3. 파이썬의 기본

3.1 자료형과 인덱싱

3.1.1. list 자료형

In [38]:

```
a=[[0,1,2,3],
   [4,5,6,7],
   [8,9,10,11]] #행렬을 행 순서로 입력, nested list

print(type(a), a)
```

<class 'list'> [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]

In [2]:

```
print(a[0]) # 첫번째 행
print(a[0][1]) # 1행 2열
print(a[2][3]) # 3행 4열
#print(a[1,1]) error
```

```
[0, 1, 2, 3]
1
11
```

3.1.2. array 자료형

In [2]:

```
import numpy as np

x = np.array(range(16))
print(x)
type(x)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
```

Out[2]:

numpy.ndarray

In [40]:

```
x = np.reshape(x,(4,4)) # numpy package, matrix

print(x)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

In [5]:

```
print(x)
print(x[0])
print(x[0:3,1:4])
print(x[:,0:2])
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[0 1 2 3]
[[ 1  2  3]
 [ 5  6  7]
 [ 9 10 11]]
[[ 0  1]
 [ 4  5]
 [ 8  9]
 [12 13]]
```

In [6]:

```
print(x[0][1], x[0,1])
```

```
1 1
```

3.1.3. dataframe 자료형

In [45]:

```
import pandas as pd

df=pd.read_csv("cdc.txt", sep=" ")
df.head()
```

Out[45]:

	genhlth	exerany	hlthplan	smoke100	height	weight	wtdesire	age	gender
1	good	0	1	0	70	175	175	77	m
2	good	0	1	1	64	125	115	33	f
3	good	1	1	1	60	105	105	49	f
4	good	1	1	0	66	132	124	42	f
5	very good	0	1	0	61	150	130	55	f

In [42]:

```
df.genhlth #genhlth 열 추출
#df['genhlth'] 같은 결과
```

Out[42]:

1	good
2	good
3	good
4	good
5	very good
6	very good
7	very good
8	very good
9	good
10	good
11	excellent
12	fair
13	excellent
14	excellent
15	fair
16	good
17	good
18	fair
19	good
20	very good
21	very good
22	good
23	very good
24	good
25	very good
26	good
27	excellent
28	excellent
29	very good
30	excellent
...	...
19971	excellent
19972	very good
19973	excellent
19974	fair
19975	good
19976	fair
19977	good
19978	very good
19979	good
19980	good
19981	excellent
19982	very good
19983	very good
19984	very good
19985	good
19986	very good
19987	excellent
19988	good
19989	very good
19990	good
19991	excellent
19992	very good
19993	very good
19994	very good
19995	good
19996	good
19997	excellent
19998	poor

```
19999         good
20000         good
Name: genhlth, Length: 20000, dtype: object
```

In [9]:

```
df[['genhlth','exerany']]
```

Out[9]:

	genhlth	exerany
1	good	0
2	good	0
3	good	1
4	good	1
5	very good	0
6	very good	1
7	very good	1
8	very good	0
9	good	0
10	good	1
11	excellent	1
12	fair	1
13	excellent	1
14	excellent	1
15	fair	1
16	good	1
17	good	0
18	fair	0
19	good	1
20	very good	1
21	very good	1
22	good	1
23	very good	0
24	good	1
25	very good	0
26	good	0
27	excellent	1
28	excellent	1
29	very good	1
30	excellent	1
...	...	...
19971	excellent	1
19972	very good	1
19973	excellent	1
19974	fair	1
19975	good	0
19976	fair	1

	genhlth	exerany
19977	good	1
19978	very good	0
19979	good	0
19980	good	1
19981	excellent	1
19982	very good	0
19983	very good	1
19984	very good	1
19985	good	1
19986	very good	1
19987	excellent	1
19988	good	1
19989	very good	1
19990	good	1
19991	excellent	1
19992	very good	1
19993	very good	1
19994	very good	0
19995	good	0
19996	good	1
19997	excellent	0
19998	poor	0
19999	good	1
20000	good	1

20000 rows × 2 columns

In [10]:

```
print(df.head())
print(df.iloc[0,0])
print(df.iloc[0:3,3:5])
```

```
     genhlth  exerany  hlthplan  smoke100  height  weight  wtdesire  age  ₩
1       good        0         1         0      70     175       175   77
2       good        0         1         1      64     125       115   33
3       good        1         1         1      60     105       105   49
4       good        1         1         0      66     132       124   42
5  very good        0         1         0      61     150       130   55

  gender
1      m
2      f
3      f
4      f
5      f
good
   smoke100  height
1         0      70
2         1      64
3         1      60
```

In [11]:

```
df.loc[df.exerany==0,]
```

Out[11]:

	genhlth	exerany	hlthplan	smoke100	height	weight	wtdesire	age	gender
1	good	0	1	0	70	175	175	77	m
2	good	0	1	1	64	125	115	33	f
5	very good	0	1	0	61	150	130	55	f
8	very good	0	1	0	67	170	160	45	m
9	good	0	1	1	65	150	130	27	f
17	good	0	0	1	67	156	150	47	m
18	fair	0	1	1	71	185	185	76	m
23	very good	0	1	1	73	160	160	43	m
25	very good	0	0	1	64	105	120	27	f
26	good	0	1	0	68	190	150	33	f
46	excellent	0	1	0	59	145	110	36	f
50	good	0	1	0	61	141	140	63	m
51	good	0	1	0	65	179	150	33	f
59	good	0	1	0	64	120	110	45	f
64	fair	0	1	0	60	160	140	56	f
65	very good	0	0	0	74	164	164	80	m
67	excellent	0	1	0	66	140	150	34	f
70	good	0	1	1	71	190	190	45	m
77	good	0	1	0	64	112	124	71	f
82	very good	0	0	0	63	120	120	30	m
88	fair	0	1	0	60	115	115	27	f
89	very good	0	0	0	66	193	170	40	f
92	very good	0	1	0	67	166	150	55	f
93	good	0	1	0	64	145	130	36	f
99	very good	0	1	0	74	180	180	58	m
100	good	0	1	1	60	100	100	28	f
104	fair	0	1	0	61	114	114	36	f
107	fair	0	0	1	66	140	136	24	m
108	very good	0	1	1	70	193	165	61	m
113	very good	0	1	0	65	132	132	75	f
...	...	...	...	...	...	...	...	...	...
19884	good	0	1	1	73	158	170	38	m
19886	very good	0	1	0	62	250	140	32	f
19889	very good	0	1	0	62	140	120	54	f
19891	very good	0	1	1	72	150	135	28	m
19895	good	0	1	0	66	150	130	31	f
19897	excellent	0	0	1	69	135	130	24	f

	genhlth	exerany	hlthplan	smoke100	height	weight	wtdesire	age	gender
19898	fair	0	1	0	57	100	100	79	f
19899	fair	0	1	1	68	215	180	72	f
19901	fair	0	1	1	64	160	140	68	f
19906	good	0	1	1	63	194	150	24	f
19907	poor	0	0	1	64	185	150	63	f
19909	very good	0	1	1	62	118	118	87	f
19914	fair	0	1	1	68	160	160	53	m
19920	excellent	0	1	0	68	160	160	25	f
19922	fair	0	0	1	68	125	140	55	f
19927	good	0	1	0	66	145	125	35	f
19929	good	0	1	0	66	265	190	26	f
19930	very good	0	1	0	61	150	120	42	f
19941	very good	0	1	1	65	125	120	38	f
19944	excellent	0	0	1	65	150	135	33	f
19965	fair	0	1	0	63	183	140	68	f
19967	good	0	1	0	64	180	150	71	f
19975	good	0	1	0	66	156	150	79	f
19978	very good	0	1	1	65	180	150	87	f
19979	good	0	1	1	64	135	135	39	m
19982	very good	0	1	1	74	210	210	40	m
19994	very good	0	1	1	63	165	120	31	f
19995	good	0	1	1	69	224	224	73	m
19997	excellent	0	1	0	73	200	185	35	m
19998	poor	0	1	0	65	216	150	57	f

5086 rows × 9 columns

3.2 계산함수

In [12]:

```
import math
import numpy as np
```

In [13]:

```
c = 9
print(math.sqrt(c)) # 제곱근
print(np.sqrt(c))
```

```
3.0
3.0
```

In [14]:

```
print(math.exp(1), np.exp(1)) # 자연상수 e의 지수함수
print(math.log(10), np.log(10)) # 10의 자연로그
print(math.log(10,10)) # 10의 밑이 10인 로그
np.pi
```

```
2.718281828459045 2.718281828459045
2.302585092994046 2.302585092994046
1.0
```

Out[14]:

```
3.141592653589793
```

In [15]:

```
print(math.pi, np.pi) # 원주율 π
print(math.sin(math.pi/2), np.sin(np.pi/2)) # 삼각함수
```

```
3.141592653589793 3.141592653589793
1.0 1.0
```

3.3 추출함수

In [16]:

```
print(np.random.randint(0, 9, size = (2,2) ))
alp=["a","b","c","d","e","f"]
print(np.random.choice(alp))
print(np.random.choice(alp,5,replace=True)) # 복원추출
print(np.random.choice(alp,5,replace=False)) # 비복원추출
print(np.random.uniform(low=0,high=1,size=1))
print(np.random.normal(3,1.5, 1)) # 평균 3, 표준편차 1.5인 정규분포에서 랜덤 넘버 생성
```

```
[[5 5]
 [4 0]]
a
['c' 'e' 'f' 'b' 'e']
['c' 'f' 'b' 'e' 'd']
[0.8995429]
[3.78317085]
```

3.4 조건문과 반복문

3.4.1 조건문

조건문은 논리형으로 표현되는 특정한 조건의 성립 여부에 따라 다른 작업을 수행한다. 조건문의 기본 구조는 다음과 같으며, elif나 else 부분은 반드시 넣을 필요는 없다.

if 조건1: 조건1이 참이면 실행할 명령어

조건1이 거짓일 때만 아래를 실행한다.

elif 조건2: 조건2가 참이면 실행할 명령어

조건2도 거짓일 때만 아래를 실행한다.

In [5]:

```
a = 1

if a==1:
    print('a is %i' %a)
```

```
a is 1
```

조건문, 반복문, 함수 등에서 각 부분의 실행할 명령어는 들여쓰기를 해야한다. 들여쓰기를 하려면 Tab키를 한 번 눌러주면 되는데, 노트북에서는 자동으로 들여쓰기를 해준다.

In [6]:

```
if a!=1:
print('a is not 1' %a) ## error
```

```
  File "<ipython-input-6-9bc7914967bd>", line 2
    print('a is not 1' %a) ## error
        ^
IndentationError: expected an indented block
```

In [7]:

```
b = 3

if b==3:
    print('b=%i' %b)
else:
    print('b!=%i %b')
```

```
b=3
```

In [8]:

```
a = 100

if a%2==0:
    print('a is even')
else:
    print('a is odd')
```

```
a is even
```

In [9]:

```
#elif

a = 101

if a%2 ==0 :
    print('a is divided by 2')
elif a%3==0:
    print('a is divided by 3')
else:
    print('a is neither divided 2 nor 3')
```

```
a is neither divided 2 nor 3
```

3.4.2 반복문

반복문은 비슷한 작업을 여러 번 수행하며, for문과 while문 두 종류가 있다. 먼저 for문은 특정한 범위 안에서 반복하며, 기본 구조는 다음과 같다.

In [10]:

```
for i in 수행 범위:
    실행할 명령어
```

```
  File "<ipython-input-10-b27f6a2e31fa>", line 1
    for i in 수행 범위:
                 ^
SyntaxError: invalid syntax
```

수행 범위에는 list, np.array 등 수열 형태의 자료가 들어갈 수 있는데, 보통 반복문에 특화된 자료형인 range를 많이 사용한다.

In [11]:

```
a = range(10) #0부터 10미만의 정수
print(a)

b = range(0,10)
print(b)

c = range(-10,0)
print(c)
```

```
range(0, 10)
range(0, 10)
range(-10, 0)
```

In [12]:

```
list(a)
```

Out[12]:

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

In [13]:

```
# 1부터 100까지 더하는 함수
sum = 0
for i in range(1,101):
    sum = sum+i

print(sum)
```

5050

In [14]:

```
# 1부터 9까지의 수중에서 홀수를 찾아내는 함수
seq = []
for i in range(1,10):
    if i%2==0:
        pass
    else:
        seq.append(i)

print(seq)
```

[1, 3, 5, 7, 9]

break은 실행 중인 반복문을 멈추고 빠져나오게 하는 명령어이다.

In [3]:

```
a = []

for i in range(1,100):
    a.append(i)
    if np.sum(a)>=100:
        print(np.sum(a))
        break #break 미입력시 계속 돌아감.

print(a)
```

105
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

while문은 특정 조건을 만족하는 동안 계속해서 반복한다. 따라서 무한히 실행되는 것을 피하기 위해 조건문 안에서 조건에 관련된 변수를 변화하거나 필요할 때 break을 사용해야 한다.

In [16]:

```
while 조건:
    실행할 명령어
# 조건을 만족하지 않게 되면 다음으로 넘어온다.
```

```
  File "<ipython-input-16-1bf520d95ebf>", line 2
    실행할 명령어
          ^
SyntaxError: invalid syntax
```

In [17]:

```
a=[]
i=1
while len(a)<10:
    a.append(i)
    i += 1
print(a)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [18]:

```
# while문은 무한루프에 빠질수 있으니 break을 적절히 사용하자

'''
while True:
    print('계속 돌아갑니다')
'''
```

Out[18]:

```
"₩nwhile True:₩n    print('계속 돌아갑니다')₩n"
```

In [19]:

```
i = 0
while True:
    print('계속 돌아갑니다')
    i += 1
    if i>10:
        break
```

```
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
계속 돌아갑니다
```

continue는 반복문 안에서 이후의 실행할 명령어를 무시하고 다음 반복 단계로 넘어가게 한다.

In [20]:

```
a=[]
i=0
while i<20:
    i+=1
    if i%2==0: continue # do not run print(i)
    print(i)
```

```
1
3
5
7
9
11
13
15
17
19
```

3.5 사용자 정의 함수

사용자가 원하는 기능을 하는 함수를 직접 만들 수 있다. 기본 형식은 다음과 같다.

In [ ]:

```

```

In [32]:

```
def 이름(입력 변수):
    실행 문장
    return 출력 변수
```

```
  File "<ipython-input-32-dfec164397bd>", line 1
    def 이름(입력 변수):
                  ^
SyntaxError: invalid syntax
```

In [53]:

```
def sum(a,b):
    return(a+b)

print(sum(1,3))
print(sum('a','b'))
```

```
4
ab
```

In [54]:

```
## print와 return의 차이

def sum1(a,b):
    print(a+b)

b = sum1(1,2)
```

3

In [55]:

```
print(b)
```

None

In [56]:

```
# 입력변수가 몇개가 될 지 모를 때

def sum(*args):
    value = 0
    for i in args:
        value += i
    return(value)

a = sum(10,11,12,13,14)
print(a)
```

60