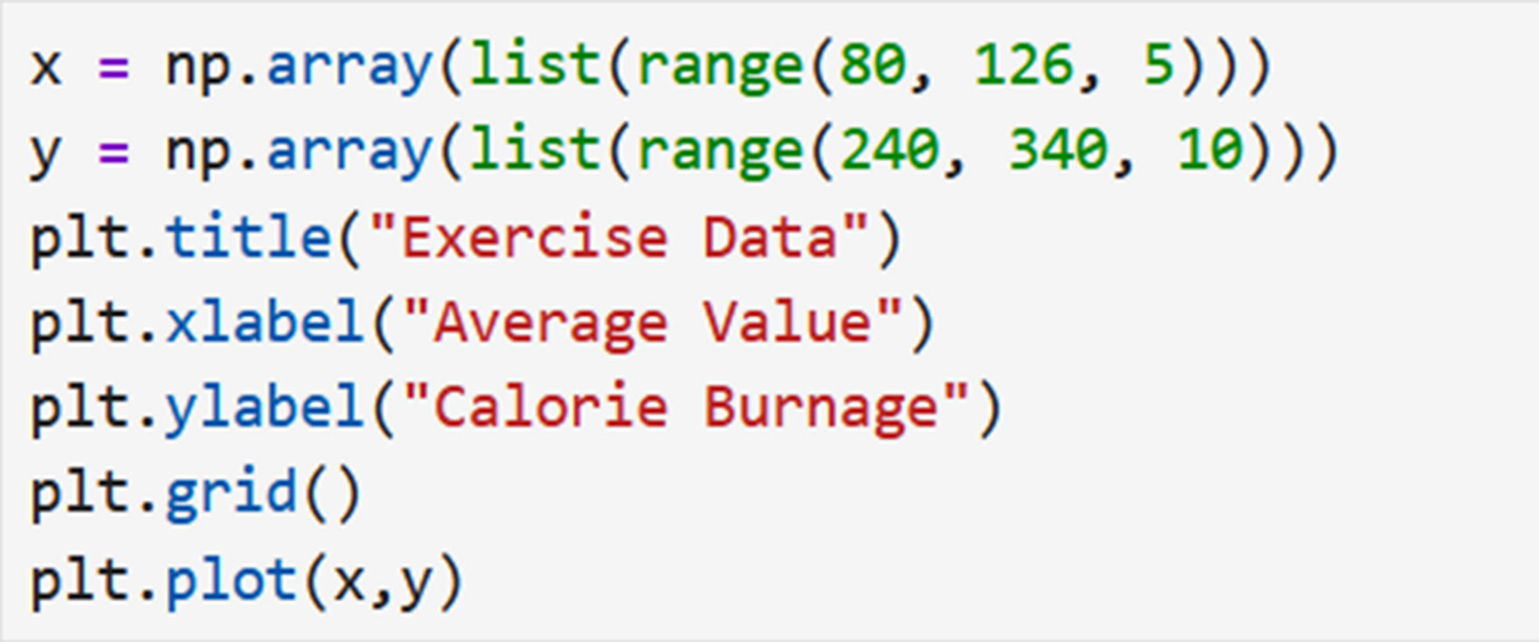
Labels



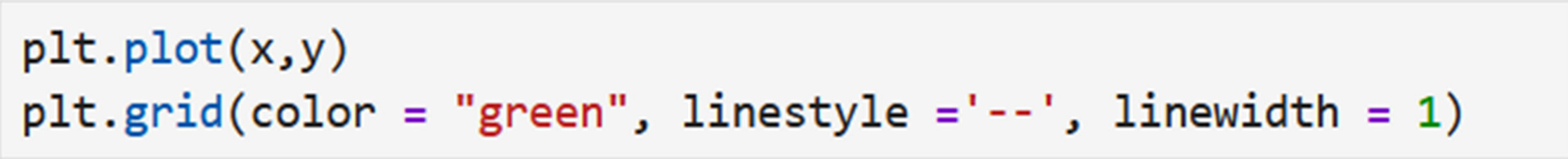
```
x = np.array(list(range(80, 126, 5)))
y = np.array(list(range(240, 340, 10)))
plt.title("Exercise Data")
plt.xlabel("Average Value")
plt.ylabel("Calorie Burnage")
plt.plot(x,y)
```

