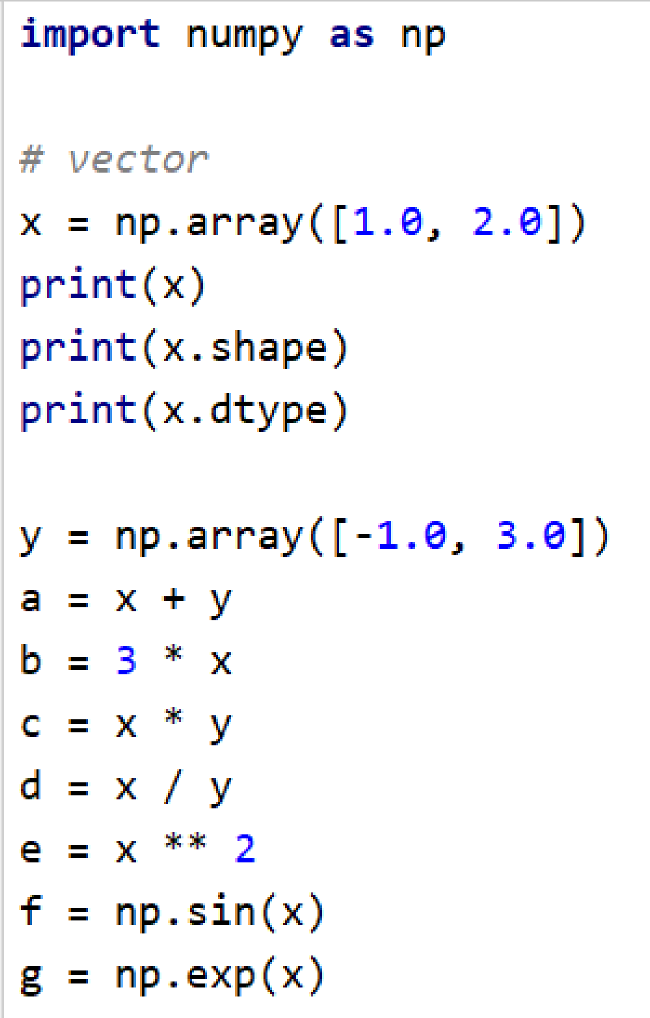
# 인공지능프로그래밍

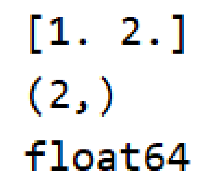
게임콘텐츠학과 박경수

https://github.com/ggorr/Machine-Learning/tree/master/Python

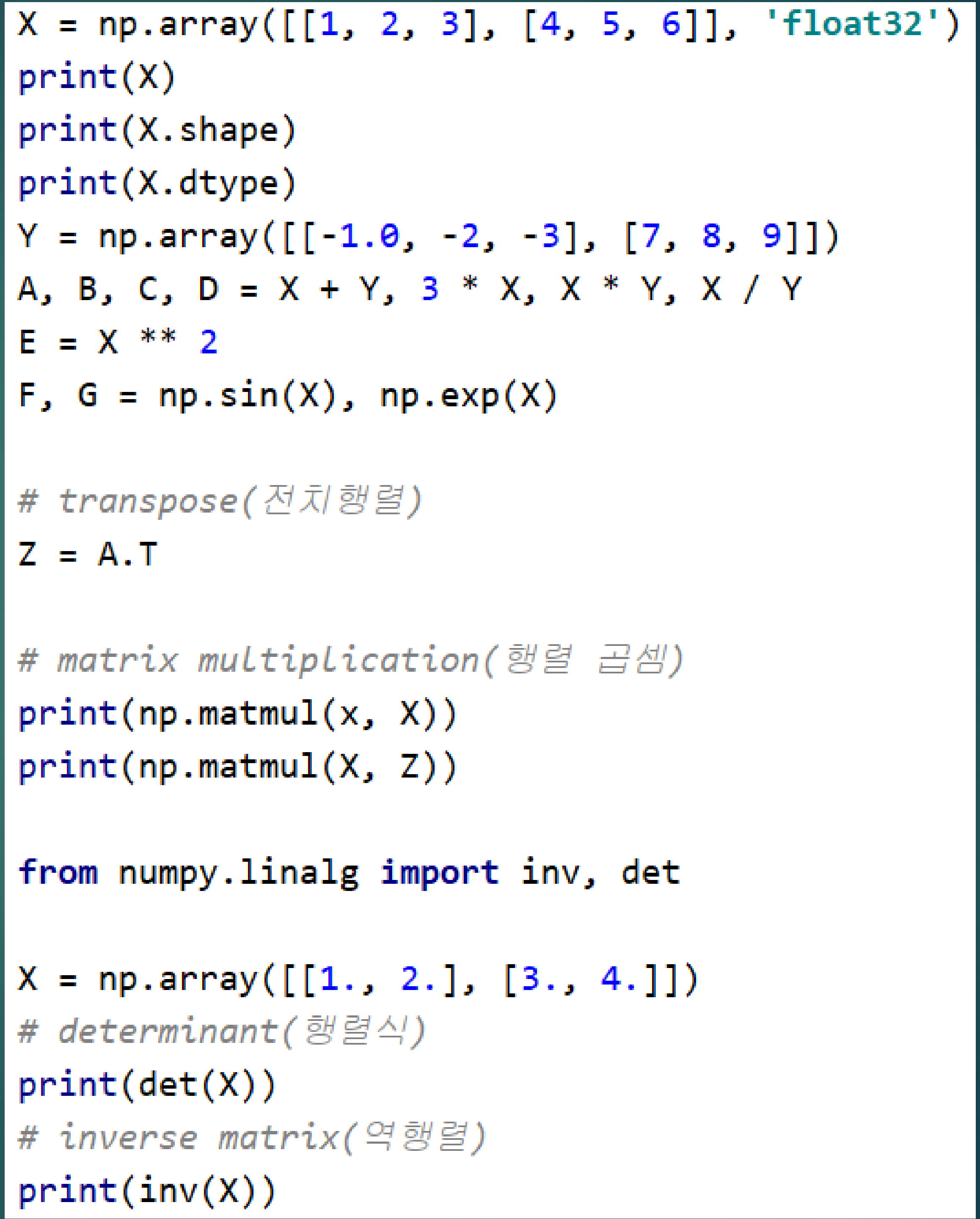
# Matrix



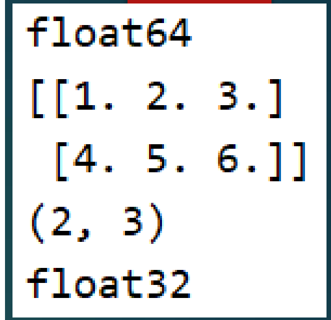
```python
import numpy as np

# vector
x = np.array([1.0, 2.0])
print(x)
print(x.shape)
print(x.dtype)

y = np.array([-1.0, 3.0])
a = x + y
b = 3 * x
c = x * y
d = x / y
e = x ** 2
f = np.sin(x)
g = np.exp(x)
```



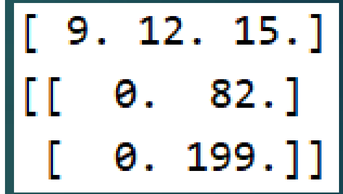
```
[1. 2.]
(2,)
float64
```



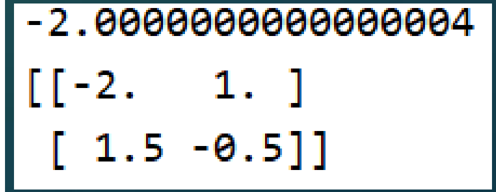
```python
X = np.array([[1, 2, 3], [4, 5, 6]], 'float32')
print(X)
print(X.shape)
print(X.dtype)
Y = np.array([[-1.0, -2, -3], [7, 8, 9]])
A, B, C, D = X + Y, 3 * X, X * Y, X / Y
E = X ** 2
F, G = np.sin(X), np.exp(X)

# transpose(전치행렬)
Z = A.T

# matrix multiplication(행렬 곱셈)
print(np.matmul(x, X))
print(np.matmul(X, Z))

from numpy.linalg import inv, det

X = np.array([[1., 2.], [3., 4.]])
# determinant(행렬식)
print(det(X))
# inverse matrix(역행렬)
print(inv(X))
```



```
float64
[[1. 2. 3.]
 [4. 5. 6.]]
(2, 3)
float32
```

