Assignment_#3

20171620 문성찬

- **< 2.9 >**

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

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

Out[2]: array([1, 2, 3])

In [3]: print(x)

[1 2 3]


In [4]: y = np.array([4,5,6])
        print(x+y)

[5 7 9]


In [5]: type(x)

Out[5]: numpy.ndarray

In [6]: x[0]

Out[6]: 1

In [7]: x[0] = 100
        print(x)

[100   2   3]


In [8]: print(np.arange(10))

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


In [9]: print(np.arange(5,10))
```

```
[5 6 7 8 9]
```

```
In [10]: a = np.array([1,1])
         b = a
         print('a=' + str(a))
         print('b=' + str(b))
         b[0] = 100
         print('a=' + str(a))
         print('b=' + str(b))
```

```
a=[1 1]
b=[1 1]
a=[100   1]
b=[100   1]
```

```
In [11]: a = np.array([1,1])
         b = a.copy()
         print('a=' + str(a))
         print('b=' + str(b))
         b[0] = 100
         print('a=' + str(b))
         print('b=' + str(a))
```

```
a=[1 1]
b=[1 1]
a=[100   1]
b=[1 1]
```

- **< 2.10 >**

```
In [12]: x = np.array([[1,2,3],[4,5,6]])
         print(x)
```

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

```
In [13]: x = np.array([[1,2,3],[4,5,6]])
         x.shape
```

```
Out[13]: (2, 3)
```

```
In [14]: w,h = x.shape
         print(w)
         print(h)
```

```
2
3
```

```
In [15]: x = np.array([[1,2,3],[4,5,6]])
         x[1,2]

Out[15]: 6

In [16]: x = np.array([[1,2,3],[4,5,6]])
         x[1,2] = 100
         print(x)

[[  1   2   3]
 [  4   5 100]]


In [17]: print(np.zeros(10))

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


In [18]: print(np.zeros((2,10)))

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


In [19]: print(np.ones((2,10)))

[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]


In [20]: np.random.rand(2,3)

Out[20]: array([[0.80508597, 0.64996135, 0.78518388],
                [0.85905462, 0.47493245, 0.7369535 ]])

In [21]: a = np.arange(10)
         print(a)

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


In [22]: a.reshape(2,5)

Out[22]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])
```

- **< 2.11 >**

```
In [23]: x = np.array([[4,4,4],[8,8,8]])
         y = np.array([[1,1,1],[2,2,2]])
         print(x+y)

[[ 5  5  5]
 [10 10 10]]


In [24]: x = np.array([[4,4,4],[8,8,8]])
         print(10*x)

[[40 40 40]
 [80 80 80]]


In [25]: x = np.array([[4,4,4],[8,8,8]])
         print(np.exp(x))

[[  54.59815003   54.59815003   54.59815003]
 [2980.95798704 2980.95798704 2980.95798704]]


In [26]: v = np.array([[1,2,3],[4,5,6]])
         w = np.array([[1,1],[2,2],[3,3]])
         print(v.dot(w))

[[14 14]
 [32 32]]
```

- **< 2.12 >**

```
In [27]: x = np.arange(10)
         print(x)
         print(x[:5])

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


In [28]: print(x[5:])

[5 6 7 8 9]


In [29]: print(x[3:8])

[3 4 5 6 7]
```

```
In [30]: print(x[3:8:2])

[3 5 7]


In [31]: print(x[::-1])

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


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

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

- **< 2.13 >**

```
In [33]: x = np.array([1,1,2,3,5,8,13])
         x>3

Out[33]: array([False, False, False, False,  True,  True,  True])

In [34]: x[x>3]

Out[34]: array([ 5,  8, 13])

In [35]: x[x>3] = 999
         print(x)

[  1   1   2   3 999 999 999]
```

