수치해석

소프트웨어학부 20162820 김영민

Homework #3 과제

(In and Out Practice)

P57 ~ P60

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

In [8]: print(x)
        [1 2 3]

In [9]: y=np.array([4,5,6])
        print(x+y)
        [5 7 9]

In [10]: type(x)
Out[10]: numpy.ndarray

In [11]: x[0]
Out[11]: 1

In [13]: x[0]=100
         print(x)
         [100   2   3]

In [14]: print(np.arange(10))
         [0 1 2 3 4 5 6 7 8 9]

In [15]: print(np.arange(5,10))
         [5 6 7 8 9]
```

소감 : 넘파이라는 라이브러리를 이용하여 기능확장을 실습하였다. Np.array이런식의 형태를 이용해 배열을 선언할 수 있고 파이썬의 특성을 이용하여 배열의 arrange를이용하여 구간설정도 가능함을 알았다.

P60 ~ P63

```
In [16]: a = np.array([1,1])
         b=a
         print('a =' + str(a))
         print('b =' + str(b))
         b[0] = 100
         print('b =' + str(b))
         print('a =' + str(a))

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

```
In [18]: a = np.array([1,1])
         b = a.copy()
         print('a =' + str(a))
         print('b =' + str(b))
         b[0] = 100
         print('b =' + str(b))
         print('a =' + str(a))

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

```
In [19]: x = np.array([[1,2,3],[4,5,6]])
         print(x)

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

```
In [20]: x=np.array([[1,2,3],[4,5,6]])
         x.shape
Out[20]: (2, 3)
```

```
In [21]: w, h = x.shape
         print(w)
         print(h)

         2
         3
```

소감 : ndarray형을 사용할 때 복사하기위해서는 즉 변수 a와 b를 같게 하려면 b = a형식이 아니라 b=a.copy()를하여 복사를 해야한다. 아니면 값변경시 똑같이 변경되기 때문이다. 그리고 행렬의 크기는 ndarray변수명.shape로 알 수 있다.

P63 ~ P65

```
In [23]: x = np.array([[1,2,3], [4,5,6]])
         x[1,2]

Out[23]: 6

In [24]: x = np.array([[1,2,3],[4,5,6]])
         x[1,2] = 100
         print(x)

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

In [26]: print(np.zeros(10))

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

In [29]: 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 [30]: 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 [31]: np.random.rand(2,3)

Out[31]: array([[0.63675456, 0.10422119, 0.09744892],
                [0.05760444, 0.8283759 , 0.72513715]])

In [33]: a = np.arange(10)
         print(a)

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

In [35]: a.reshape(2,5)

Out[35]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])
```

소감 : np.zeros(size), np.ones(size)로 배열의 값을 0 그리고 1로 채워넣을 수 있다. 그리고 random이라는 값을 사용하면 범위내에있는 값들 중 랜덤으로 배열에 들어가게된다.

P65 ~ P68

```
In [36]: 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 [45]: x = np.array([[4,4,4],[8,8,8]])
         print(10+x)
         print(np.exp(x))
```

```
[[40 40 40]
 [80 80 80]]
[[  54.59815003   54.59815003   54.59815003]
 [2980.95798704 2980.95798704 2980.95798704]]
```

```
In [40]: 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]]
```

```
In [42]: 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 [43]: print(x[5:])
         print(x[3:8])
         print(x[3:8:2])
         print(x[::-1])
```

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

```
In [44]: 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]]
```

소감 : 행렬의 사칙연산은 일반 사칙연산과 같이 가능하고 배열에 저장돼있는값끼리 사칙연산을 수행한다. 파이썬의 인덱스 슬라이싱을통해 배열범위중 원하는 범위만 출력가능하고 순서를 바꿔줄 수 있는걸 알았다.

P69 ~ P73

```
In [49]: x = np.array([1,1,2,3,6,8,13])
         print(x > 3)
         print(x[x>3])
         x[x>3]=999
         print(x)

         [False False False False  True  True  True]
         [ 6  8 13]
         [  1   1   2   3 999 999 999]
```

```
In [51]: def my_func1():
             print('hi')
         my_func1()

         hi
```

```
In [52]: def my_func2(a,b):
             c=a+b
             return c

         my_func2(1,2)

Out[52]: 3
```

```
In [53]: def my_func3(D):
             m = np.mean(D)
             s = np.std(D)
             return m, s
```

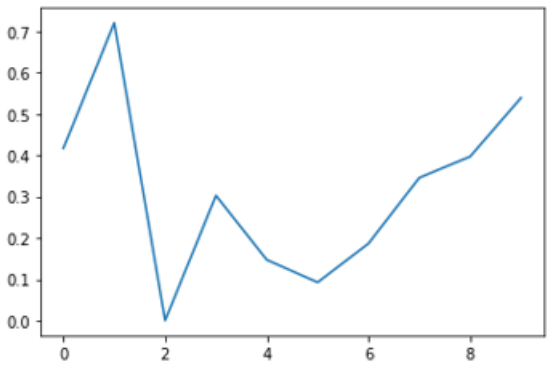
```
In [56]: data = np.random.randn(100)
         data_mean, data_std= my_func3(data)
         print('mean: {0:3.2f}, std:{1:3.2f}'.format(data_mean, data_std))
         output = my_func3(data)
         print(output)

         mean: -0.11, std:1.14
         (-0.11234226865799803, 1.1443370227892886)
```

```
In [57]: data = np.random.randn(5)
         print(data)
         np.save('datafile.npy',data)
         data = []
         print(data)
         data = np.load('datafile.npy')
         print(data)

         [ 0.10714143  0.45328147 -0.52061406 -0.47264563  0.10413133]
         []
         [ 0.10714143  0.45328147 -0.52061406 -0.47264563  0.10413133]
```

소감 : 함수 설정하여 return값에 반환값을 주어 함수 호출시 return에 지정돼있는 값을 출력할 수 있다. 아니면 함수자체에 print()를주어 함수만 호출해도 출력되게 할 수 있다. 파일에 저장하기위해 np.save()를통해 가능하다는걸 알았고, 로드 np.load()도가능한걸 알았다..

