

# Lab08

---

소프트웨어의 이해



# 1. Matplotlib

## 시각화 통합 라이브러리 패키지

파이썬을 이용하여 주어진 데이터를 정적, 동적, 대화식 그래프 형식으로 시각화.

설치

```
C:\W>pip install matplotlib
```

설치 확인

```
>>> import matplotlib.pyplot as plt
```

```
>>>
```

```
관리자: 명령 프롬프트
C:\WINDOWS\system32>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.3.3-cp39-cp39-win_and64.whl (8.5 MB)
    |#####| 8.5 MB 96 kB/s
Requirement already satisfied: numpy>=1.15 in c:\python39\lib\site-packages (from matplotlib) (1.19.3)
Collecting cycler>=0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.1-cp39-cp39-win_and64.whl (51 kB)
    |#####| 51 kB 206 kB/s
Collecting pillow>=6.2.0
  Downloading Pillow-8.1.0-cp39-cp39-win_and64.whl (2.2 MB)
    |#####| 2.2 MB 156 kB/s
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
    |#####| 67 kB 244 kB/s
Collecting python-dateutil>=2.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
    |#####| 227 kB 94 kB/s
Collecting six
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: six, python-dateutil, pyparsing, pillow, kiwisolver, cycler, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.3.3 pillow-8.1.0 pyparsing-2.4.7 python-dateutil-2.8.1 six-1.15.0
C:\WINDOWS\system32>
```

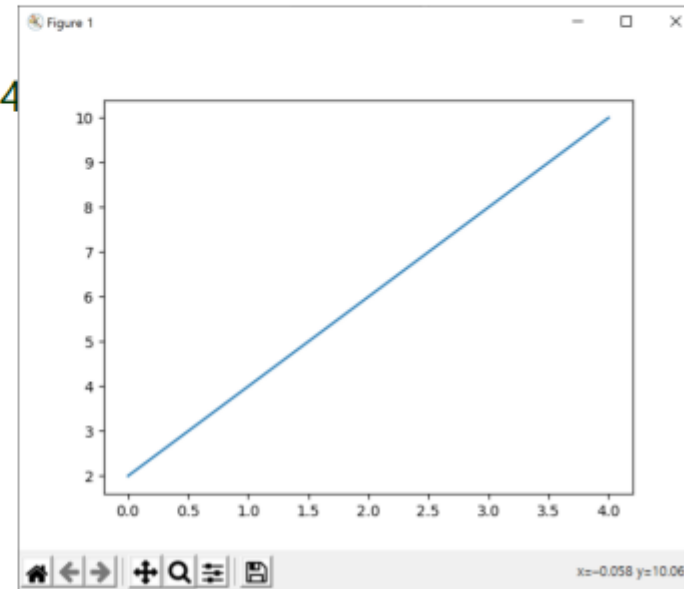


# 1. Matplotlib

## 막대 그래프 그리기 (plt.plot(A))

넘파이 배열을 전달하여 그래프 그리기

```
>>> import matplotlib.pyplot as plt
>>> B = [2, 4, 6, 8, 10]
# (0, 2), (1, 4), (2, 6), (3, 8), (4, 10)
>>> plt.plot(B)
[<matplotlib.lines.Line2D object at 0x000001B4...>]
>>> plt.show()
```