Matplotlib – Grid

Grid



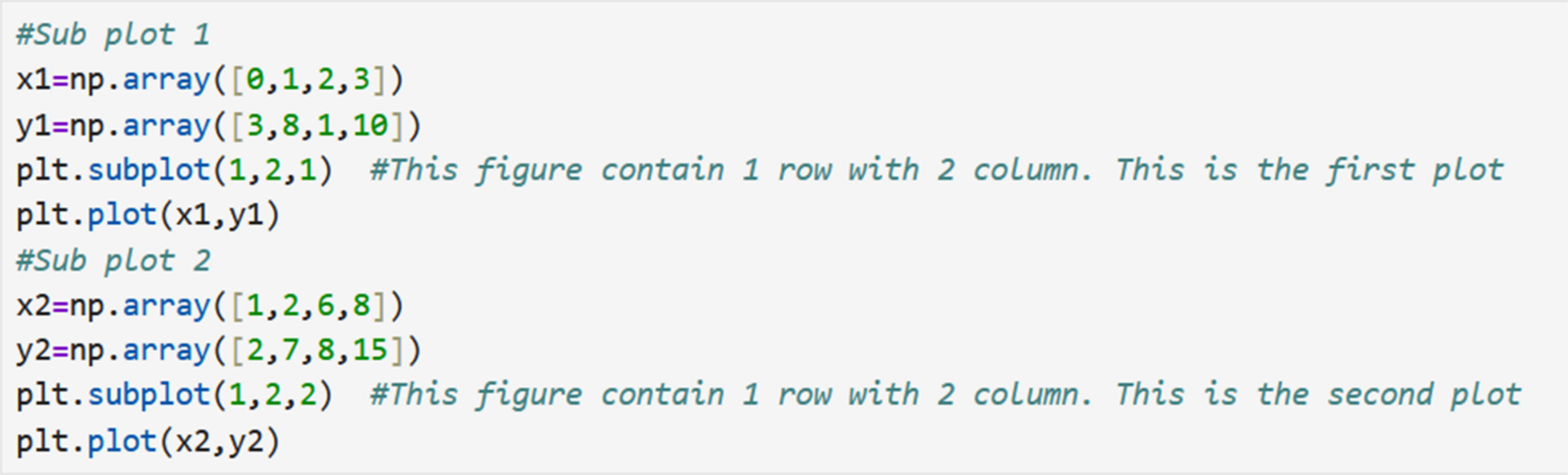
```
x = np.array(list(range(80, 126, 5)))
y = np.array(list(range(240, 340, 10)))
plt.title("Exercise Data")
plt.xlabel("Average Value")
plt.ylabel("Calorie Burnage")
plt.grid()
plt.plot(x,y)
```



```
plt.plot(x,y)
plt.grid(color = "green", linestyle ='--', linewidth = 1)
```

Matplotlib – Multi-plot

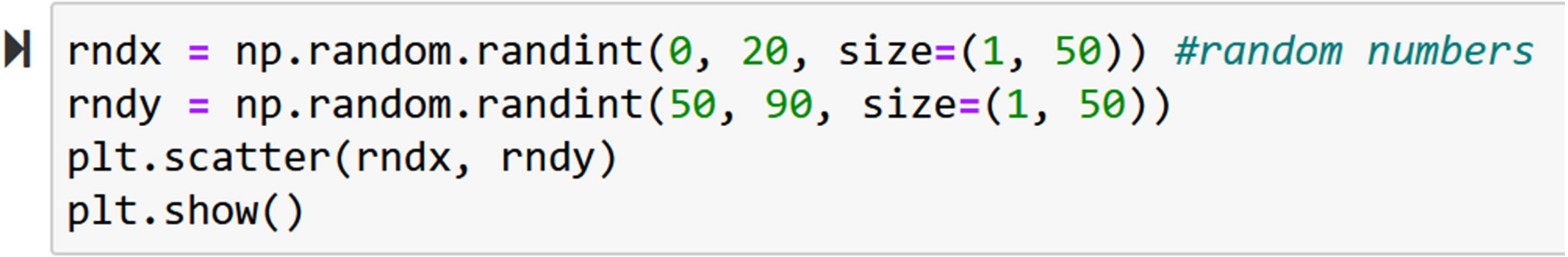
Multi-Plot (with subplots)



```
#Sub plot 1
x1=np.array([0,1,2,3])
y1=np.array([3,8,1,10])
plt.subplot(1,2,1)  #This figure contain 1 row with 2 column. This is the first plot
plt.plot(x1,y1)
#Sub plot 2
x2=np.array([1,2,6,8])
y2=np.array([2,7,8,15])
plt.subplot(1,2,2)  #This figure contain 1 row with 2 column. This is the second plot
plt.plot(x2,y2)
```