- **< 2.14 >**

```
In [36]: help(np.random.randint)

Help on built-in function randint:

randint(...) method of numpy.random.mtrand.RandomState instance
    randint(low, high=None, size=None, dtype=int)

    Return random integers from `low` (inclusive) to `high` (exclusive).

    Return random integers from the "discrete uniform" distribution of
```

```
the specified dtype in the "half-open" interval [`low`, `high`). If
`high` is None (the default), then results are from [0, `low`).

.. note::
    New code should use the ``integers`` method of a ``default_rng()``
    instance instead; see `random-quick-start`.

Parameters
----------
low : int or array-like of ints
    Lowest (signed) integers to be drawn from the distribution (unless
    ``high=None``, in which case this parameter is one above the
    *highest* such integer).
high : int or array-like of ints, optional
    If provided, one above the largest (signed) integer to be drawn
    from the distribution (see above for behavior if ``high=None``).
    If array-like, must contain integer values
size : int or tuple of ints, optional
    Output shape.  If the given shape is, e.g., ``(m, n, k)``, then
    ``m * n * k`` samples are drawn.  Default is None, in which case a
    single value is returned.
dtype : dtype, optional
    Desired dtype of the result. Byteorder must be native.
    The default value is int.

    .. versionadded:: 1.11.0

Returns
-------
out : int or ndarray of ints
    `size`-shaped array of random integers from the appropriate
    distribution, or a single such random int if `size` not provided.

See Also
--------
random_integers : similar to `randint`, only for the closed
    interval [`low`, `high`], and 1 is the lowest value if `high` is
    omitted.
Generator.integers: which should be used for new code.

Examples
--------
>>> np.random.randint(2, size=10)
array([1, 0, 0, 0, 1, 1, 0, 0, 1, 0]) # random
>>> np.random.randint(1, size=10)
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

Generate a 2 x 4 array of ints between 0 and 4, inclusive:
```

```
>>> np.random.randint(5, size=(2, 4))
array([[4, 0, 2, 1], # random
       [3, 2, 2, 0]])

Generate a 1 x 3 array with 3 different upper bounds

>>> np.random.randint(1, [3, 5, 10])
array([2, 2, 9]) # random

Generate a 1 by 3 array with 3 different lower bounds

>>> np.random.randint([1, 5, 7], 10)
array([9, 8, 7]) # random

Generate a 2 by 4 array using broadcasting with dtype of uint8

>>> np.random.randint([1, 3, 5, 7], [[10], [20]], dtype=np.uint8)
array([[ 8,  6,  9,  7], # random
       [ 1, 16,  9, 12]], dtype=uint8)
```

- **< 2.15 >**

```
In [37]: def my_func1():
             print('Hi!')
         my_func1()
```

```
Hi!
```

```
In [38]: def my_func2(a,b):
             c = a+b
             return c
         my_func2(1,2)
```

```
Out[38]: 3
```

```
In [39]: def my_func3(D):
             m = np.mean(D)
             s = np.std(D)
             return m,s
```

```
In [40]: data = np.random.randn(100)
         data_mean, data_std = my_func3(data)
         print('mean:{0:3.2f}, std:{1:3.2f}'.format(data_mean,data_std))
```

```
mean:0.08, std:0.96
```

```
In [41]: output = my_func3(data)
         print(output)
         print(type(output))
         print('mean:{0:3.2f}, std:{1:3.2f}'.format(output[0],output[1]))
```

```
(0.08400073520820288, 0.9625787495367815)
<class 'tuple'>
mean:0.08, std:0.96
```

- **< 2.16 >**

```
In [42]: data = np.random.randn(5)
         print(data)
         np.save('datafile.npy',data)
         data = []
         print(data)
         data = np.load('datafile.npy')
         print(data)
```

```
[ 0.32039821 -0.07817565  1.00649085  0.11657722  0.36309864]
[]
[ 0.32039821 -0.07817565  1.00649085  0.11657722  0.36309864]
```