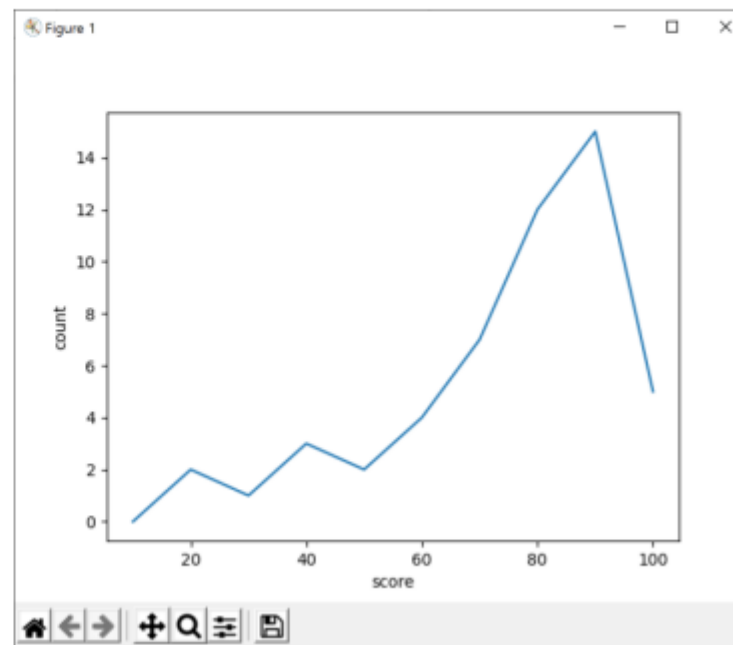


# 1. Matplotlib

## X, Y축 레이블 지정

```
import matplotlib.pyplot as plt  
X = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]  
Y = [0, 2, 1, 3, 2, 4, 7, 12, 15, 5]
```

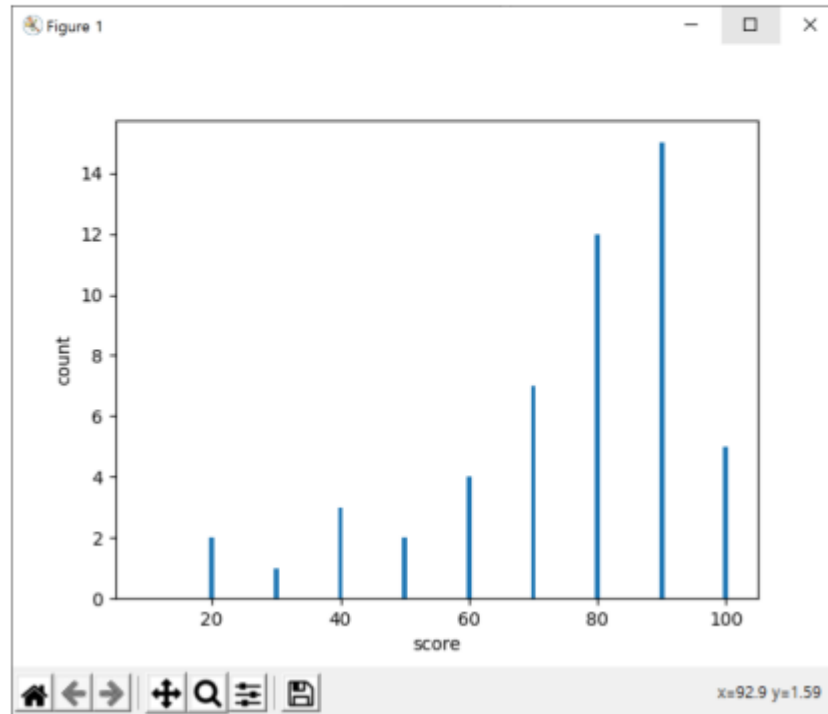
```
plt.plot(X, Y)      # X, Y축 값 할당  
plt.xlabel("score") # 레이블 지정  
plt.ylabel("count")  
plt.show()
```