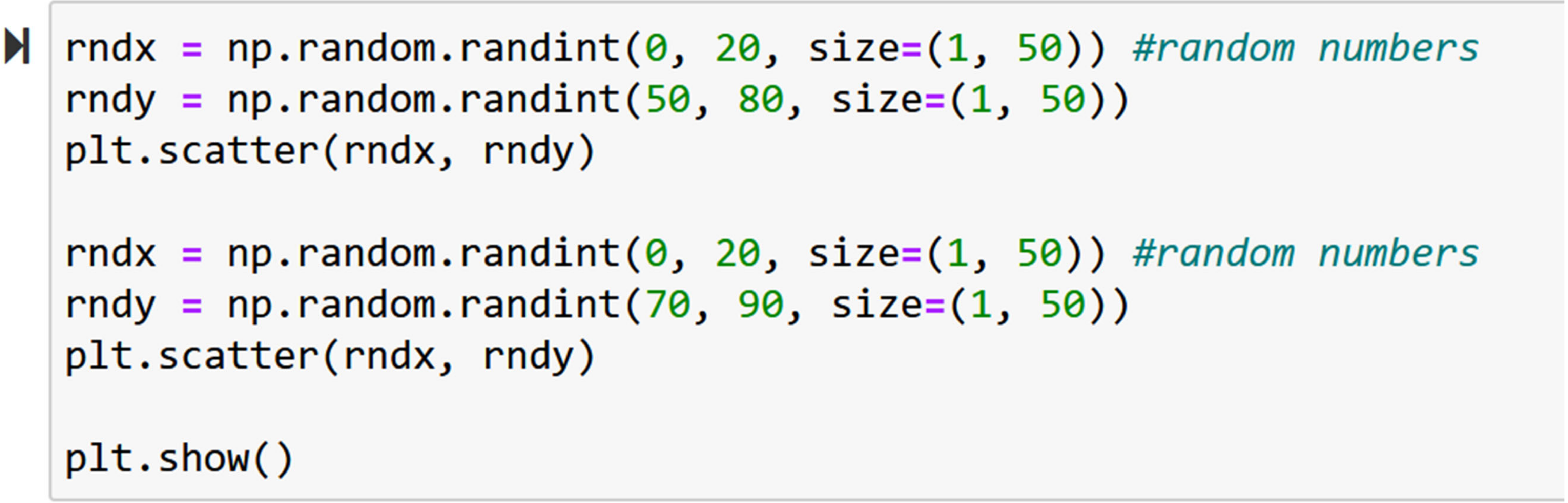
Matplotlib – Scatter Plot

Scatter-plots



```
rndx = np.random.randint(0, 20, size=(1, 50)) #random numbers
rndy = np.random.randint(50, 90, size=(1, 50))
plt.scatter(rndx, rndy)
plt.show()
```

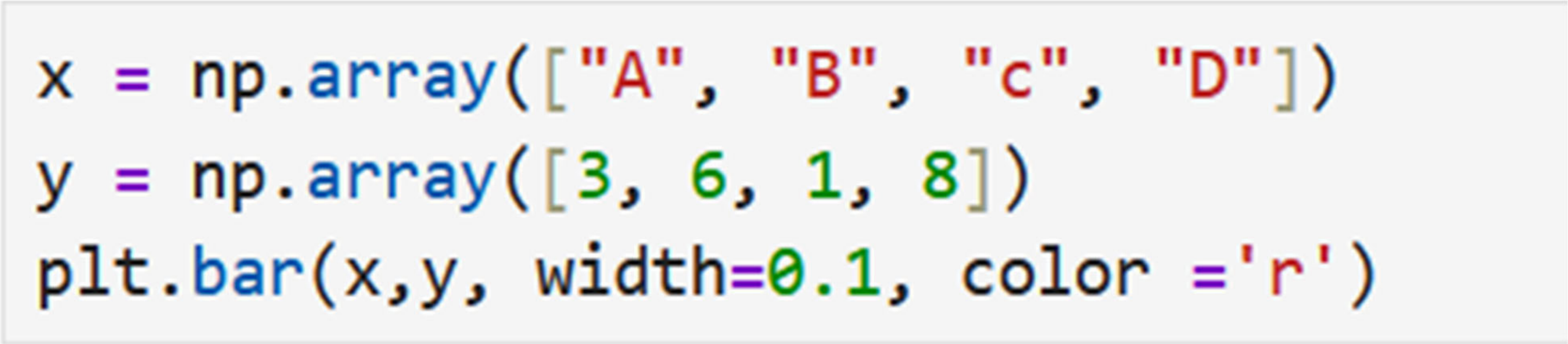
Multi scatter Plot



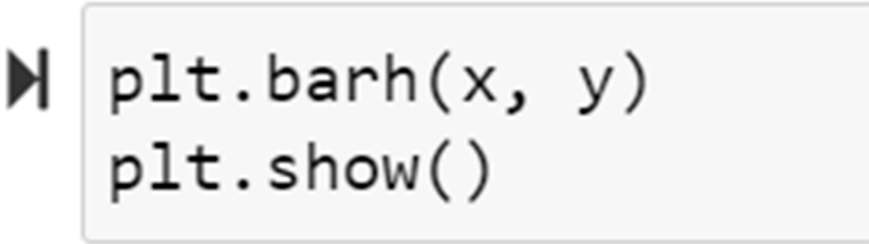
```
rndx = np.random.randint(0, 20, size=(1, 50)) #random numbers
rndy = np.random.randint(50, 80, size=(1, 50))
plt.scatter(rndx, rndy)

rndx = np.random.randint(0, 20, size=(1, 50)) #random numbers
rndy = np.random.randint(70, 90, size=(1, 50))
plt.scatter(rndx, rndy)

plt.show()
```

Matplotlib – Bar graph

Bar Graph

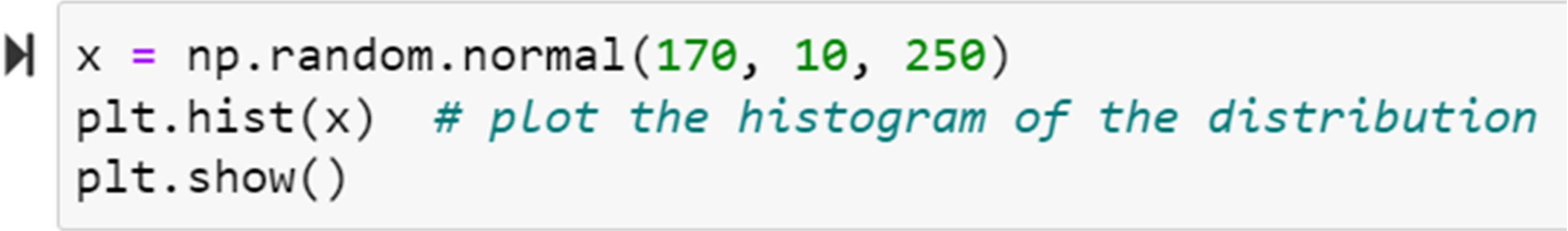


```
x = np.array(["A", "B", "c", "D"])
y = np.array([3, 6, 1, 8])
plt.bar(x,y, width=0.1, color ='r')
```