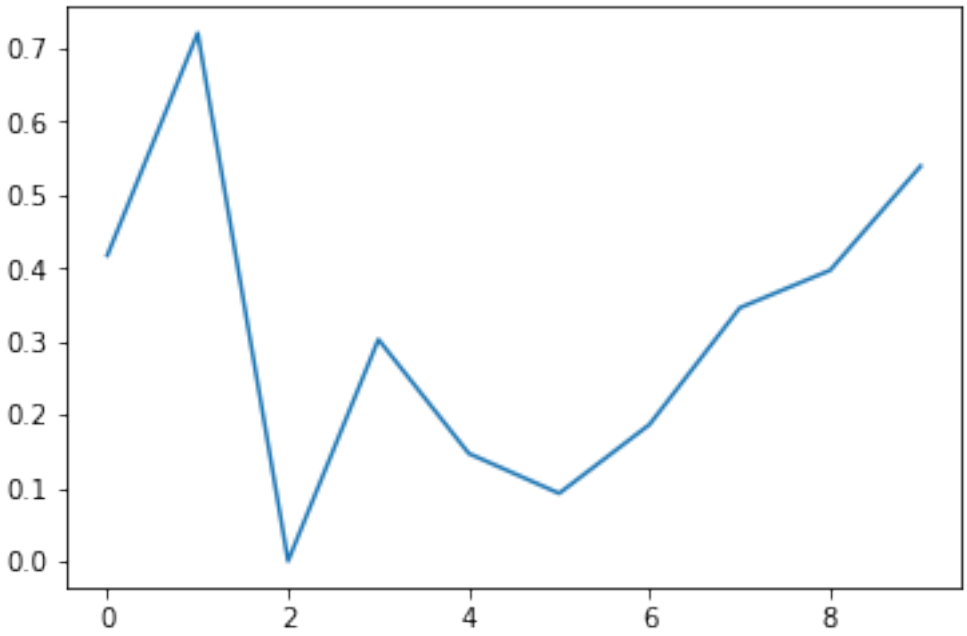
```
In [43]: data1 = np.array([1,2,3])
         data2 = np.array([10,20,30])
         np.savez('datafile2.npz',data1 = data1,data2 = data2)
         data1 = []
         data2 = []
         outfile = np.load('datafile2.npz')
         print(outfile.files)
         data1 = outfile['data1']
         data2 = outfile['data2']
         print(data1)
         print(data2)
```

```
['data1', 'data2']
[1 2 3]
[10 20 30]
```

- ** < 3.1 >**

```
In [44]: import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline

         np.random.seed(1)
```

```
x = np.arange(10)
y = np.random.rand(10)

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




```
In [45]: %reset

Once deleted, variables cannot be recovered. Proceed (y/[n])? y


In [46]: import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline

         def f(x):
             return (x-2) * x * (x+2)

In [47]: print(f(1))

-3


In [48]: print(f(np.array([1,2,3])))

[-3  0 15]
```

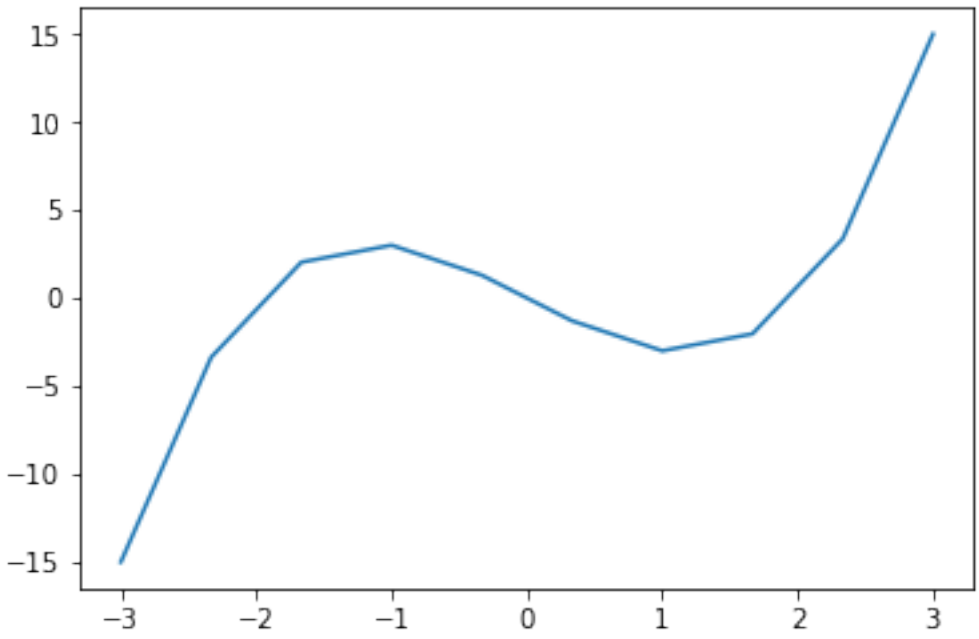
```
In [49]: x = np.arange(-3,3.5,0.5)
         print(x)
```