# 1. Matplotlib

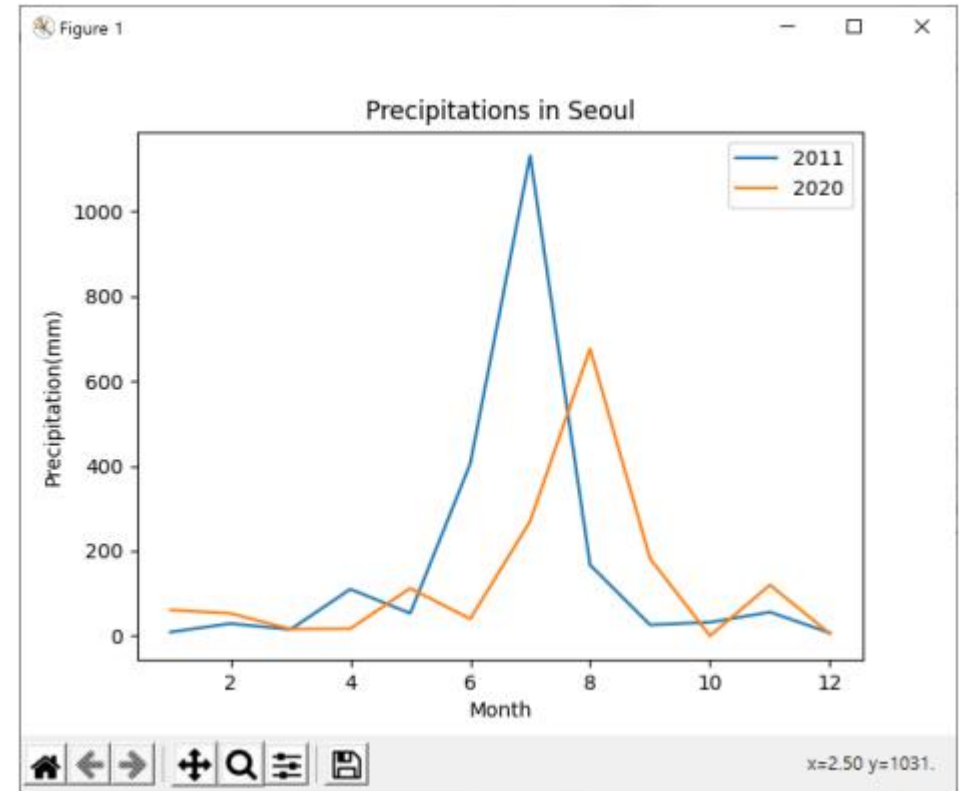
막대 그래프 (plt.bar(X,Y))

```
import matplotlib.pyplot as plt  
X = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]  
Y = [0, 2, 1, 3, 2, 4, 7, 12, 15, 5]  
plt.bar(X, Y)  
plt.xlabel("score")  
plt.ylabel("count")  
plt.show()
```



# 1. 예제 - 강수량 그래프 그리기

```
import matplotlib.pyplot as plt
X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Y2011 = [9, 29, 15, 110, 53, 405, 1131, 167, 26, 32, 56, 7]
Y2020 = [61, 53, 16, 17, 112, 40, 270, 676, 182, 0, 120, 5]
plt.plot(X, Y2011, label="2011")
plt.plot(X, Y2020, label="2020")
plt.xlabel("Month")
plt.ylabel("Precipitation(mm)")
plt.legend(loc="upper right")
plt.title("Precipitations in Seoul")
plt.show()
```



## 2. Numpy

---

파이썬 라이브러리 패키지

행렬 계산을 도와주는 함수

설치

```
C:\>pip install numpy
```

설치확인

```
import numpy as np
```

```
명령 프롬프트
C:\Users\Wsj>pip install numpy
Collecting numpy
  Downloading numpy-1.19.4-cp39-cp39-win_amd64.whl (13.0 MB)
    | 13.0 MB 192 kB/s
Installing collected packages: numpy
Successfully installed numpy-1.19.4
```



## 2. Numpy

---

### 넘파이 1차원 배열

`np.array(list)`

```
>>> temp = [1, 3, 5, 7, 9]
```

```
>>> A = np.array(temp)
```

```
>>> A
```

```
array([1, 3, 5, 7, 9])
```

```
>>> A * 2
```

```
array([ 2,  6, 10, 14, 18])
```

```
>>> A * 0.3
```

```
array([0.3, 0.9, 1.5, 2.1, 2.7])
```

```
>>> A = A * 0.3
```

```
>>> A
```

```
array([0.3, 0.9, 1.5, 2.1, 2.7])
```





## 2. 예제 - 배열 연산

### 배열 연산

- 자산에서 대출을 모두 공제한 후 자산 잔액을 계산.
- 배열 원소 또는 배열 간에 수학 연산자 사용 가능.
- 배열 원소에 접근 가능.

```
import numpy as np
asset = np.array([1000, 2000, 1500, 900])
debt = np.array([300, 1200, 0, 1500])
print("asset", asset)
print("debt ", debt)
asset = asset - 500
print("500 deducted each")
print("asset", asset)
```

