jupyter notebook 단축키

A : 위쪽에 셀 삽입

B : 아래쪽에 셀 삽입

X : 해당 셀 삭제

Z : 삭제할 셀 복구

M : Mark Down

ctrl + D : 해당 커서 라인 삭제

ctrl + A : 해당 셀 전체 선택

ctrl + Z : 셀 내용 복구

python에서 주석 처리는 # 여러줄 주석 처리는 '''

ctrl + / : 한꺼번에 주석처리

ctrl + Enter : 셀 실행

shift + Enter : 셀 실행 후 커서 아래쪽으로 이동

In [1]:
```
# 파이썬에서 주석처리
# 필기할 내용이 있으면 이렇게 주석처리 하시면 됩니다
```

In [ ]:
```
'''
여러줄 주석처리 하고 싶을때 사용
나중에 class라는 개념을 배울때 자주 사용됩니다
수업 시간에서는 한줄 주석이 많이 사용되니까 #을 활용해주세요
'''
```

range(start, end, by)

start에서 시작해서 end까지 by만큼 증가

In [5]:
```
# 1 ~100 사이의 홀수만 출력
range(1, 101, 2)
```

Out[5]:
```
range(1, 101, 2)
```

In [6]:
```
# 1 ~100 사이의 짝수만 출력
range(2, 101, 2)
```

Out[6]:
```
range(2, 101, 2)
```

for문

In [8]:
```
for i in range(1, 101, 2):
    print(i)
```

```
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
```

구구단(2단~9단) 만들어보기

2 * 1 = 2

2 * 2 = 4

2 * 3 = 6

2 * 4 = 8

2 * 5 = 10

2 * 6 = 12

2 * 7 = 14

2 * 8 = 16

2 * 9 = 18

In [15]:
```
for i in range(2, 10):
    for j in range(1, 10):
        print(i, "*", j, "=", i*j)
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
4 * 1 = 4
```

피라미드 만들어보기

In [ ]:
```
# HINT
# *을 4번 출력할 경우
print('*' * 4)
```

In [1]:
```

```

```
   *
  ***
 *****
*******
```

if 문

In [19]:
```
# 1~100 사이의 짝수 홀수 출력
for i in range(1, 101):
    if i % 2 == 0:
        print(i)

for i in range(1, 101):
    if i % 2 == 1:
        print(i)
```

```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
```

list 자료 구조

In [26]:
```
# [] 대괄호를 통하여 list자료 구조를 선언
odd = [1, 3, 5]
odd

# for문을 이용하여 1~100사이의 홀수를 list안에 담기
odd_100 = [x for x in range(1, 101, 2)]
odd_100
```

Out[26]:
```
[1,
 3,
 5,
 7,
 9,
 11,
 13,
 15,
 17,
 19,
 21,
 23,
 25,
 27,
 29,
 31,
 33,
 35,
 37,
```

In [ ]:

In [27]:
```
# len() : list의 길이 확인
len(odd_100)
```

Out[27]:
```
50
```

In [28]:
```
for i in odd_100:
    print(i)
```

```
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
```

In [38]:
```
# list slicing

# odd_100에서 1~10번째 값만 추출
print('1~10번째 값 : ', odd_100[0:10])
print('1~10번째 값 : ', odd_100[:10])
print(len(odd_100[0:10]))

# odd_100에서 40~50번째 값만 추출
print('41~50번째 값 : ', odd_100[40:50])
print('41~50번째 값 : ', odd_100[40:])

# odd_100에서 마지막 값 추출
print('마지막 값 : ', odd_100[49])
print('마지막 값 : ', odd_100[-1])
```

```
1~10번째 값 :  [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
1~10번째 값 :  [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
10
41~50번째 값 :  [81, 83, 85, 87, 89, 91, 93, 95, 97, 99]
41~50번째 값 :  [81, 83, 85, 87, 89, 91, 93, 95, 97, 99]
마지막 값 :  99
마지막 값 :  99
```

In [45]:
```
# [start:end:by] 세미콜론 2개를 통한 index slicing
num = [x for x in range(1, 101)]
# num[0:100:2]
print(num[::2])

# 역순 출력
print(num[::-1])
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 4
1, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 7
9, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99]
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82,
81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 6
2, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 4
3, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 2
4, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4,
3, 2, 1]
```