```
[-3.  -2.5 -2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2.   2.5  3. ]
```

```
In [50]: x = np.linspace(-3,3,10)
         print(np.round(x,2))
```

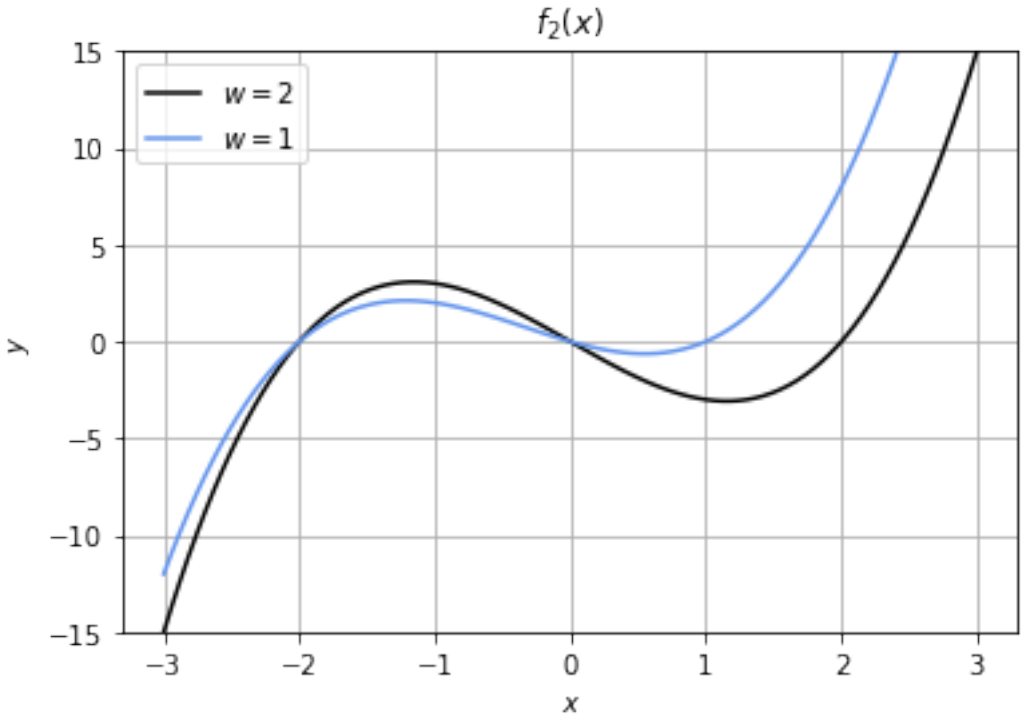
```
[-3.   -2.33 -1.67 -1.   -0.33  0.33  1.    1.67  2.33  3.  ]
```

```
In [51]: plt.plot(x,f(x))
         plt.show()
```




```
In [52]: def f2(x,w):
             return (x-w)*x*(x+2)

         x = np.linspace(-3,3,100)

         plt.plot(x,f2(x,2),color= 'black', label = '$w=2$')
         plt.plot(x,f2(x,1),color = 'cornflowerblue',label = '$w=1$')
         plt.legend(loc="upper left")
         plt.ylim(-15,15)
         plt.title('$f_2(x)$')
         plt.xlabel('$x$')
```