### 실행결과

asset [1000 2000 1500 900]

debt [ 300 1200 0 1500]

500 deducted each

asset [ 500 1500 1000 400]



## 2. 예제 - 배열 연산

### 배열 연산

### 실행결과

balance after debt deduction

```
print("balance after debt deduction")
```

```
balance = asset - debt
```

```
print("balance", balance)
```

```
print("asset", asset)
```

```
print("balances over 500")
```

```
print(balance > 500)
```

```
asset = balance
```

```
debt = debt - debt
```

```
print("asset", asset[0], asset[1], asset[2], asset[3])
```

```
print("debt ", debt)
```

balance [ 200 300 1000 -1100]

asset [ 500 1500 1000 400]

balances over 1000

[False False True False]

asset 200 300 1000 -1100

debt [0 0 0 0]



### 3. 넘파이 arange

---

`np.arange(start, stop, step)`

Start 부터 stop 까지 step 만큼의 값으로 값을 생성한다.

```
>>> num = np.arange(1, 100)
```

```
[ 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 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
 97 98 99]
```



### 3. 넘파이 arange

---

`np.arange(start, stop, step)`

Start 부터 stop 까지 step 만큼의 값으로 값을 생성한다.

```
>>> odd = np.arange(1, 100, 2)
```

```
[ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47
 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95
 97 99]
```

```
>>> even = np.arange(2, 100, 2)
```

```
[ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48
 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96
 98]
```



### 3. 예제 - 넘파이로 구구단 출력하기

---

```
import numpy as np
```

```
first = np.arange(2, 10)
```

```
second = np.arange(1, 10)
```

```
for i in range (0, first.size):
```

```
    print(first[i]*second)
```

실행결과

[ 2 4 6 8 10 12 14 16 18]

[ 3 6 9 12 15 18 21 24 27]

[ 4 8 12 16 20 24 28 32 36]

[ 5 10 15 20 25 30 35 40 45]

[ 6 12 18 24 30 36 42 48 54]

[ 7 14 21 28 35 42 49 56 63]

[ 8 16 24 32 40 48 56 64 72]

[ 9 18 27 36 45 54 63 72 81]



### 3. 예제 - 구간 분할하기 1

`np.linspace(start, end, level)` #level 생략시 디폴트는 50

Start에서 end 범위를 level-1 개수의 구간으로 분할하는 level개의  
경계값을 배열을 반환

`np.linspace(1, 100, 7)` # 1~ 100을 6개 구간으로 분할

□  $(100-1)/6 = 16.5$

```
import numpy as np
```

```
a = np.linspace(1, 100, 7)
```

```
print(a)
```

실행결과

```
[ 1.  17.5  34.  50.5  67.  83.5 100.]
```

```
import numpy as np
```

```
a = np.linspace(50, 1000)
```

```
print (a)
```

실행결과

```
[ 50.          69.3877551  88.7755102 108.16326531 127.55102041
 146.93877551 166.32653061 185.71428571 205.10204082 224.48979592
 243.87755102 263.26530612 282.65306122 302.04081633 321.42857143
 340.81632653 360.20408163 379.59183673 398.97959184 418.36734694
 437.75510204 457.14285714 476.53061224 495.91836735 515.30612245
 534.69387755 554.08163265 573.46938776 592.85714286 612.24489796
 631.63265306 651.02040816 670.40816327 689.79591837 709.18367347
 728.57142857 747.95918367 767.34693878 786.73469388 806.12244898
 825.51020408 844.89795918 864.28571429 883.67346939 903.06122449
 922.44897959 941.83673469 961.2244898  980.6122449 1000.      ]
```



### 3. 예제 - 원소 초기화 방법

```
print( np.zeros(9) )  
print( np.zeros( (5, 5) ) )  
print( np.zeros( (3, 4, 5) ) )
```

실행결과

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

실행결과

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

```
x = np.full(2, 5)
```

```
y = np.full((3, 4), fill_value = 1)
```

```
z = np.full((3, 4, 5), fill_value = 3)
```

```
print("ndim ", x.ndim, y.ndim, z.ndim)
```

```
print("shape ", x.shape, y.shape, z.shape)
```

```
print("size ", x.size, y.size, z.size)
```