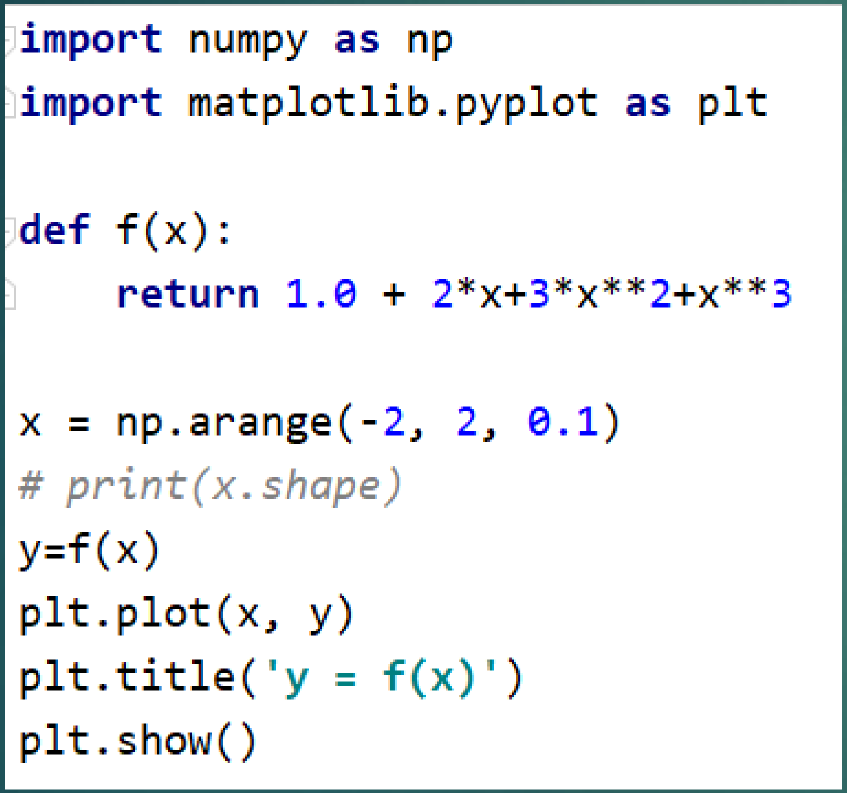


```
[ 9. 12. 15.]
[[  0.  82.]
 [  0. 199.]]
```

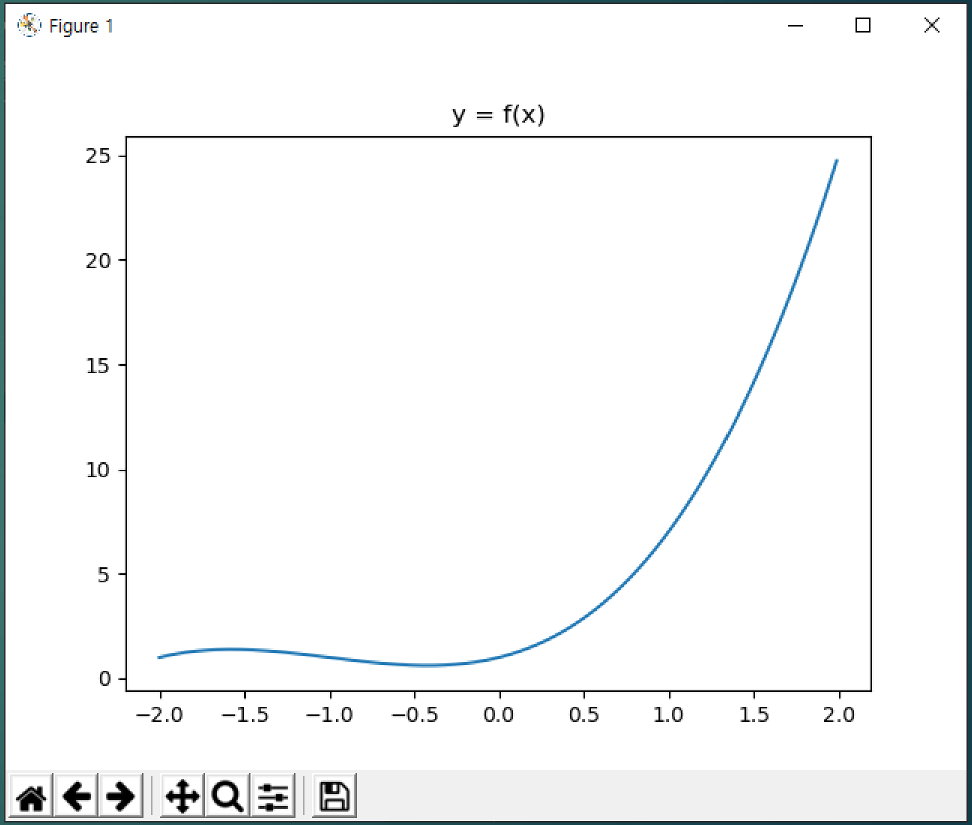


```
-2.0000000000000004
[[-2.   1. ]
 [ 1.5 -0.5]]
```