```
plt.ylabel('$y$')
plt.grid(True)
plt.show()
```




```
In [53]: import matplotlib
         matplotlib.colors.cnames
```

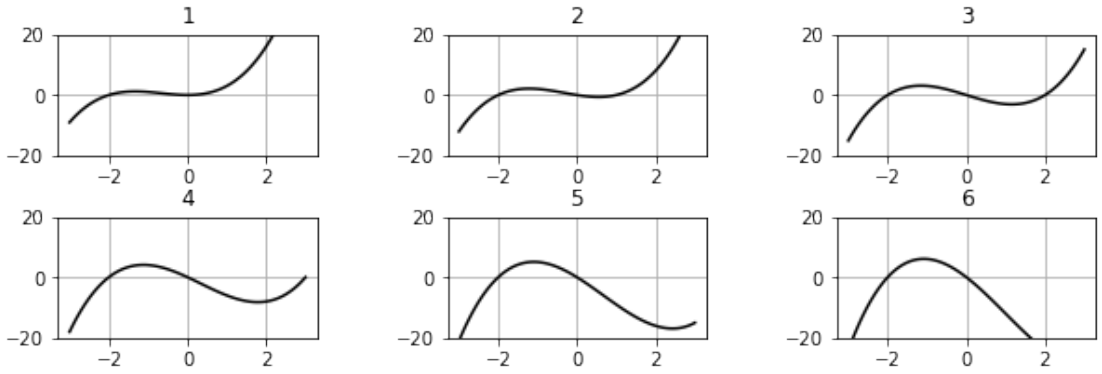
```
Out[53]: {'aliceblue': '#F0F8FF',
          'antiquewhite': '#FAEBD7',
          'aqua': '#00FFFF',
          'aquamarine': '#7FFFD4',
          'azure': '#F0FFFF',
          'beige': '#F5F5DC',
          'bisque': '#FFE4C4',
          'black': '#000000',
          'blanchedalmond': '#FFEBCD',
          'blue': '#0000FF',
          'blueviolet': '#8A2BE2',
          'brown': '#A52A2A',
          'burlywood': '#DEB887',
          'cadetblue': '#5F9EA0',
          'chartreuse': '#7FFF00',
          'chocolate': '#D2691E',
          'coral': '#FF7F50',
```

```
'cornflowerblue': '#6495ED',
'cornsilk': '#FFF8DC',
'crimson': '#DC143C',
'cyan': '#00FFFF',
'darkblue': '#00008B',
'darkcyan': '#008B8B',
'darkgoldenrod': '#B8860B',
'darkgray': '#A9A9A9',
'darkgreen': '#006400',
'darkgrey': '#A9A9A9',
'darkkhaki': '#BDB76B',
'darkmagenta': '#8B008B',
'darkolivegreen': '#556B2F',
'darkorange': '#FF8C00',
'darkorchid': '#9932CC',
'darkred': '#8B0000',
'darksalmon': '#E9967A',
'darkseagreen': '#8FBC8F',
'darkslateblue': '#483D8B',
'darkslategray': '#2F4F4F',
'darkslategrey': '#2F4F4F',
'darkturquoise': '#00CED1',
'darkviolet': '#9400D3',
'deeppink': '#FF1493',
'deepskyblue': '#00BFFF',
'dimgray': '#696969',
'dimgrey': '#696969',
'dodgerblue': '#1E90FF',
'firebrick': '#B22222',
'floralwhite': '#FFFAF0',
'forestgreen': '#228B22',
'fuchsia': '#FF00FF',
'gainsboro': '#DCDCDC',
'ghostwhite': '#F8F8FF',
'gold': '#FFD700',
'goldenrod': '#DAA520',
'gray': '#808080',
'green': '#008000',
'greenyellow': '#ADFF2F',
'grey': '#808080',
'honeydew': '#F0FFF0',
'hotpink': '#FF69B4',
'indianred': '#CD5C5C',
'indigo': '#4B0082',
'ivory': '#FFFFF0',
'khaki': '#F0E68C',
'lavender': '#E6E6FA',
'lavenderblush': '#FFF0F5',
```