실행결과

```
ndim 1 2 3
```

```
shape (2,) (3, 4) (3, 4, 5)
```

```
size 2 12 60
```





### 3. 예제 - 인구 통계 그래프 그리기

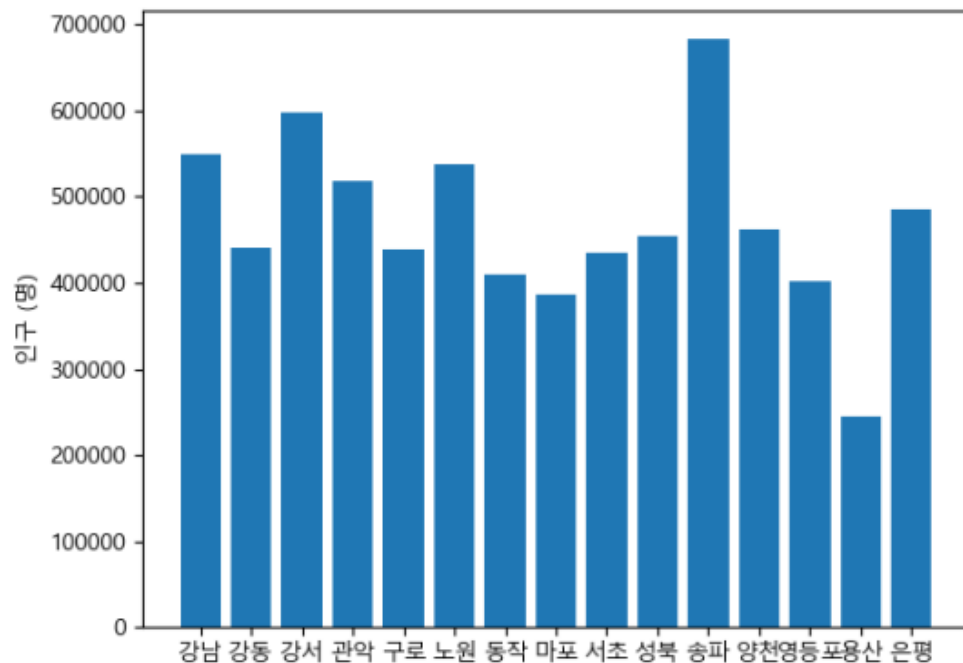
```
import numpy as np
import matplotlib.pyplot as plt

gu=['강남', '강동', '강서', '관악', '구로', '노원', '동작', '마포', '서초', '성북', '송파', '양천', '영등포', '용산',
'은평']

population=[550209, 440390, 598273, 517334, 439371, 537303, 408912, 385925, 435107, 454744,
682741, 462285, 400986, 245185, 484546]

area=[39.5, 24.59, 41.44, 29.57, 20.12, 35.44, 16.35, 23.85, 46.98, 24.57, 33.87, 17.41, 24.55, 21.87,
29.71]
```

```
np_gu = np.array(gu)
np_population = np.array(population)
np_area = np.array(area)
plt.bar(np_gu, np_population)
plt.ylabel("인구 (명)")
plt.show()
```