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

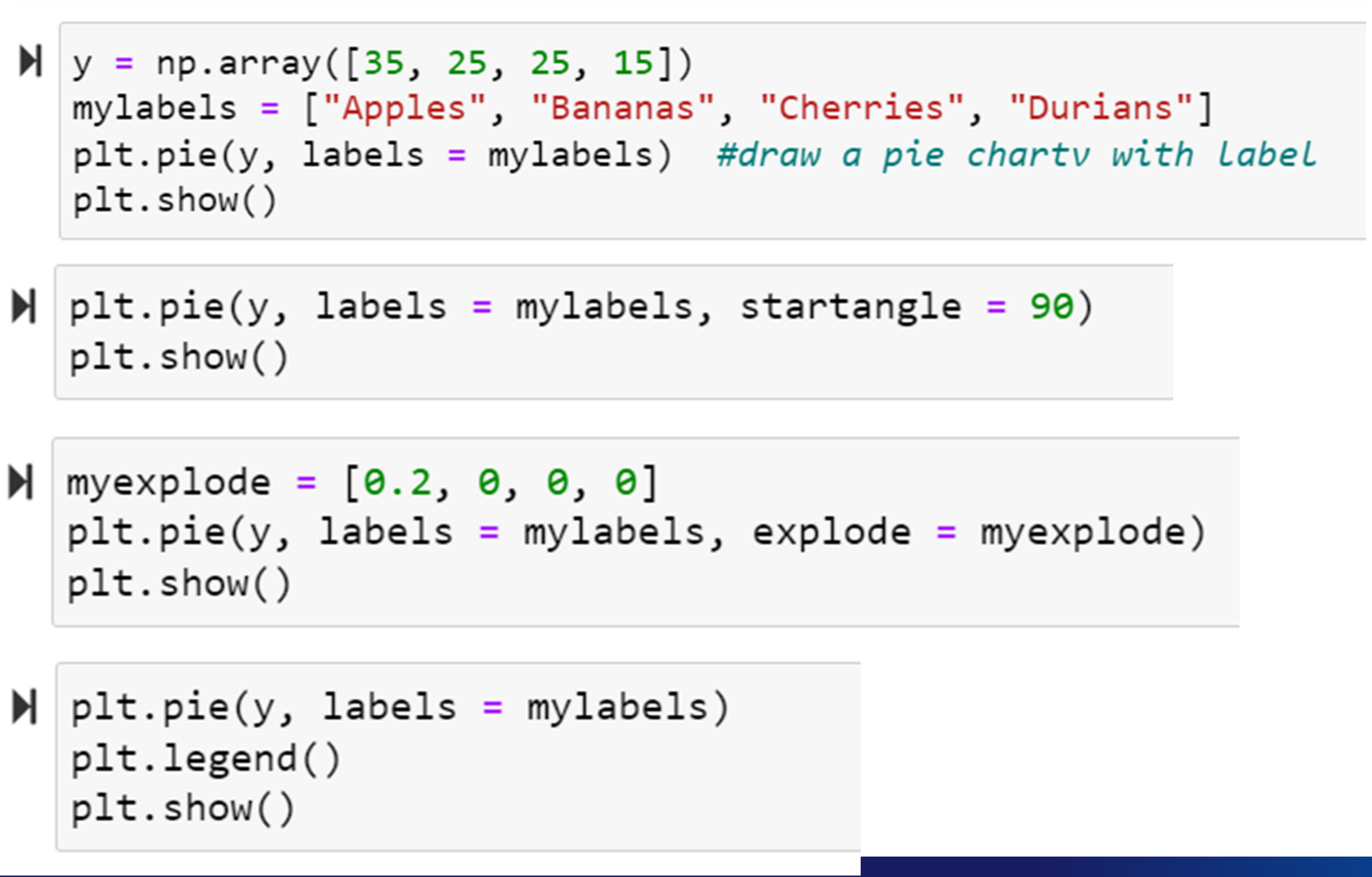
Histogram



```
x = np.random.normal(170, 10, 250)
plt.hist(x)  # plot the histogram of the distribution
plt.show()
```

Matplotlib – Pie Chart

Pie Chart



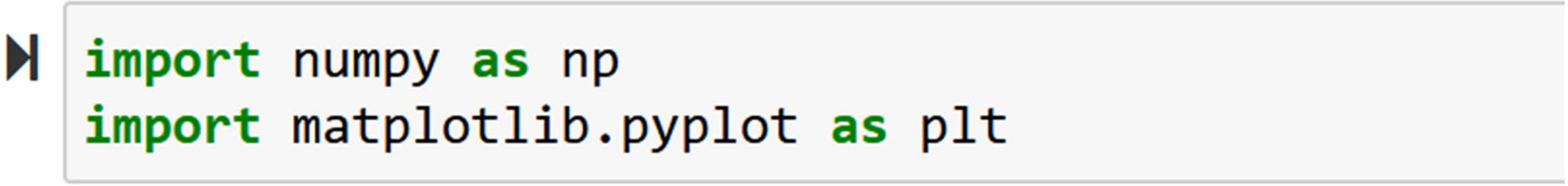
```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Durians"]
plt.pie(y, labels = mylabels)  #draw a pie chartv with label
plt.show()
```