P75 ~ P78

```
In [61]: 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 [62]: %reset

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

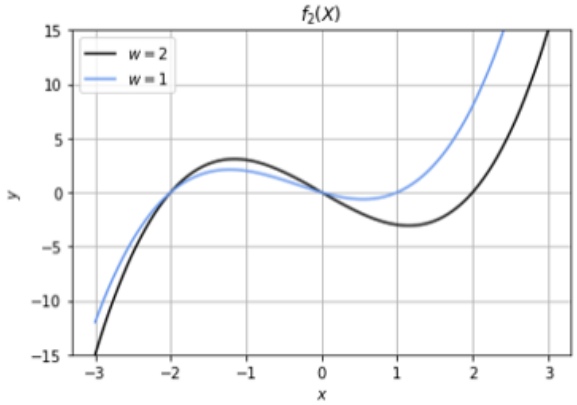
```
In [66]: import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline

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

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

-3
[-3  0 15]
```

```
In [68]: x = np.arange(-3,3.5,0.5)
         print(x)
         x = np.linspace(-3,3,10)
         print(np.round(x,2))

[-3.  -2.5 -2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2.   2.5  3. ]
[-3.   -2.33 -1.67 -1.   -0.33  0.33  1.   1.67  2.33  3.  ]
```

소감 : 본격적인 2차원 그래프 그리기를 실습하였다. Random.seed(1)를 통해 난수고정법을 배웠고,plot(x,y)를주어 꺾은선그래프 그리고 show()를 통해 그래프 출력을 배웠다. 그리고 $reset을통해 그래프를 리셋하였다.

P79 ~ P81

```
In [72]: 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()
```



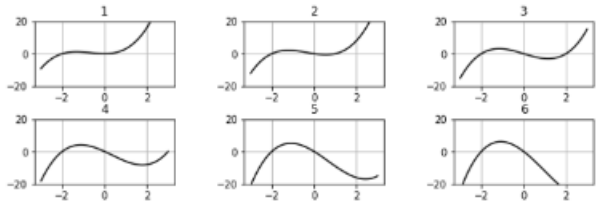

```
In [73]: import matplotlib
         matplotlib.colors.cnames
```

```
Out[73]: {'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'
```

소감 : 그래프 장식하는방법을 실습해 보았다. 이전보다는 조금더 손질된 버전으로 표현하였고,x라벨,y라벨 닉네임을 지정해주고 그래프 title또한 이름을 설정해주었다. 그리고 사용가능한 색상리스트 표현하는 colors.cnames를 통해 정보를얻을 수 있는걸 알았다.

P82 ~ P85

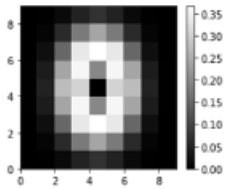
In [11]:
```
plt.figure(figsize=(10, 6))
plt.subplots_adjust(wspace=0.5, hspace=0.5) for i in range(6):
    plt.subplot(2, 3, i + 1)
    plt.title(i + 1)
    plt.plot(x, f2(x, i), 'k')
    plt.ylim(-20, 20)
    plt.grid(True)
plt.show()
```



In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

def f8(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] = f8(x0[i0], x1[i1])

print(x0)
print(np.round(y, 1))
```

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

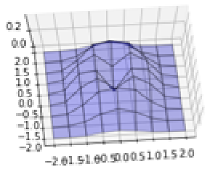
In [16]:
```
plt.figure(figsize=(3.5, 3))
plt.gray()
plt.pcolor(y)
plt.colorbar()
plt.show()
```



소감 : 3 차원그래프를 표현하는 방법을 배웠고 그래프를 여러 개 표현하기 위해서는 figure 을통해 전체 영역의 크기를 지정하여 표현이가능하는 것을 알았다.그리고 수치를 색으로 표현하기위해서는 pcolor 를 이용하는방법도 알았다.

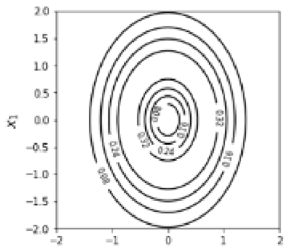
P85 ~ P90

```
In [4]: 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()

        print(x0)
        print(" ")
        print(x1)
        print(" ")
        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. ]
 [-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 [20]: 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 : 함수의 표면을 표시하는 surface 를통해 3 차원 입체그래프로 표시하는법을 배웠다, 그리고 좌표점의 리스트를 출력하여 값을 파악하였다. 그리고 contour 를통해 함수의 높이를알아보는 등고선플롯을 배웠다.