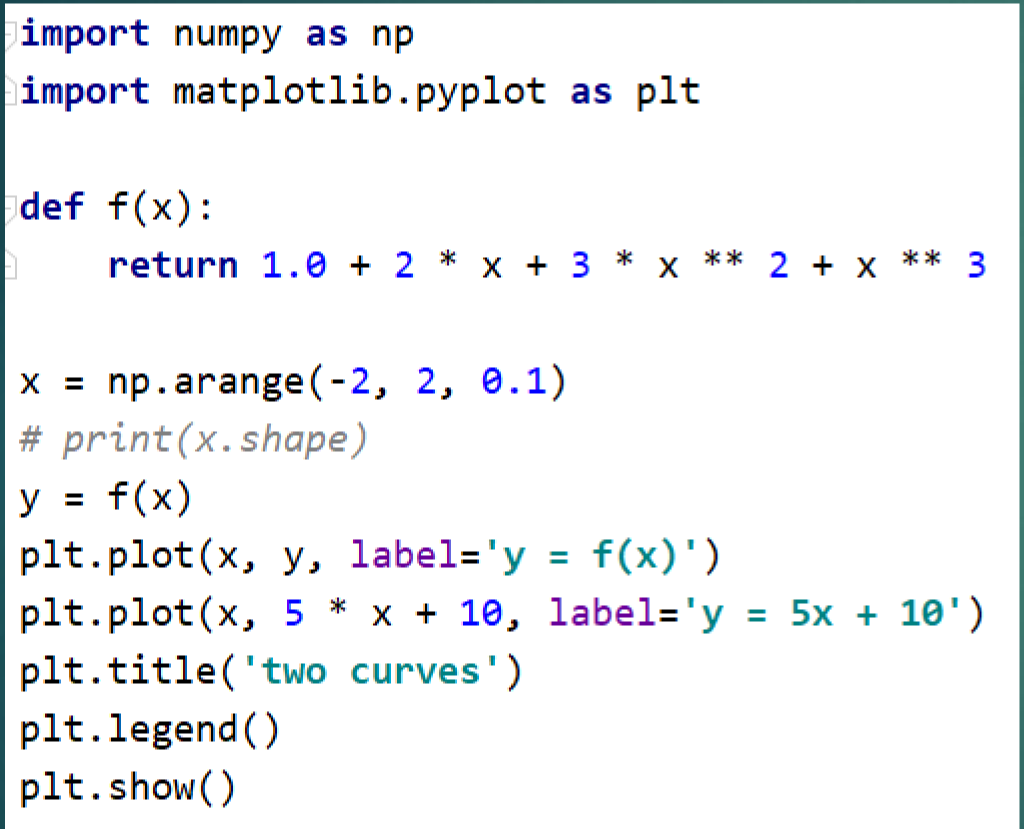
# Plot2D



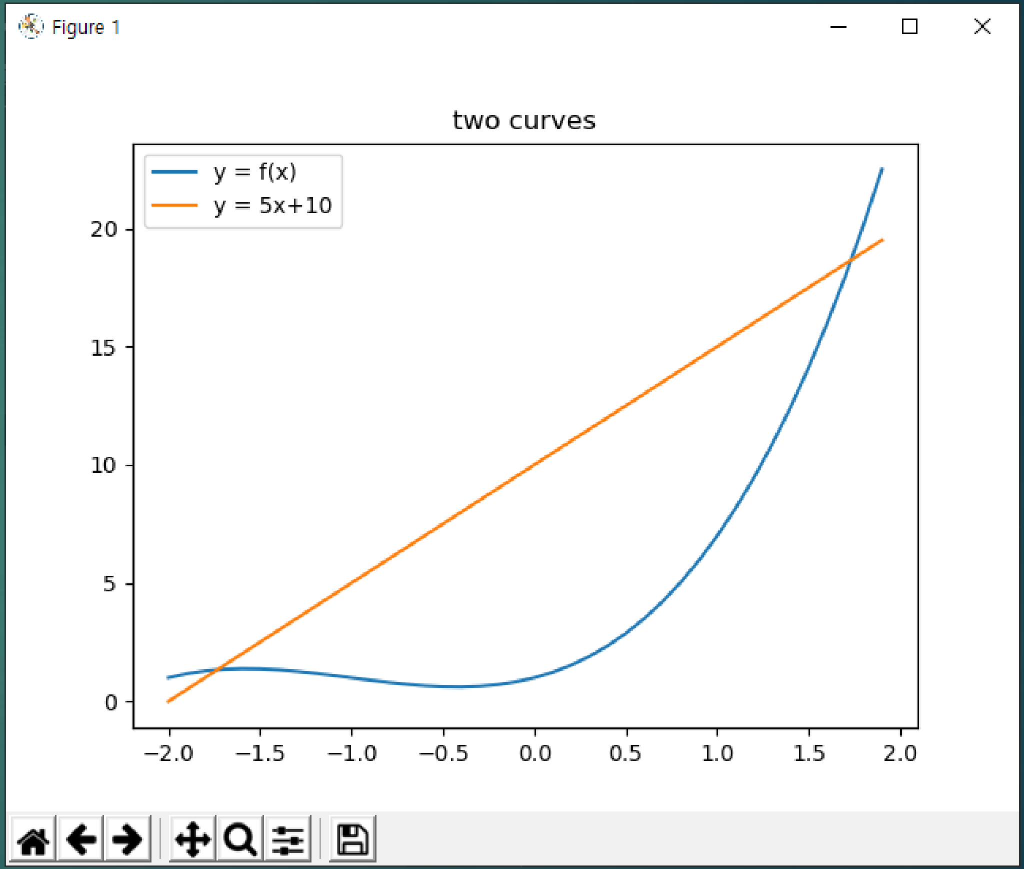
```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return 1.0 + 2*x+3*x**2+x**3

x = np.arange(-2, 2, 0.1)
# print(x.shape)
y=f(x)
plt.plot(x, y)
plt.title('y = f(x)')
plt.show()
```



# Plot2D

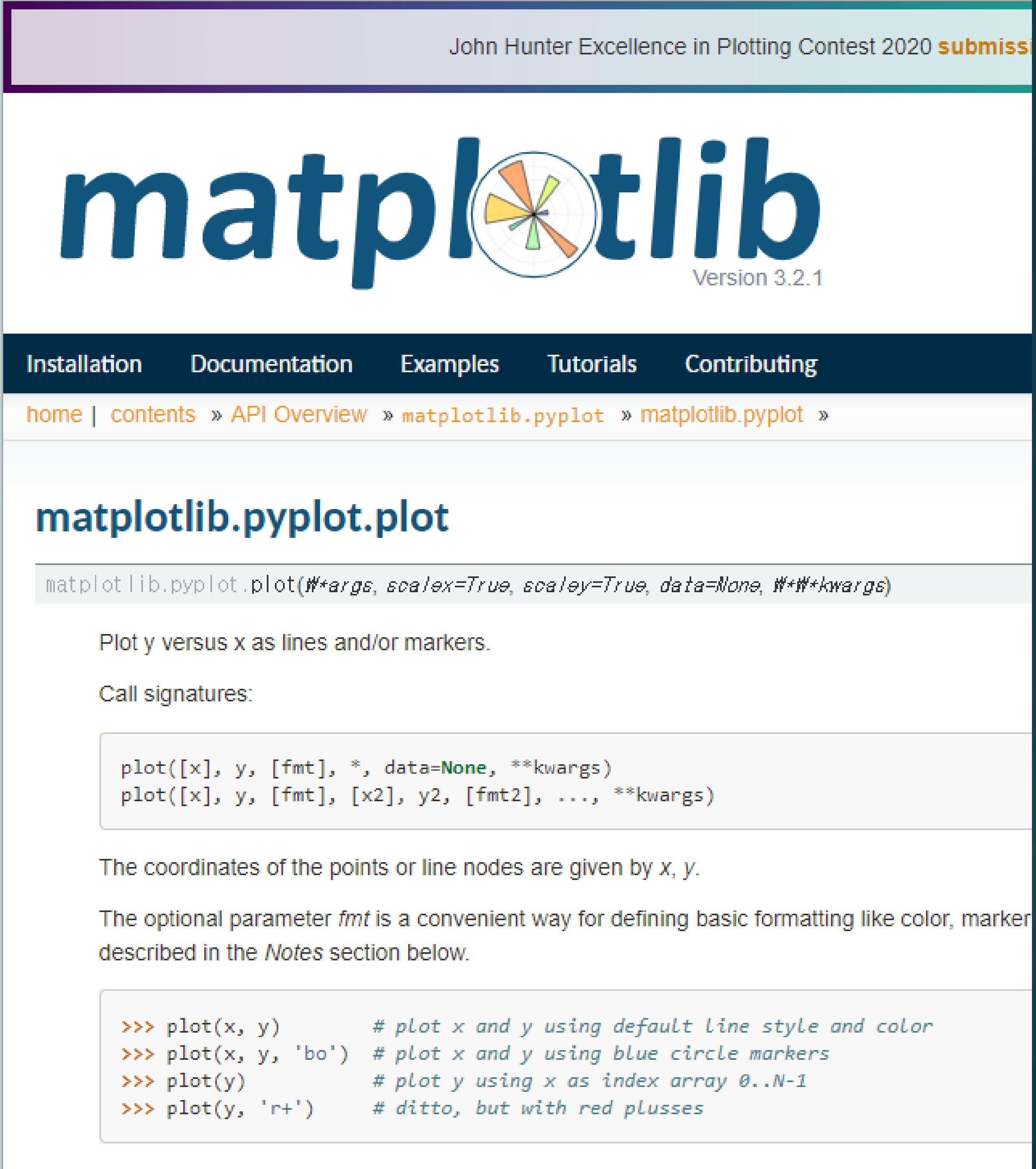


```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return 1.0 + 2 * x + 3 * x ** 2 + x ** 3

x = np.arange(-2, 2, 0.1)
# print(x.shape)
y = f(x)
plt.plot(x, y, label='y = f(x)')
plt.plot(x, 5 * x + 10, label='y = 5x + 10')
plt.title('two curves')
plt.legend()
plt.show()
```



# Plot2D

- web search “matplotlib.pyplot.plot”


John Hunter Excellence in Plotting Contest 2020 submiss
matplotlib
Version 3.2.1
Installation
Documentation
Examples
Tutorials
Contributing
home | contents » API Overview » matplotlib.pyplot » matplotlib.pyplot »

## matplotlib.pyplot.plot

```
matplotlib.pyplot.plot(₩*args, scalex=True, scaley=True, data=None, ₩*₩*kwargs)
```

Plot y versus x as lines and/or markers.

Call signatures:

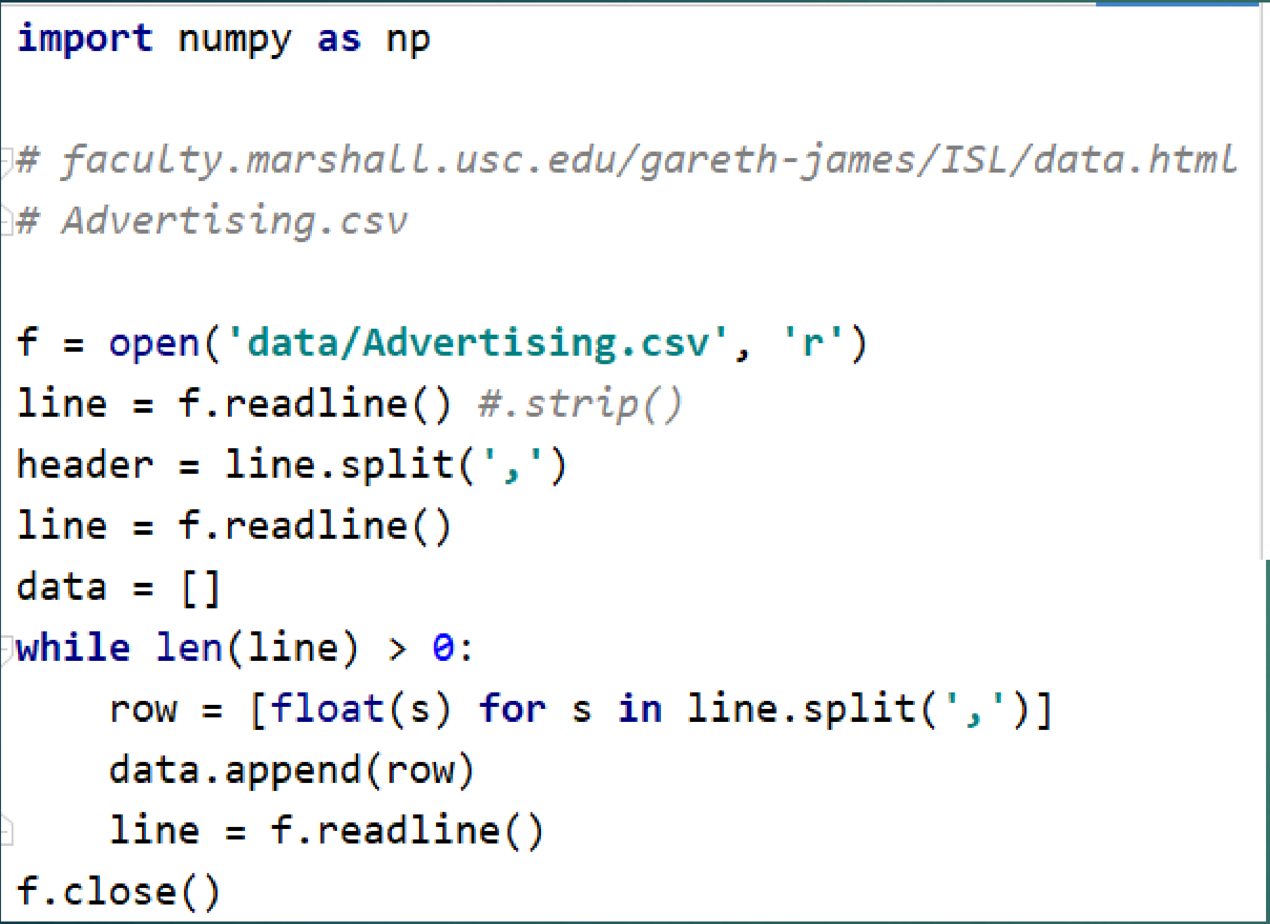
```
plot([x], y, [fmt], *, data=None, **kwargs)
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

The coordinates of the points or line nodes are given by *x*, *y*.

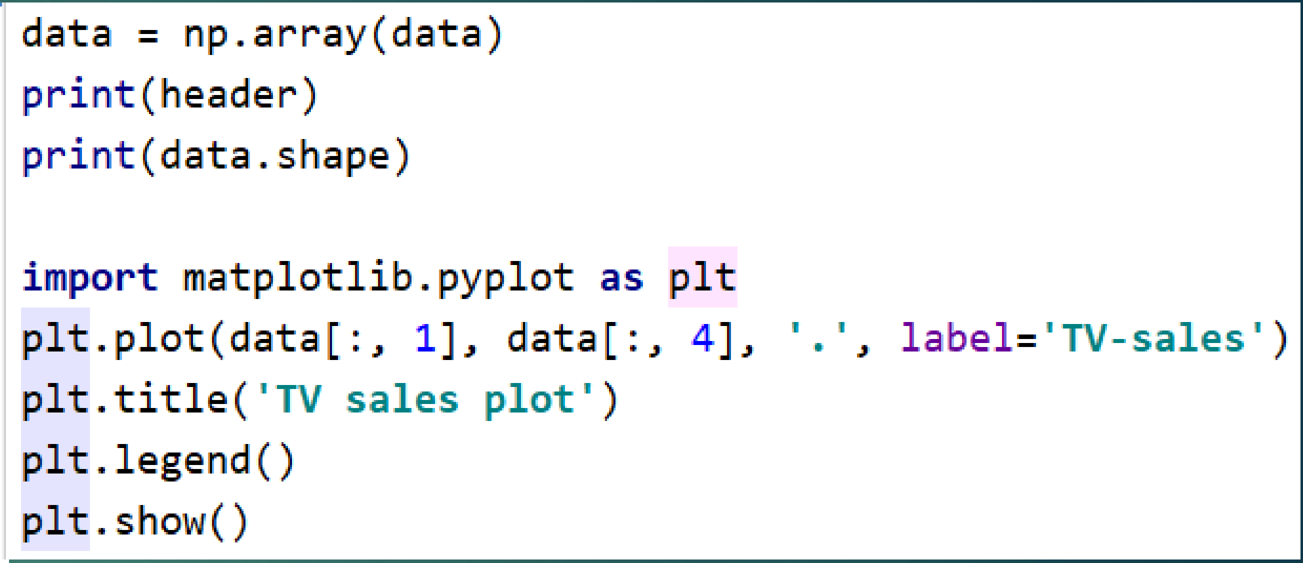
The optional parameter *fmt* is a convenient way for defining basic formatting like color, marker described in the *Notes* section below.

```
>>> plot(x, y)        # plot x and y using default line style and color
>>> plot(x, y, 'bo')  # plot x and y using blue circle markers
>>> plot(y)           # plot y using x as index array 0..N-1
>>> plot(y, 'r+')     # ditto, but with red plusses
```

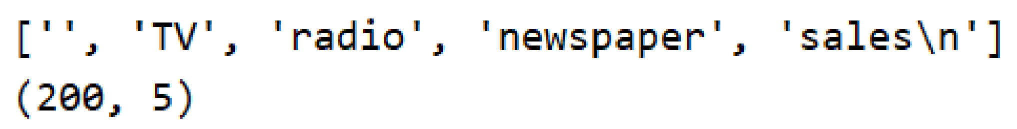
# Advertising



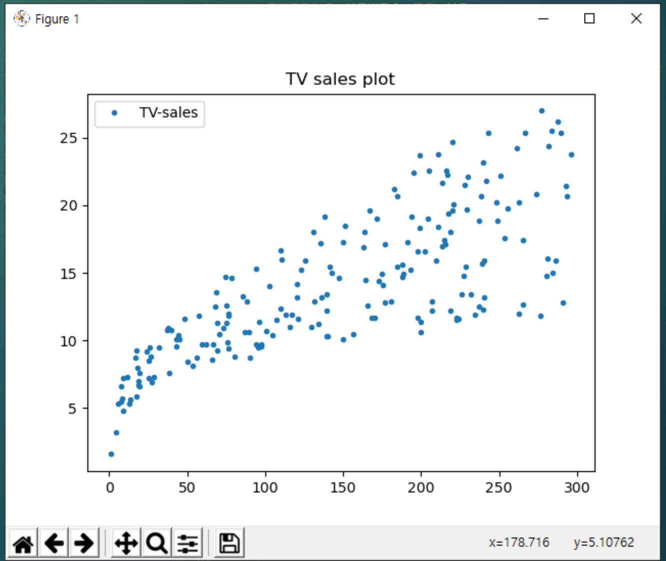
```python
import numpy as np

# faculty.marshall.usc.edu/gareth-james/ISL/data.html
# Advertising.csv

f = open('data/Advertising.csv', 'r')
line = f.readline() #.strip()
header = line.split(',')
line = f.readline()
data = []
while len(line) > 0:
    row = [float(s) for s in line.split(',')]
    data.append(row)
    line = f.readline()
f.close()
```



```python
data = np.array(data)
print(header)
print(data.shape)