```
'lawngreen': '#7CFC00',
'lemonchiffon': '#FFFACD',
'lightblue': '#ADD8E6',
'lightcoral': '#F08080',
'lightcyan': '#E0FFFF',
'lightgoldenrodyellow': '#FAFAD2',
'lightgray': '#D3D3D3',
'lightgreen': '#90EE90',
'lightgrey': '#D3D3D3',
'lightpink': '#FFB6C1',
'lightsalmon': '#FFA07A',
'lightseagreen': '#20B2AA',
'lightskyblue': '#87CEFA',
'lightslategray': '#778899',
'lightslategrey': '#778899',
'lightsteelblue': '#B0C4DE',
'lightyellow': '#FFFFE0',
'lime': '#00FF00',
'limegreen': '#32CD32',
'linen': '#FAF0E6',
'magenta': '#FF00FF',
'maroon': '#800000',
'mediumaquamarine': '#66CDAA',
'mediumblue': '#0000CD',
'mediumorchid': '#BA55D3',
'mediumpurple': '#9370DB',
'mediumseagreen': '#3CB371',
'mediumslateblue': '#7B68EE',
'mediumspringgreen': '#00FA9A',
'mediumturquoise': '#48D1CC',
'mediumvioletred': '#C71585',
'midnightblue': '#191970',
'mintcream': '#F5FFFA',
'mistyrose': '#FFE4E1',
'moccasin': '#FFE4B5',
'navajowhite': '#FFDEAD',
'navy': '#000080',
'oldlace': '#FDF5E6',
'olive': '#808000',
'olivedrab': '#6B8E23',
'orange': '#FFA500',
'orangered': '#FF4500',
'orchid': '#DA70D6',
'palegoldenrod': '#EEE8AA',
'palegreen': '#98FB98',
'paleturquoise': '#AFEEEE',
'palevioletred': '#DB7093',
'papayawhip': '#FFEFD5',
```

```
 'peachpuff': '#FFDAB9',
 'peru': '#CD853F',
 'pink': '#FFC0CB',
 'plum': '#DDA0DD',
 'powderblue': '#B0E0E6',
 'purple': '#800080',
 'rebeccapurple': '#663399',
 'red': '#FF0000',
 'rosybrown': '#BC8F8F',
 'royalblue': '#4169E1',
 'saddlebrown': '#8B4513',
 'salmon': '#FA8072',
 'sandybrown': '#F4A460',
 'seagreen': '#2E8B57',
 'seashell': '#FFF5EE',
 'sienna': '#A0522D',
 'silver': '#C0C0C0',
 'skyblue': '#87CEEB',
 'slateblue': '#6A5ACD',
 'slategray': '#708090',
 'slategrey': '#708090',
 'snow': '#FFFAFA',
 'springgreen': '#00FF7F',
 'steelblue': '#4682B4',
 'tan': '#D2B48C',
 'teal': '#008080',
 'thistle': '#D8BFD8',
 'tomato': '#FF6347',
 'turquoise': '#40E0D0',
 'violet': '#EE82EE',
 'wheat': '#F5DEB3',
 'white': '#FFFFFF',
 'whitesmoke': '#F5F5F5',
 'yellow': '#FFFF00',
 'yellowgreen': '#9ACD32'}
```

In [54]:
```
plt.figure(figsize=(10,3))
plt.subplots_adjust(wspace = 0.5,hspace = 0.5)
for i in range(6):
    plt.subplot(2,3,i+1)
    plt.title(i+1)
    plt.plot(x,f2(x,i),'k')
    plt.ylim(-20,20)
    plt.grid(True)
plt.show()
```



- **< 3.2 >**