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

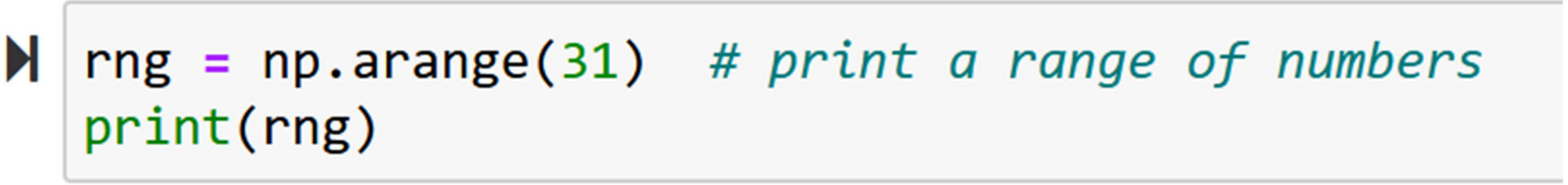
```
myexplode = [0.2, 0, 0, 0]
plt.pie(y, labels = mylabels, explode = myexplode)
plt.show()
```

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

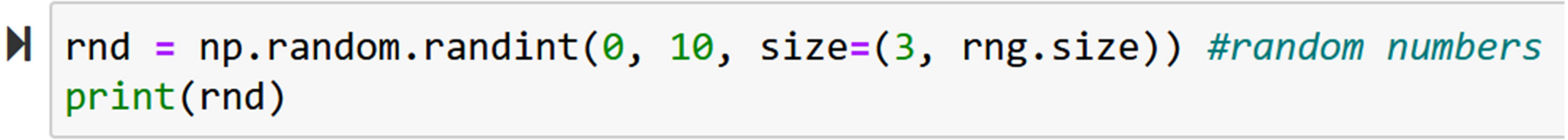
Finger Exercise: NumPy & Matplotlib



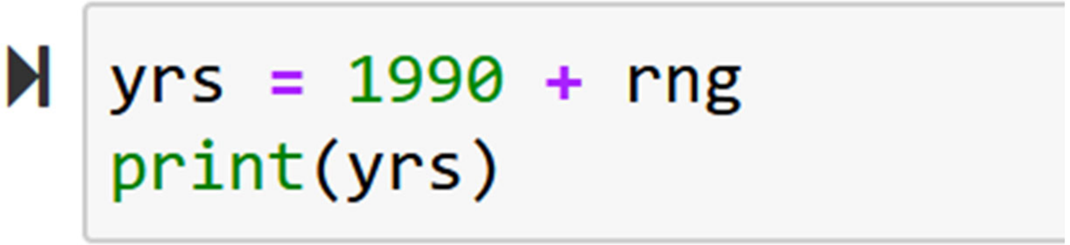
```python
import numpy as np
import matplotlib.pyplot as plt
```



```python
rng = np.arange(31)  # print a range of numbers
print(rng)
```

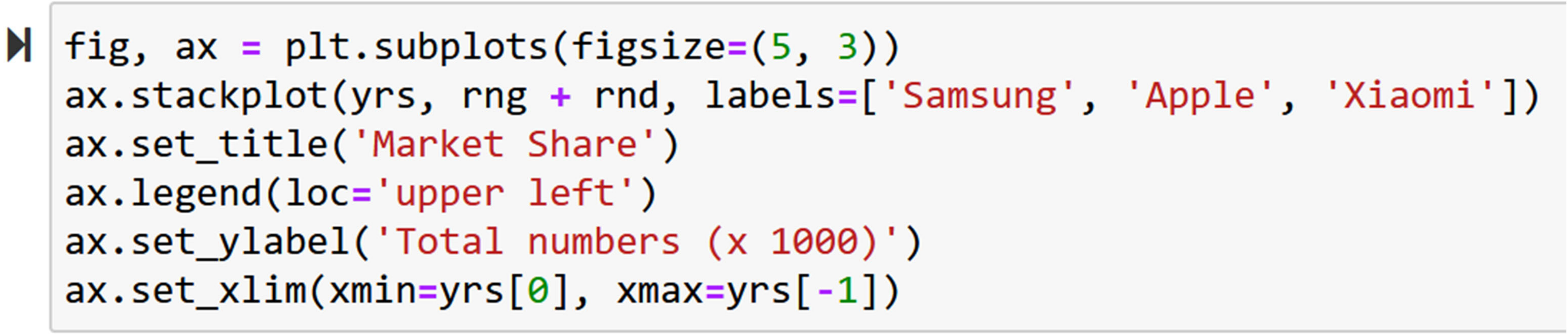


```python
rnd = np.random.randint(0, 10, size=(3, rng.size)) #random numbers
print(rnd)
```