import matplotlib.pyplot as plt
plt.plot(data[:, 1], data[:, 4], '.', label='TV-sales')
plt.title('TV sales plot')
plt.legend()
plt.show()
```

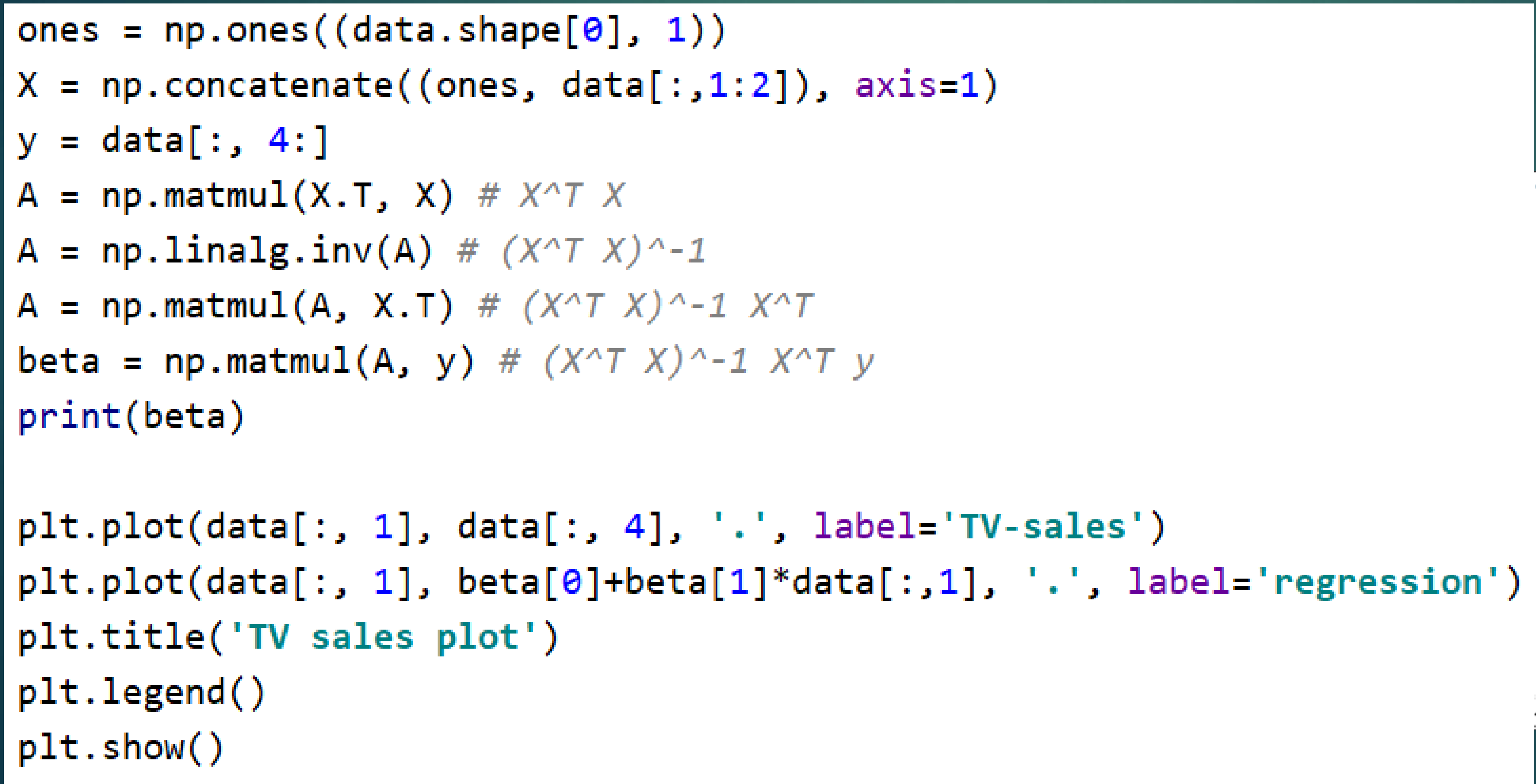


```
['', 'TV', 'radio', 'newspaper', 'sales\n']
(200, 5)
```

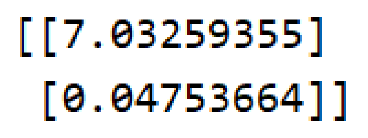


# Advertising - Regression

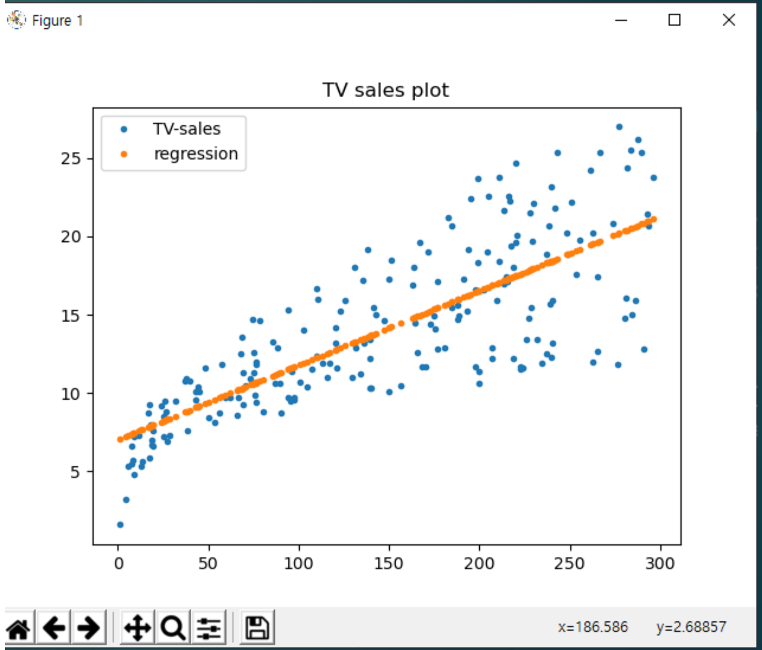
- predictor: TV
- response: sales



```
ones = np.ones((data.shape[0], 1))
X = np.concatenate((ones, data[:,1:2]), axis=1)
y = data[:, 4:]
A = np.matmul(X.T, X) # X^T X
A = np.linalg.inv(A) # (X^T X)^-1
A = np.matmul(A, X.T) # (X^T X)^-1 X^T
beta = np.matmul(A, y) # (X^T X)^-1 X^T y
print(beta)

plt.plot(data[:, 1], data[:, 4], '.', label='TV-sales')
plt.plot(data[:, 1], beta[0]+beta[1]*data[:,1], '.', label='regression')
plt.title('TV sales plot')
plt.legend()
plt.show()
```

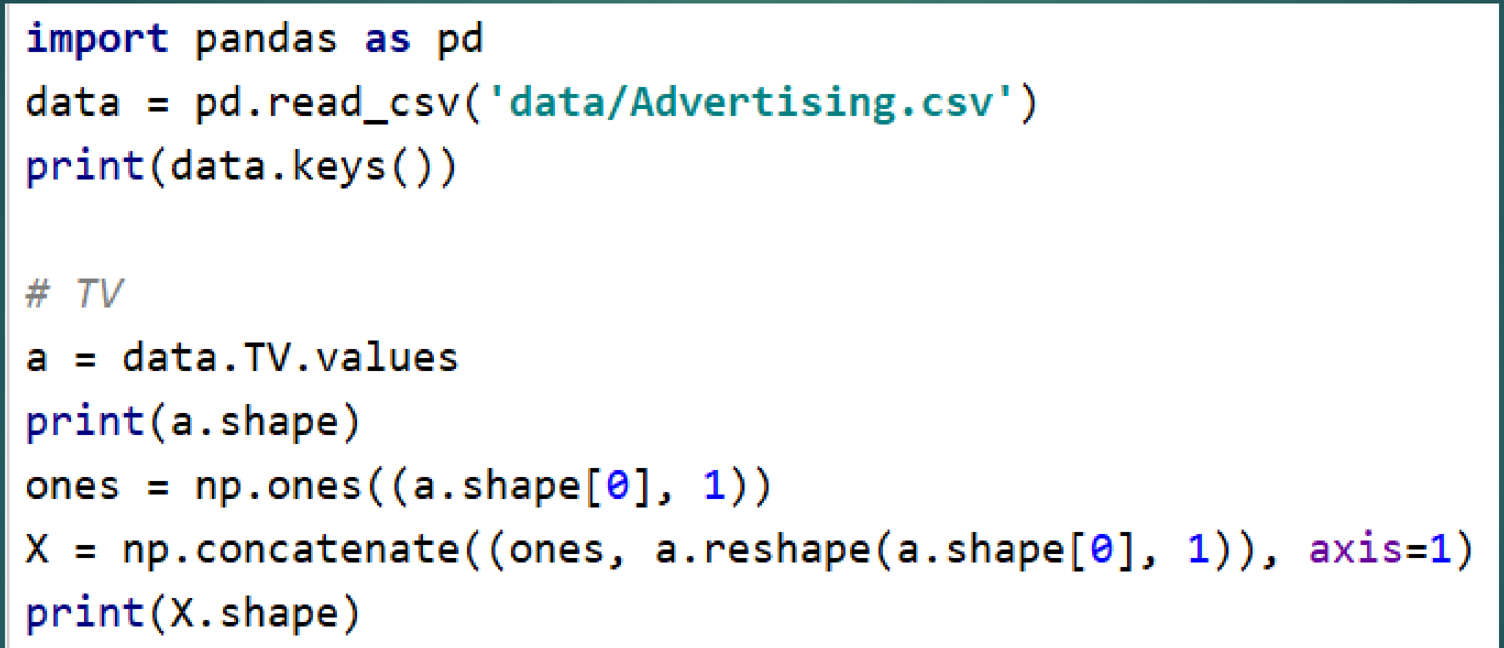