```
In [55]: import numpy as np
         import matplotlib.pyplot as plt

         def f3(x0,x1):
             r = 2*x0**2 + x1**2
             ans = r * np.exp(-r)
             return ans

         xn = 9
         x0 = np.linspace(-2,2,xn)
         x1 = np.linspace(-2,2,xn)
         y = np.zeros((len(x0),len(x1)))
         for i0 in range(xn):
             for i1 in range(xn):
                 y[i1,i0] = f3(x0[i0],x1[i1])
```

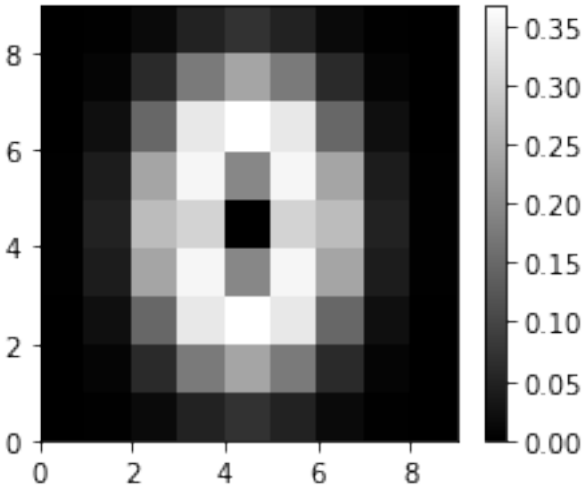
```
In [56]: print(x0)
```

```
[-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
```

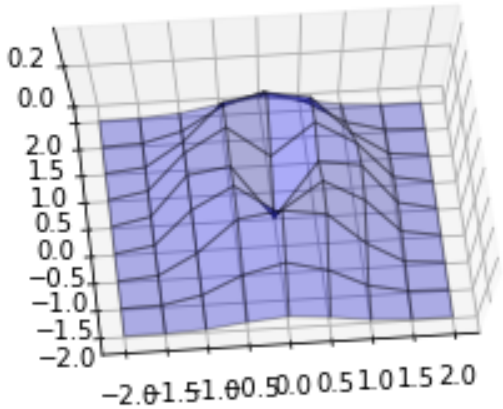
```
In [57]: print(np.round(y,1))
```

```
[[0.  0.  0.  0.  0.1 0.  0.  0.  0. ]
 [0.  0.  0.1 0.2 0.2 0.2 0.1 0.  0. ]
 [0.  0.  0.1 0.3 0.4 0.3 0.1 0.  0. ]
 [0.  0.  0.2 0.4 0.2 0.4 0.2 0.  0. ]
 [0.  0.  0.3 0.3 0.  0.3 0.3 0.  0. ]
 [0.  0.  0.2 0.4 0.2 0.4 0.2 0.  0. ]
 [0.  0.  0.1 0.3 0.4 0.3 0.1 0.  0. ]
 [0.  0.  0.1 0.2 0.2 0.2 0.1 0.  0. ]
 [0.  0.  0.  0.  0.1 0.  0.  0.  0. ]]
```

```
In [58]: plt.figure(figsize = (3.5,3))
         plt.gray()
         plt.pcolor(y)
         plt.colorbar()
         plt.show()
```




```
In [59]: from mpl_toolkits.mplot3d import Axes3D
  xx0, xx1 = np.meshgrid(x0,x1)

  plt.figure(figsize=(5,3.5))
  ax = plt.subplot(1,1,1,projection= '3d')
  ax.plot_surface(xx0,xx1,y,rstride=1,cstride=1,alpha=0.3,color='blue',edgecolor = 'black')
  ax.set_zticks((0,0.2))
  ax.view_init(75,-95)
  plt.show()
```