### 3. 예제 - 삼각함수 그리기

- `np.linspace(0, 4*np.pi, 100)` # 0 ~ 4파이 범위를 100-1개 구간으로 분할

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

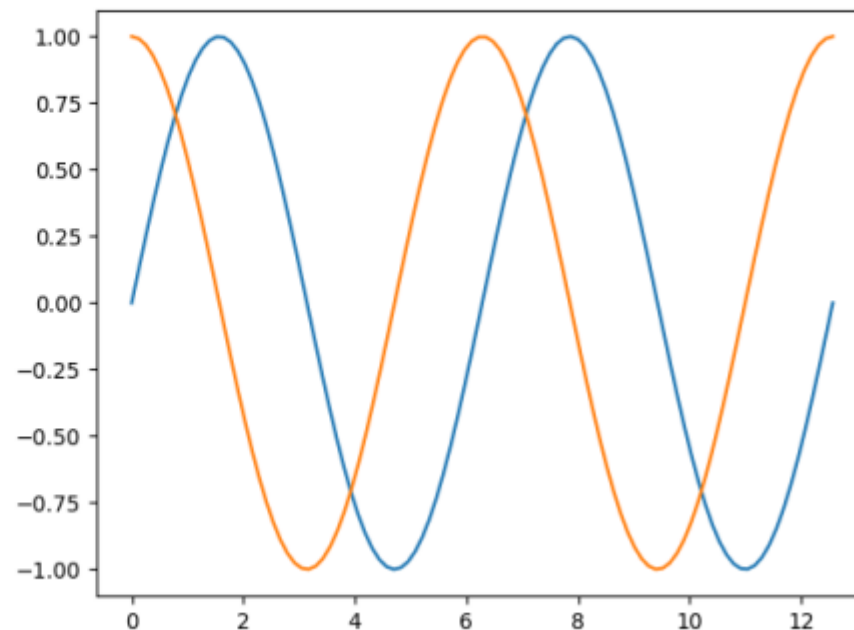
```
x = np.linspace(0, 4*np.pi, 100)
```

```
y1 = np.sin(x)
```

```
y2 = np.cos(x)
```

```
plt.plot(x, y1, x, y2)
```

```
plt.show()
```



# 과제 공지

---

소프트웨어의 이해



# 과제 제출

---

- 과제 제출 기한
  - 실습 **다음주 수요일(5월 24일) 오후 11시 00분**까지
- 제출 장소
  - **Snowboard** 해당 실습 과제 제출 페이지에 업로드

★ 표절 검사 및 기한 내 제출 필수!

# #1. 파일 쓰기 - fileWrite.py

문제) [ 국어 영어 ] 두 과목의 점수를 동시에 입력 받아 텍스트 파일에 저장하는 프로그램을 작성하시오.

- 공백으로 구분한 5쌍 이상의 [ 국어 영어 ] 점수를 입력 받아 리스트에 저장한다.
- 아무것도 입력하지 않고 Enter를 누를 경우, 입력을 종료하고 리스트의 점수를 파일 저장한다.

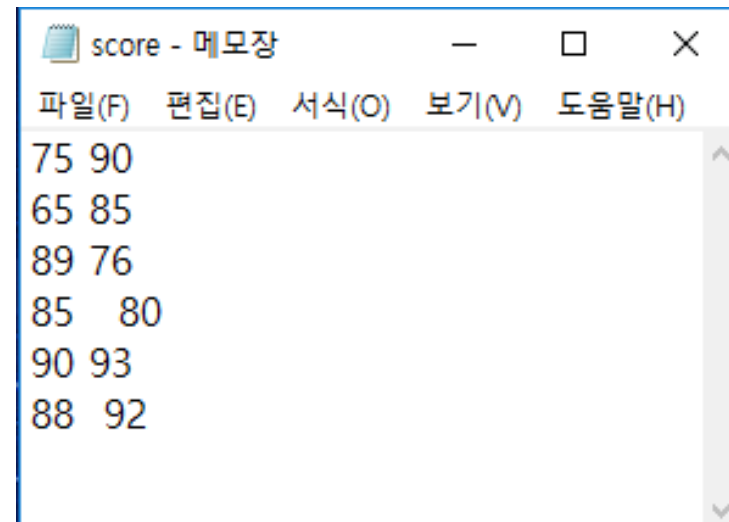
점수 텍스트 파일명 : score.txt

✓ fileWrite.py 실행화면

```
===== RESTART: E:\소프트웨어\익미해1
국어와 영어 점수를 입력하세요(종료= Enter)
예) 90 85
75 90
65 85
89 76
85 80
90 93
88 92

score.txt 저장 완료
>>>
```

✓ score.txt    보고서에 함께 첨부할 것.



```
score - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
75 90
65 85
89 76
85 80
90 93
88 92
```