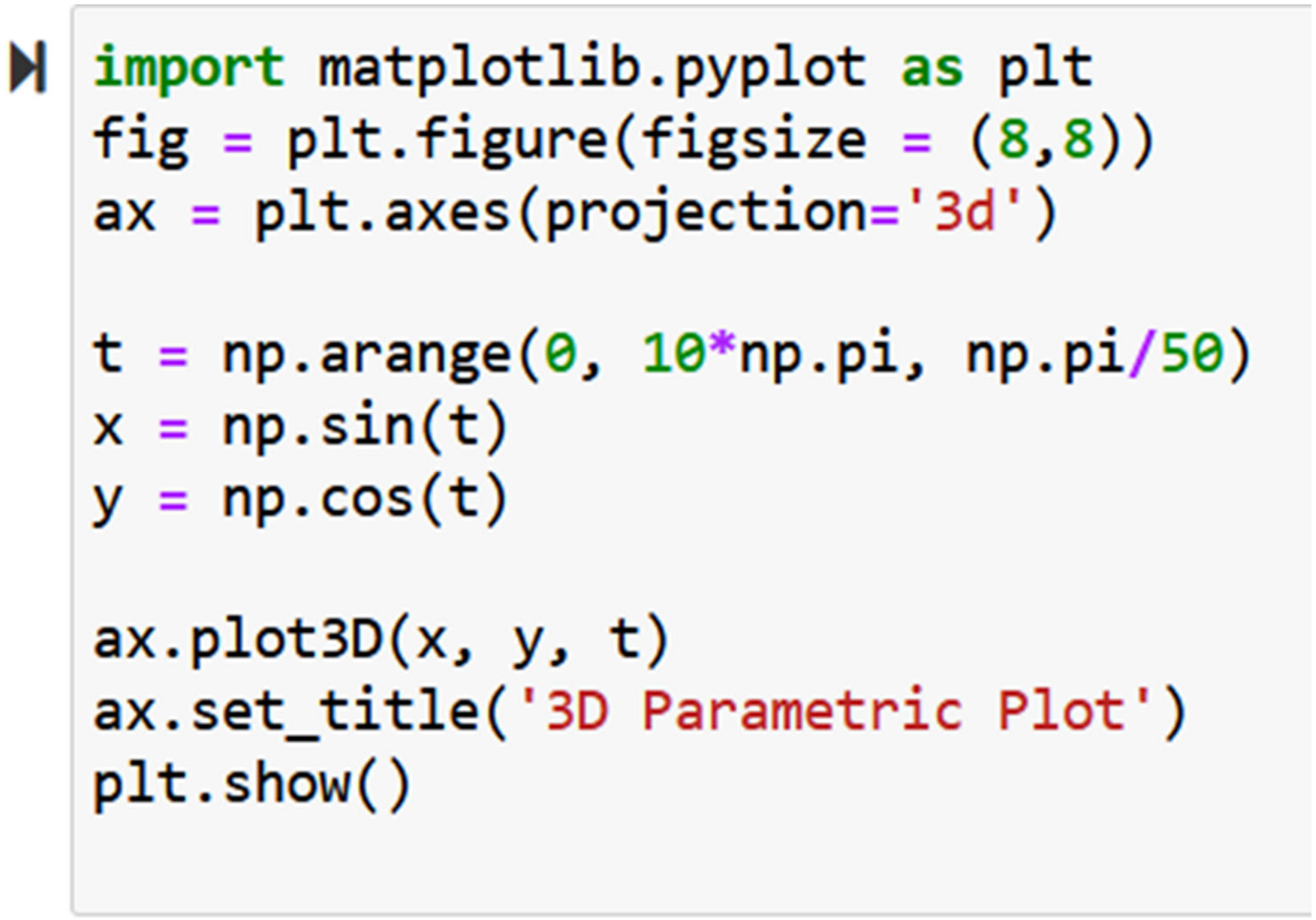
```python
yrs = 1990 + rng
print(yrs)
```

Finger Exercise: NumPy & Matplotlib (cont.)