```
In [60]: print(x0)
         print(x1)
```

```
[-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
[-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
```

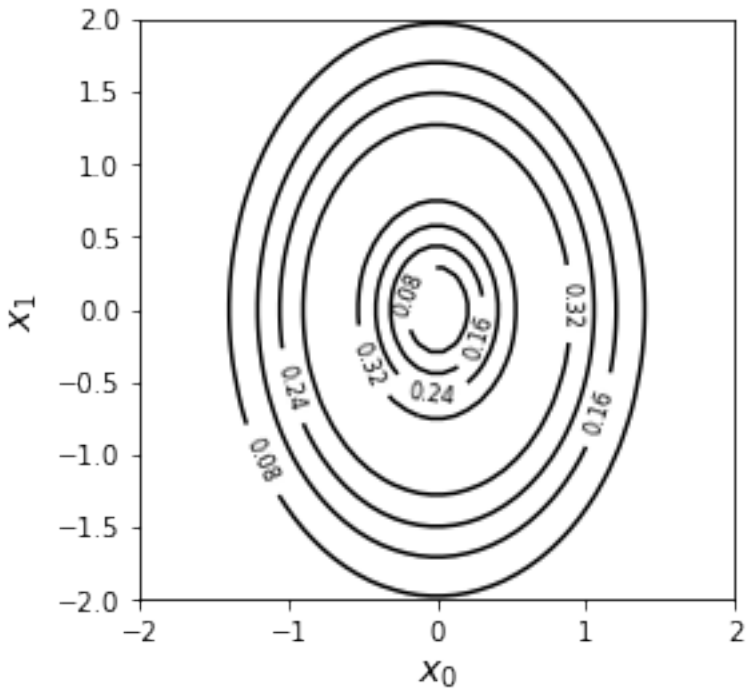
```
In [61]: print(xx0)
```

```
[[-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
 [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]]
```

```
In [62]: print(xx1)
```

```
[[-2.  -2.  -2.  -2.  -2.  -2.  -2.  -2.  -2. ]
 [-1.5 -1.5 -1.5 -1.5 -1.5 -1.5 -1.5 -1.5 -1.5]
 [-1.  -1.  -1.  -1.  -1.  -1.  -1.  -1.  -1. ]
 [-0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5]
 [ 0.   0.   0.   0.   0.   0.   0.   0.   0. ]
 [ 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5]
 [ 1.   1.   1.   1.   1.   1.   1.   1.   1. ]

 [ 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5]
 [ 2.   2.   2.   2.   2.   2.   2.   2.   2. ]]
```

In [63]:
```
xn = 50
x0 = np.linspace(-2,2,xn)
x1 = np.linspace(-2,2,xn)

y = np.zeros((len(x0),len(x1)))
for i0 in range(xn):
    for i1 in range(xn):
        y[i1,i0] = f3(x0[i0],x1[i1])

xx0,xx1  = np.meshgrid(x0,x1)

plt.figure(1,figsize = (4,4))
cont = plt.contour(xx0,xx1,y,5,colors='black')
cont.clabel(fmt='%3.2f',fontsize=8)
plt.xlabel('$x_0$', fontsize = 14)
plt.ylabel('$x_1$', fontsize = 14)
plt.show()
```



소감 :

이번 수치해석 과제를 통해 파이썬에서의 벡터,행렬,함수 등 앞으로 수치해석 과목에서 사용할지도 모르는 요소들에 대한 정의와 연산 등을 실습해보았는데 이전에 알았던 개념들도 다시 복습해보는 기회가 되었던 것 같아 좋았고,
bool배열처럼 이전에 몰랐던 정보도 얻을 수 있어서 유익한 시간이었습니다.
또한 이전 수치해석 과제로 해본 그래프 그리기를 다시 한번 실습해볼 수 있었는데 이전 과제에서 다루었던 부분들을 복습하는 개념으로 실습할 수 있었고 3차원 그래프 등 새로운 그래프들도 한번 구현해봄으로써 그래프 그리기에 대해서 더 많은 지식을 얻는 뜻깊은 시간이었습니다.