## #2. 파일 읽기와 모듈 사용 - `fileRead.py`, `mod1.py`

- 저장된 `score.txt` 파일을 읽어와서 과목별 평균과 중간값을 출력한다. - `fileRead.py`
- 과목별 평균과 중간값을 계산하는 모듈을 작성한다. - `mod1.py`

### `fileRead.py`

- `strip()` , `split()` 을 사용할 것

✓ `fileRead.py` 실행화면

```
===== RESTART:
평균      국어   영어
중간값    (82.0, 86.0)
          (86.5, 87.5)
>>>
```

### `mod1.py`

- `mean(kor, eng)` : 각 과목 점수가 들어있는 리스트를 매개변수로 전달, 과목별 평균을 튜플로 반환
- `median(kor, eng)` : 각 과목 점수가 들어있는 리스트를 매개변수로 전달, 과목별 중간값을 튜플로 반환

### #3. 구구단 - multiple\_t.py

---

실행 결과와 같이 구구단 10단~19단을 출력하는 프로그램을 작성하시오.  
Numpy의 `arrange()` 함수를 이용하시오.

#### 실행결과

```
[10 20 30 40 50 60 70 80 90]
[11 22 33 44 55 66 77 88 99]
[ 12  24  36  48  60  72  84  96 108]
[ 13  26  39  52  65  78  91 104 117]
[ 14  28  42  56  70  84  98 112 126]
[ 15  30  45  60  75  90 105 120 135]
[ 16  32  48  64  80  96 112 128 144]
[ 17  34  51  68  85 102 119 136 153]
[ 18  36  54  72  90 108 126 144 162]
[ 19  38  57  76  95 114 133 152 171]
```

# 과제 제출 주의사항

---

소프트웨어의 이해



# 과제 제출 주의사항

---

- 과제 제출 기한
  - 실습 다음주 수요일 오후 11시 00분까지
- 제출 장소
  - **Snowboard** 해당 실습 과제 제출 페이지에 업로드

★ 표절검사!






# 과제 제출

---

- 과제 파일 형식

- 소스파일(.py)과 보고서(.pdf)를 한 폴더에 넣고 압축(.zip)하여 제출
- 파일명 : **Lab과제번호\_학번\_이름.zip**  
ex) Lab01\_2201234\_김눈송.zip

**\* 반드시 압축 파일 이름과 내부 파일 이름을 지켜 주시기 바랍니다\***

|                                                                                                     |                    |                     |
|-----------------------------------------------------------------------------------------------------|--------------------|---------------------|
|  Lab01_2201234_김눈송 | 2022-03-02 오후 9:00 | Microsoft Edge P... |
|  octagon           | 2022-03-02 오후 8:40 | Python File         |
|  triangle          | 2022-03-02 오후 8:52 | Python File         |

- 소스파일(.py)

- 파일명 : 매 실습마다 제공하는 실습 자료 이름  
ex) triangle.py

- 보고서(.pdf)

- 파일명 : Lab과제번호\_학번\_이름 .pdf  
ex) “Lab01\_2201234\_김눈송.pdf”
- 보고서 포함 항목 : 문제-기능별 실행화면 캡처, 소스코드 텍스트

# 과제 질문 주의사항



조교 이메일  
nayeonjo@sookmyung.ac.kr

1. 질문이 생길 경우, **Q&A 게시판에 글 작성**
2. Q&A 게시판 질문 시, **반드시 설명과 함께 질문**
3. **과목, 분반, 전공, 이름, 학번 작성 필수**
4. 주말에는 메일 답장이 없을 수 있음
5. 실습 제출 마감 당일에는 답장이 늦을 수 있음

## Office Hour

메일로 약속 시간을 미리 정하고 방문해주세요.

## 메일 예시

컴퓨터학과 전공 2301234 소프트웨어의 이해 3분반  
김눈송 입니다.

n번 과제의 코드 4번째 줄에서 에러가 발생해요.  
or  
1번 과제에서 테스트 파일 업로드 과정에서 문제가  
발생했어요.

코드를 첨부해서 메일 보내드려요.