In [48]:
```
# append() : list에 값 추가
num.append(101)
num

# odd_100에 방금 추가한 값 확인해보기
num[-1]
```

Out[48]:
```
101
```

In [52]:
```
# del  : list 값 삭제

# 괄호안에 해당 index를 통하여 접근
del num[-1]
num
```

Out[52]:
```
[1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17,
 18,
 19,
```

In [55]:
```
# 나이를 넣은 list 생성
age = [22, 21, 23, 25, 24, 20]

# sort() : list 정렬
age.sort()
print(age)
print(age[::-1])
```

```
[20, 21, 22, 23, 24, 25]
[25, 24, 23, 22, 21, 20]
```

dictionary 자료 구조

key : value라는 구조로 대응 관계를 표현

ex) {'name' : '양용준', 'school' : '연세대', 'age' : 'secret'}

{}를 통하여 표현하고 ,를 통해 구분

list: []

dictionary: {}

In [56]:
```
dic = {'name': '박세희', 'school': '숙명여대', 'age': 23}
dic
```

Out[56]:
```
{'name': '박세희', 'school': '숙명여대', 'age': 23}
```

In [63]:
```
# key를 통하여 value 접근
dic['name']
dic['school']
dic['age']
```

Out[63]:
```
23
```

In [95]:
```
# dictionary에 key, value 추가

# value 수정
dic['alcohol'] = 1
dic['alcohol']
```

Out[95]:
```
1
```

In [96]:
```
dic['age'] = 23
dic['age']
```

Out[96]:
```
23
```

In [97]:
```
# del dictionary[key] : key를 통해 key:value 한쌍 지우기
del dic['alcohol']
dic
```

Out[97]:
```
{'name': '박세희', 'school': '숙명여대', 'age': 23}
```

In [98]:
```
# dictionary key 모음
dic.keys()

# dictionary value 모음
dic.values()

# for문을 통하여 key를 통해 각 key에 해당하는 value 접근
for i in dic.keys():
    print(dic[i])
```

```
박세희
숙명여대
23
```

In [109]:
```
# dictionary key,value 모음
dic.items()

# for문을 통하여 key, value 접근
for k,v in dic.items():
    print('key', k, '. value :', v)
```

```
key name . value : 박세희
key school . value : 숙명여대
key age . value : 23
```

In [110]:
```
# 새로운 dictionary 생성
dic = {'a': 2, 'b': 3, 'c': 1, 'd': 7, 'e': 5}
dic
```

Out[110]:
```
{'a': 2, 'b': 3, 'c': 1, 'd': 7, 'e': 5}
```

In [111]:
```
dic.items()
```

Out[111]:
```
dict_items([('a', 2), ('b', 3), ('c', 1), ('d', 7), ('e', 5)])
```

In [112]:
```
# sorted(a, key = lambda x : 정렬 기준)

# value를 기준으로 정렬
sorted(dic.items(), key = lambda x:x[1])
```

Out[112]:
```
[('c', 1), ('a', 2), ('b', 3), ('e', 5), ('d', 7)]
```

Set 자료구조

집합 : 데이터 분석에서 Set 자료구조를 사용하는 이유는 중복된 값을 제거 해주기 때문

In [115]:
```
# set 자료구조 만들기
set1 = {1, 2, 3, 4, 5, 6, 6, 6}
set2 = {4, 5, 6, 7, 8, 9}

set1
```

Out[115]:
```
{1, 2, 3, 4, 5, 6}
```

In [118]:
```
# intersection : 교집합
set1 & set2
set1.intersection(set2)

set1 & set2 == set1.intersection(set2)
```

Out[118]:
```
True
```

In [122]:
```
# union: 합집합
set1 | set2
set1.union(set2)

set1 | set2 == set1.union(set2)
```

Out[122]:
```
True
```

In [130]:
```
# difference :차집합
print('set1 - set2 :', set1 - set2)
print('set2 - set1 :', set2 - set1)

print('set1 - set2 :', set1.difference(set2))
print('set2 - set1 :', set2.difference(set1))
```

```
set1 - set2 : {1, 2, 3}
set2 - set1 : {8, 9, 7}
set1 - set2 : {1, 2, 3}
set2 - set1 : {8, 9, 7}
```