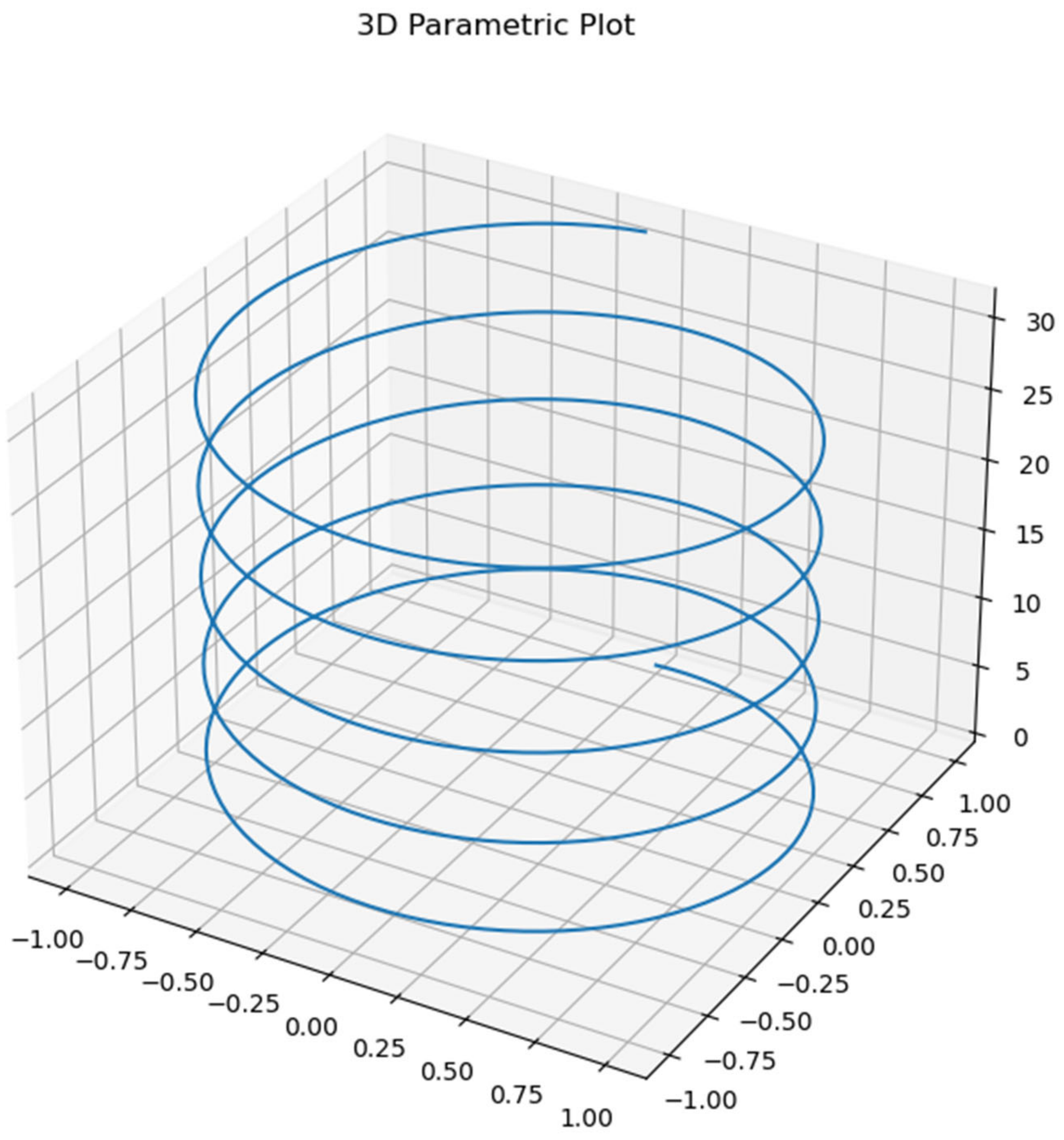


```
fig, ax = plt.subplots(figsize=(5, 3))
ax.stackplot(yrs, rng + rnd, labels=['Samsung', 'Apple', 'Xiaomi'])
ax.set_title('Market Share')
ax.legend(loc='upper left')
ax.set_ylabel('Total numbers (x 1000)')
ax.set_xlim(xmin=yrs[0], xmax=yrs[-1])
```

Aside: 3D plotting

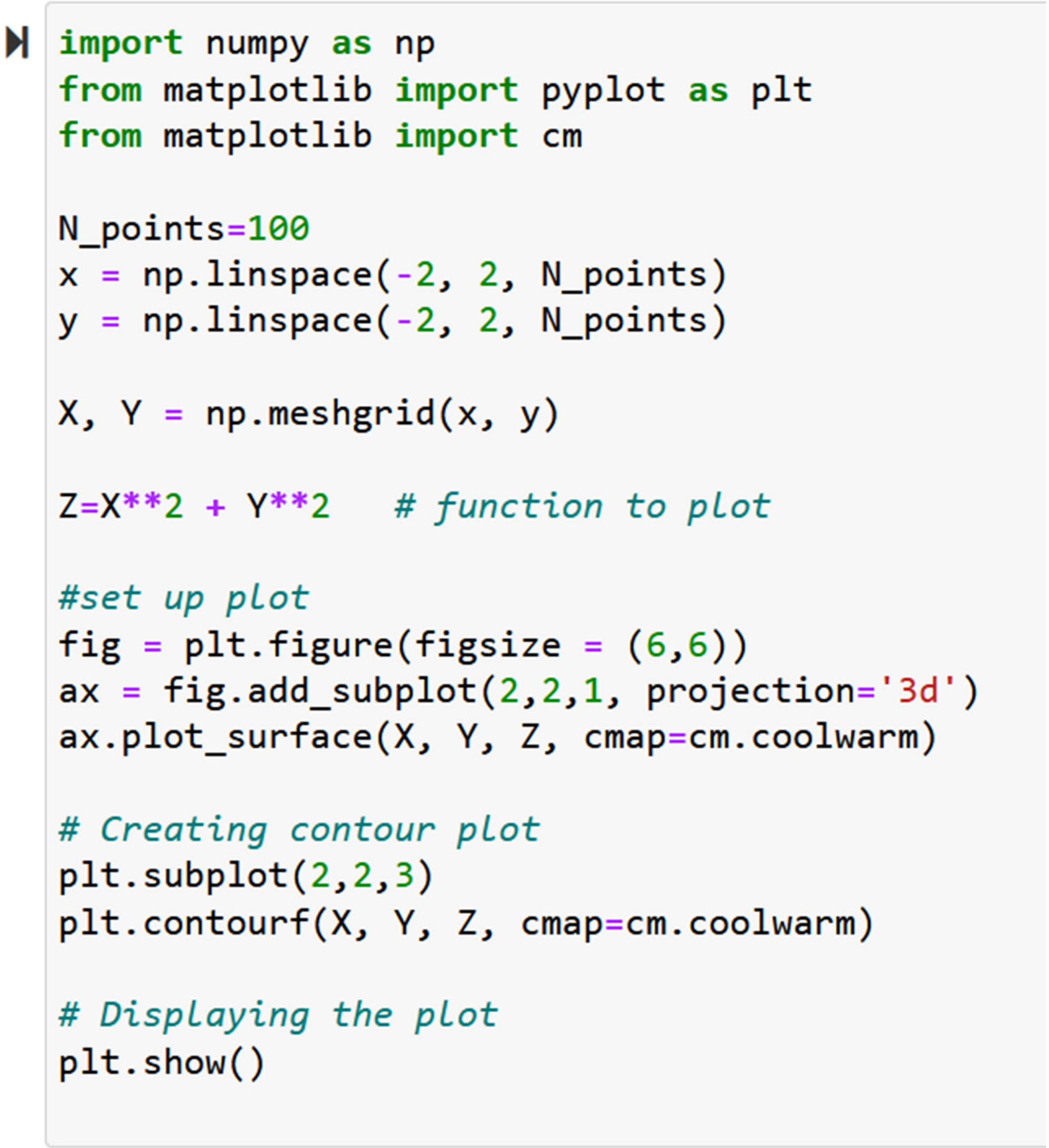


```python
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,8))
ax = plt.axes(projection='3d')

t = np.arange(0, 10*np.pi, np.pi/50)
x = np.sin(t)
y = np.cos(t)

ax.plot3D(x, y, t)
ax.set_title('3D Parametric Plot')
plt.show()
```