```
[[7.03259355]
 [0.04753664]]
```



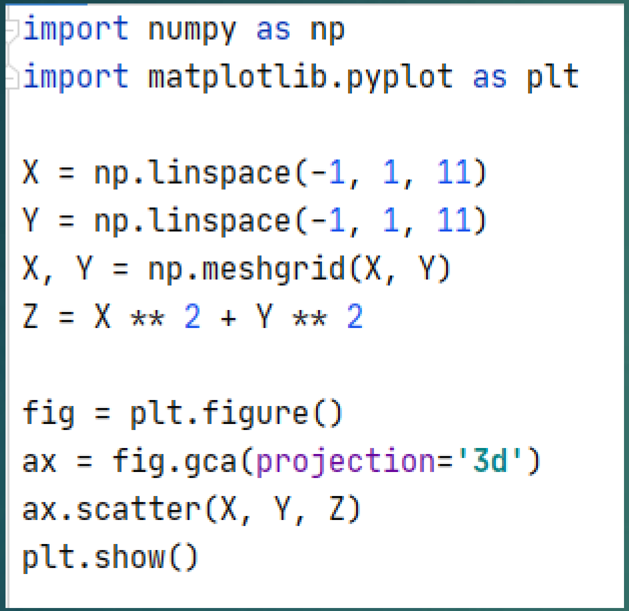
- 연습문제 1. linear regression
  predictors: radio
  response: sales
- 연습문제 2. linear regression
  predictors: TV, radio
  response: sales
- 연습문제 3. polynomial regression and plot
  predictors: $\mathrm{TV}$, $\mathrm{TV}^2$
  response: sales
  plot: regression curve

# Using pandas

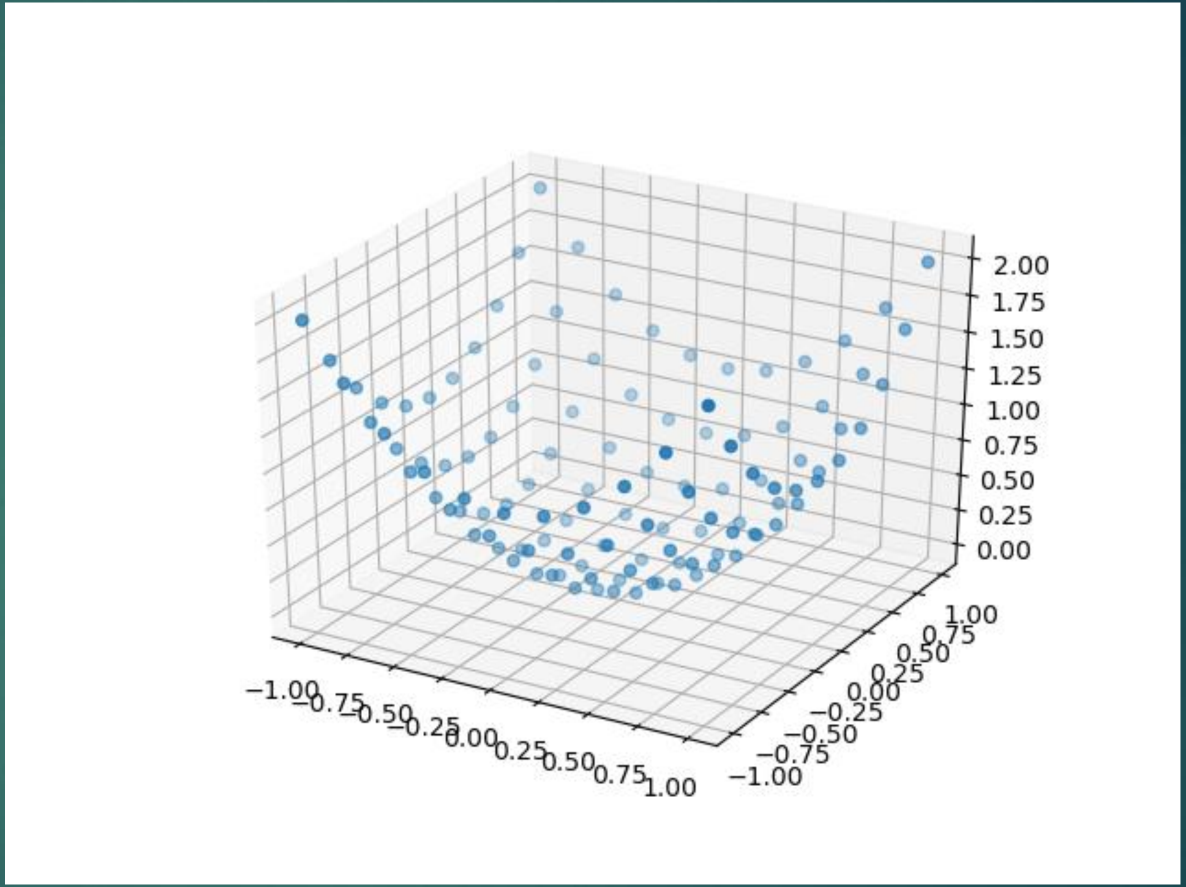


```python
import pandas as pd
data = pd.read_csv('data/Advertising.csv')
print(data.keys())

# TV
a = data.TV.values
print(a.shape)
ones = np.ones((a.shape[0], 1))
X = np.concatenate((ones, a.reshape(a.shape[0], 1)), axis=1)
print(X.shape)
```

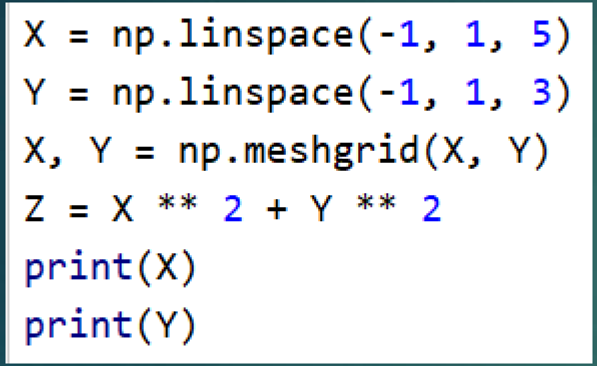
# Scatter



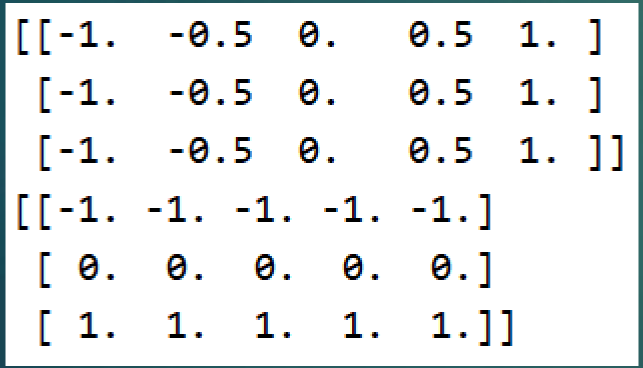
```
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-1, 1, 11)
Y = np.linspace(-1, 1, 11)
X, Y = np.meshgrid(X, Y)
Z = X ** 2 + Y ** 2

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.scatter(X, Y, Z)
plt.show()
```



# numpy.meshgrid



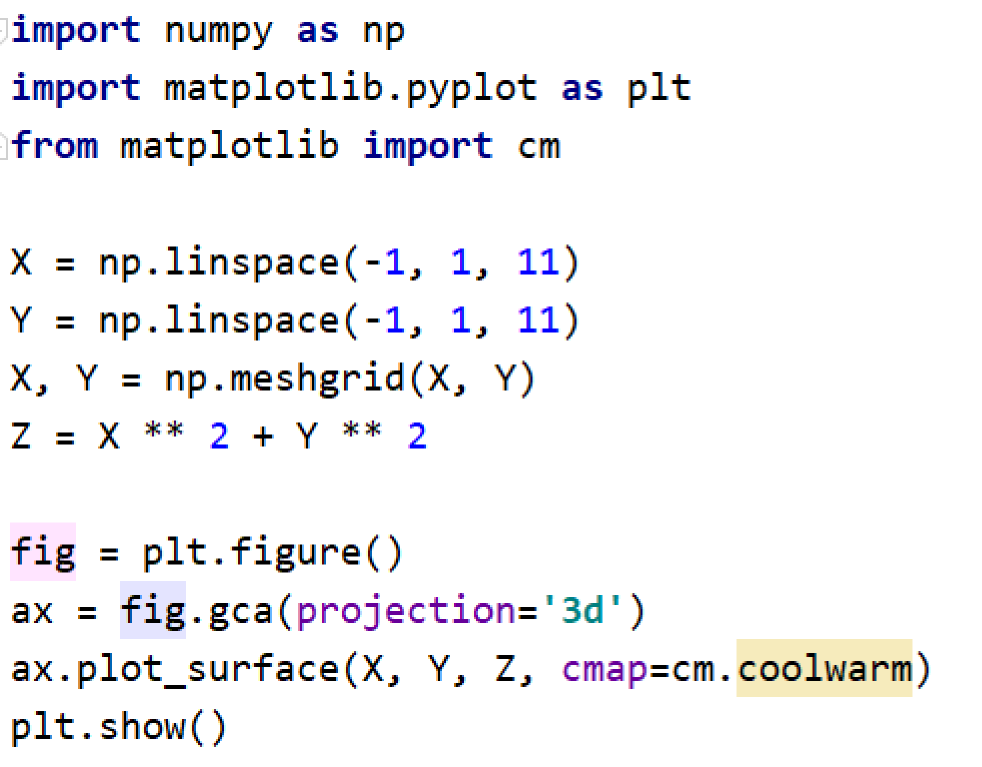
```
X = np.linspace(-1, 1, 5)
Y = np.linspace(-1, 1, 3)
X, Y = np.meshgrid(X, Y)
Z = X ** 2 + Y ** 2
print(X)
print(Y)
```



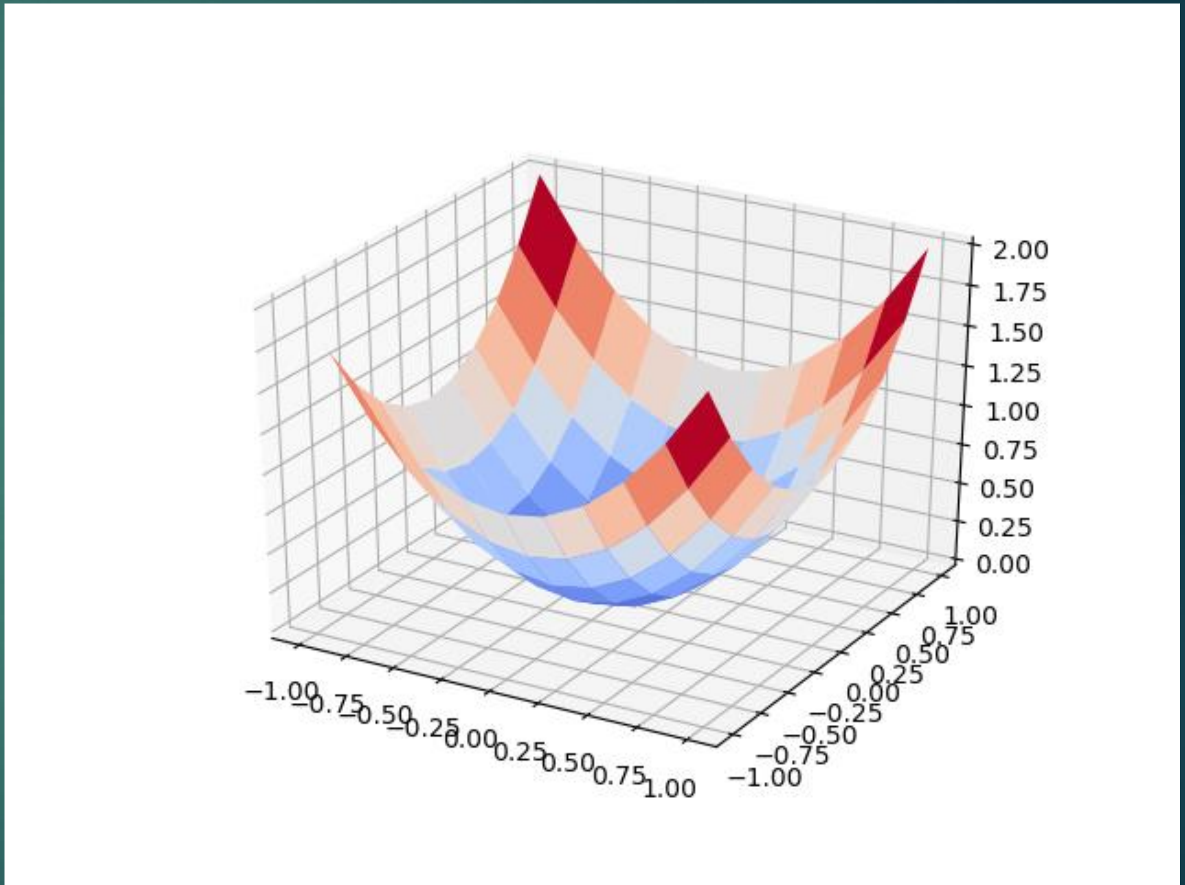
```
[[-1.  -0.5  0.   0.5  1. ]
 [-1.  -0.5  0.   0.5  1. ]
 [-1.  -0.5  0.   0.5  1. ]]
[[-1. -1. -1. -1. -1.]
 [ 0.  0.  0.  0.  0.]
 [ 1.  1.  1.  1.  1.]]
```

| | | | | |
|---|---|---|---|---|
| (-1, -1) | (-0.5, -1) | (0, -1) | (0.5, -1) | (1, -1) |
| (-1, 0) | (-0.5, 0) | (0, 0) | (0.5, 0) | (1, 0) |
| (-1, 1) | (-0.5, 1) | (0, 1) | (0.5, 1) | (1, 1) |

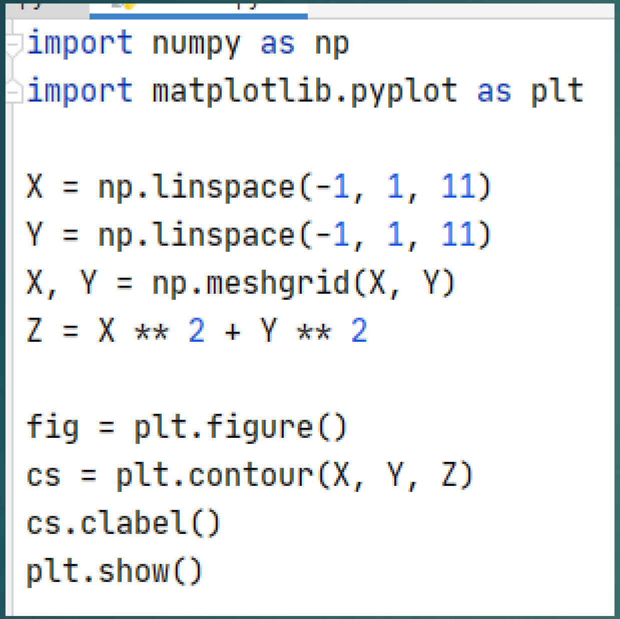
# Plot3D



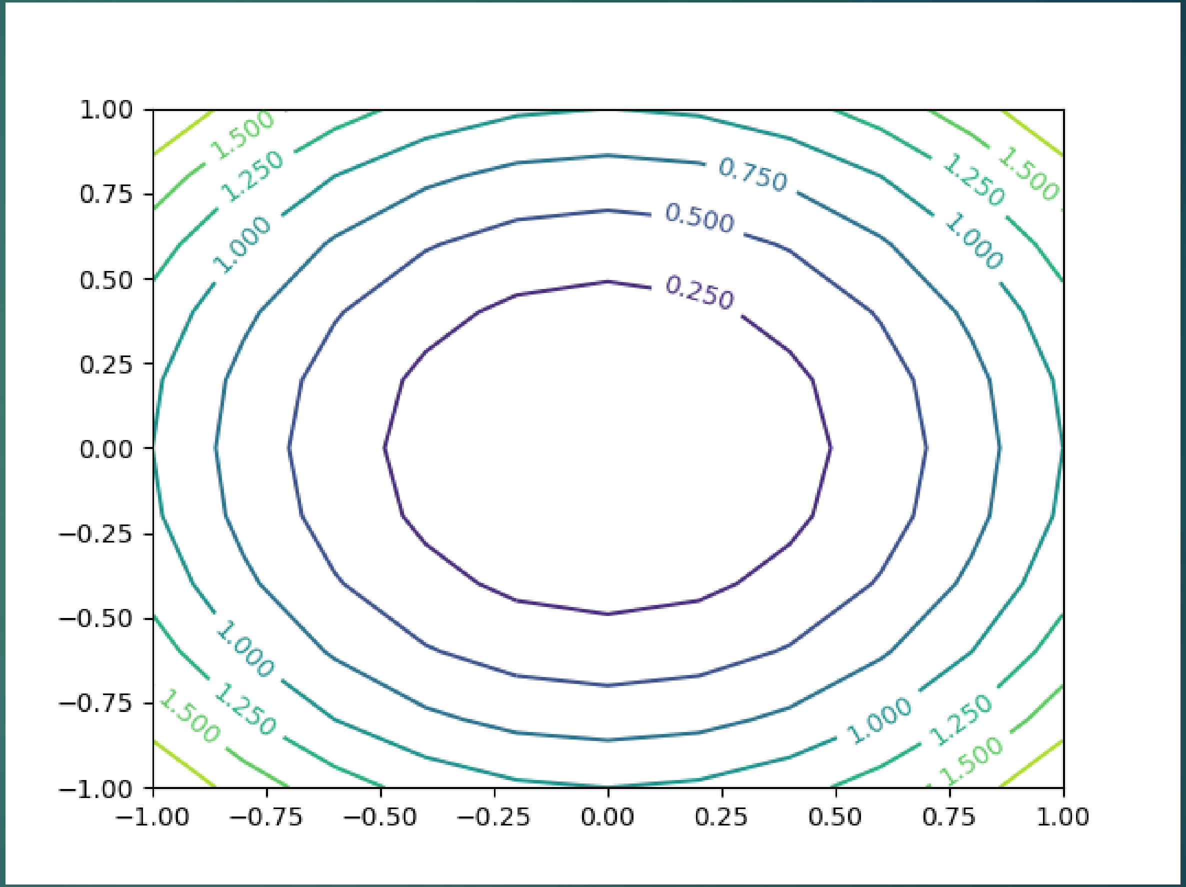
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

X = np.linspace(-1, 1, 11)
Y = np.linspace(-1, 1, 11)
X, Y = np.meshgrid(X, Y)
Z = X ** 2 + Y ** 2

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_surface(X, Y, Z, cmap=cm.coolwarm)
plt.show()
```