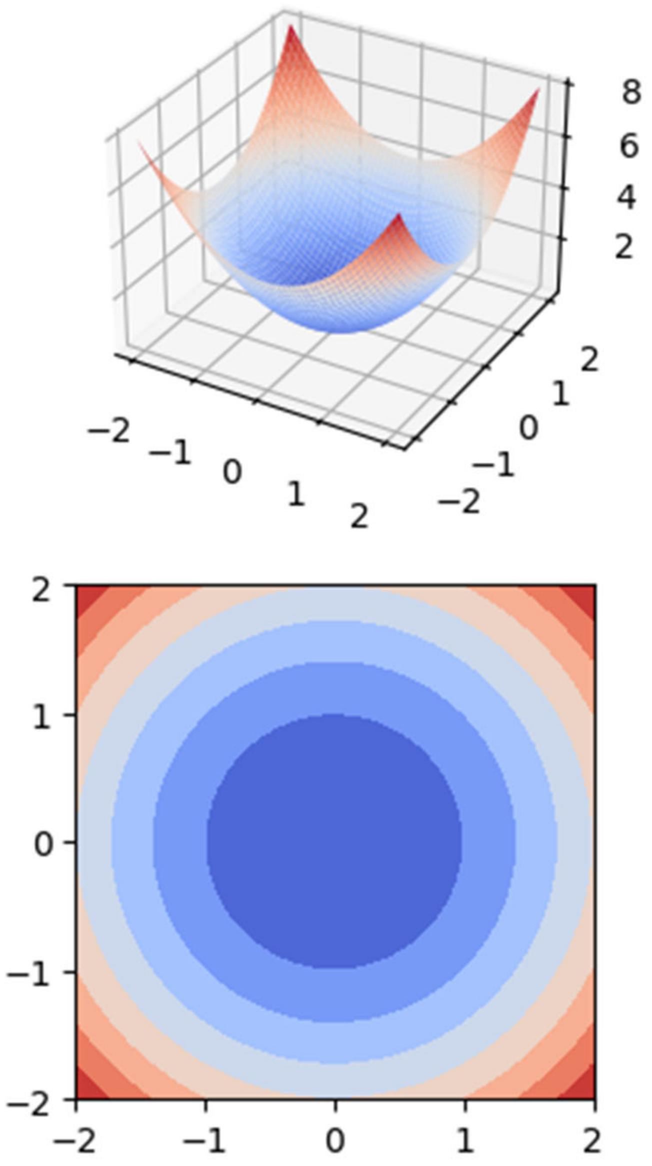


3D Parametric Plot

Aside:3D Surface plotting



```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import cm

N_points=100
x = np.linspace(-2, 2, N_points)
y = np.linspace(-2, 2, N_points)

X, Y = np.meshgrid(x, y)

Z=X**2 + Y**2   # function to plot

#set up plot
fig = plt.figure(figsize = (6,6))
ax = fig.add_subplot(2,2,1, projection='3d')
ax.plot_surface(X, Y, Z, cmap=cm.coolwarm)

# Creating contour plot
plt.subplot(2,2,3)
plt.contourf(X, Y, Z, cmap=cm.coolwarm)

# Displaying the plot
plt.show()
```



8
6
4
2
−2
−1
0
1
2
2
1
0
−1
−2
2
1
0
−1
−2
−2
−1
0
1
2