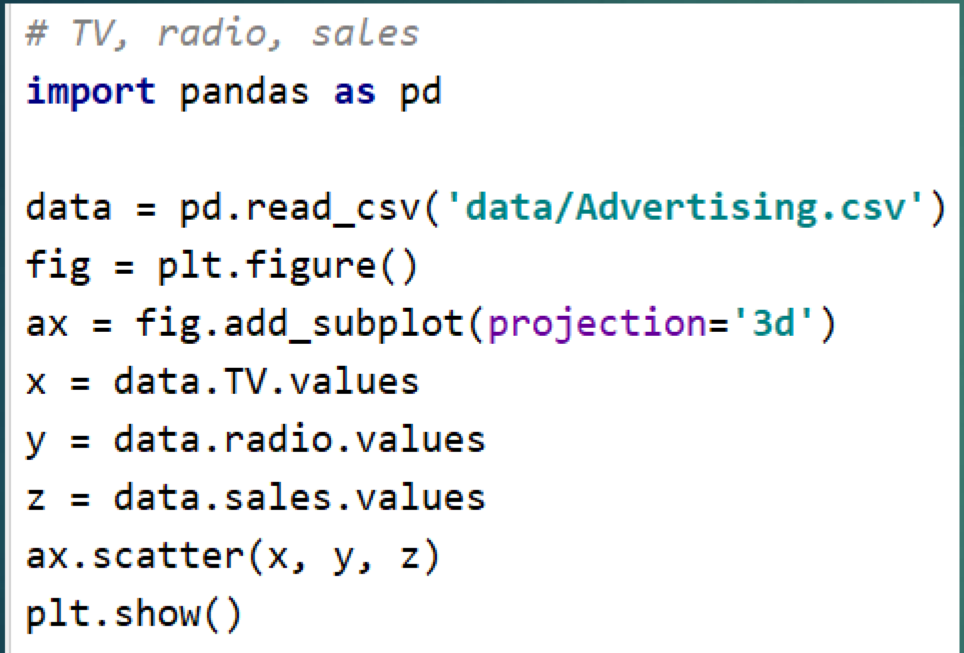
# Contour



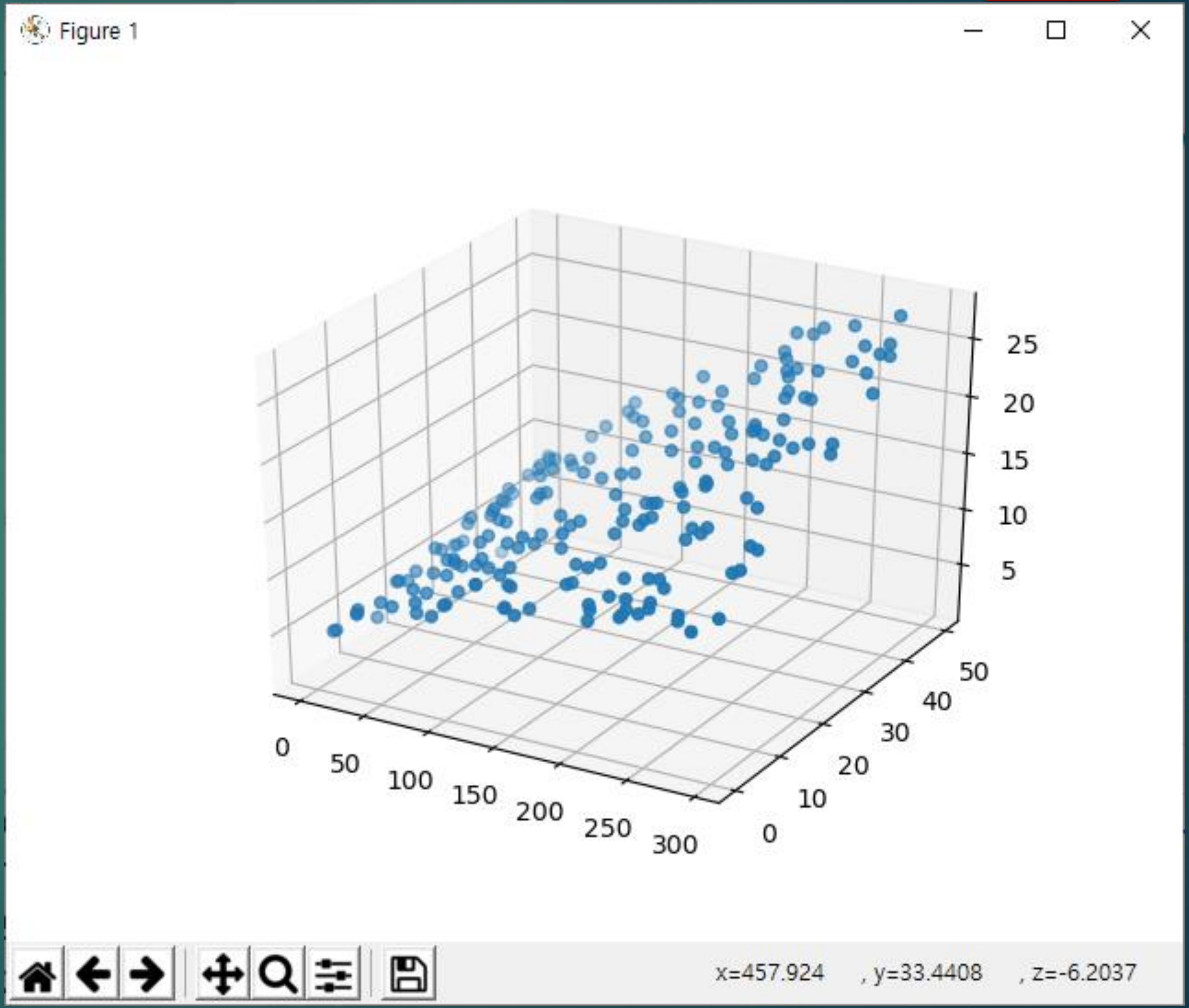
```
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-1, 1, 11)
Y = np.linspace(-1, 1, 11)
X, Y = np.meshgrid(X, Y)
Z = X ** 2 + Y ** 2

fig = plt.figure()
cs = plt.contour(X, Y, Z)
cs.clabel()
plt.show()
```



# scatter



```python
# TV, radio, sales
import pandas as pd

data = pd.read_csv('data/Advertising.csv')
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
x = data.TV.values
y = data.radio.values
z = data.sales.values
ax.scatter(x, y, z)
plt.show()
```



x=457.924 , y=33.4408 , z=-6.2037

- 연습문제 4. plot $z = xy$

- 연습문제 5. linear regression and 3D plot
  predictors: TV, radio
  response: sales
  plot – surface(plane) and scatter