1일차 문제

1. numpy

000 numpy 모듈 불러오기 및 버전 확인

- numpy 모듈을 불러오고 버전을 확인해보세요.
- numpy 모듈을 np로 줄여 사용할 수 있도록 작성하세요.

001 타입 확인

- 다음 `data` 변수의 타입을 확인하세요.

In

```
data = np.array([1, 2, 3])
# 주석을 지우고 이곳에 코드를 작성하세요.
```

Out

```
numpy.ndarray
```

002 배열생성

- data 변수를 생성하고 아래의 출력과 같은 값이 나오도록 배열을 생성하세요.

In

```
# 주석을 지우고 이곳에 코드를 작성하세요.
data
```

Out

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

003 배열생성

- `data1` 변수를 생성하고 다음의 출력과 같은 값이 나오도록 배열을 생성하세요.

In

```
# 주석을 지우고 이곳에 코드를 작성하세요.
data1
```

Out

```
array([[3.5 , 4.31, 1.12],
       [9.  , 1.  , 3.  ]])
```

004 차원 확인(ndim)

- `data1` 과 `data2` 가 각각 몇 차원인지 예상해보고 확인하세요.

In

```
data1 = np.array([3, 1, 7, 9])
data2 = np.array([[3.5, 4.31, 1.12], [9, 1, 3]])

print(data1) # 주석을 지우고 print 안에 코드를 작성하세요.
print(data2) # 주석을 지우고 print 안에 코드를 작성하세요.
```

005 차원 구조 확인(shape)

- `data1` 과 `data2` 의 차원 구조가 어떨지 예상해보고 확인하세요.

In

```
data1 = np.array([3, 1, 7, 9])
data2 = np.array([[3.5, 4.31, 1.12], [9, 1, 3]])

print(data1) # 주석을 지우고 print 안에 코드를 작성하세요.
print(data2) # 주석을 지우고 print 안에 코드를 작성하세요.
```

006 데이터 타입 확인(dtype)

- `data1` 과 `data2` 의 데이터 타입을 예상해보고 확인하세요.

In

```
data1 = np.array([3, 1, 7, 9])
data2 = np.array([[3.5, 4.31, 1.12], [9, 1, 3]])

print(data1) # 주석을 지우고 print 안에 코드를 작성하세요.
print(data2) # 주석을 지우고 print 안에 코드를 작성하세요.
```

007 데이터 개수 확인(size)

- `data1` 과 `data2` 의 데이터 개수를 예상해보고 확인하세요.

In

```
data1 = np.array([3, 1, 7, 9])
data2 = np.array([[3.5, 4.31, 1.12], [9, 1, 3]])

print(data1) # 주석을 지우고 print 안에 코드를 작성하세요.
print(data2) # 주석을 지우고 print 안에 코드를 작성하세요.
```

008 데이터 타입 지정(dtype)

- `dtype` 이 `int32` 인 `data` 의 타입을 `float64` 로 변경하세요.

In

```
data = np.array([3, 1, 7, 9])
print(data.dtype)
new_data =  # 주석을 지우고 이곳에 코드를 작성하세요.
print(new_data.dtype)
```

Print

```
int32
float64
```

009 행·열의 교환, 전치(Transpose)

- `data` 의 행과 열을 교환하세요.

In

```
data = np.array([[3., 4., 1., 5.], [9., 1., 3., 2.]])
print(data) # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
[[3. 9.]
 [4. 1.]
 [1. 3.]
 [5. 2.]]
```

010 형태 변경(reshape)

- `data` 의 데이터 형태를 보기의 출력값처럼 변경하세요.

In

```
data = np.array([[3., 4., 1., 5.], [9., 1., 3., 2.]])
print(data.reshape(8, 1)) # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
[[3.]
 [4.]
 [1.]
 [5.]
 [9.]
 [1.]
 [3.]
 [2.]]
```

011 데이터 생성(arange)

- `arange`를 사용하여 보기와 같은 데이터를 생성하세요.

In

```
data1 = # 주석을 지우고 이곳에 코드를 작성하세요.
data2 = # 주석을 지우고 이곳에 코드를 작성하세요.
print(data1)
print(data2)
```

Print

```
[0 1 2 3 4]
[0 2 4 6 8]
```

012 데이터 생성(linspace)

- `linspace`를 사용하여 보기와 같은 데이터를 생성하세요.

In

```
data1 = # 주석을 지우고 이곳에 코드를 작성하세요.
data2 = # 주석을 지우고 이곳에 코드를 작성하세요.
print(data1)
print(data2)
```

Print

```
[10.  5.  0.]
[  0.  25.  50.  75. 100.]
```

013 데이터 생성(eye)

- `eye`를 사용하여 보기와 같은 데이터를 생성하세요.

In

```
data1 = # 주석을 지우고 이곳에 코드를 작성하세요.
data2 = # 주석을 지우고 이곳에 코드를 작성하세요.
print(data1)
print(data2)
```

Print

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

014 데이터 생성(zeros, ones)

- `zeros`와 `ones`를 사용하여 보기와 같은 데이터를 생성하세요.

In

```
data1 = # 주석을 지우고 이곳에 코드를 작성하세요.
data2 = # 주석을 지우고 이곳에 코드를 작성하세요.
print(data1)
print(data2)
```

Print

```
[0. 0. 0. 0. 0.]
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

015 데이터 생성(full)

- `full` 을 사용하여 보기와 같은 데이터를 생성하세요.

In

```
data = # 주석을 지우고 이곳에 코드를 작성하세요.
print(data)
```

Print

```
[[5 5 5 5]
 [5 5 5 5]]
```

016 데이터 생성(randn)

- `randn` 을 사용하여 보기와 같은 정규(가우시안)분포 범위의 난수를 생성하세요.

In

```
data = # 주석을 지우고 이곳에 코드를 작성하세요.
print(data)
```

Print

```
[[-0.69190273 -0.52112558 -1.72167669 -0.77865483]
 [ 0.83461144  0.06283456 -0.84170318  1.11110502]]
```

017 기본 연산

- 기본 연산을 활용하여 보기와 같은 값이 나올 수 있도록 코드를 작성하세요.

In

```
data1 = np.array([[3, 1], [2, 6]])
data2 = np.array([[4, 0], [6, 6]])
print() # 주석을 지우고 print 안에 코드를 작성하세요.
print() # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
[[-1  1]
 [-4  0]]
[[ 7  1]
 [ 8 12]]
```

018 기본 연산

- 기본 연산을 활용하여 보기와 같은 값이 나올 수 있도록 코드를 작성하세요.

In

```
data1 = np.array([[3, 1], [2, 6]])
data2 = np.array([[4, 0], [6, 6]])
print() # 주석을 지우고 print 안에 코드를 작성하세요.
print() # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
[[12  0]
 [12 36]]
[[1.33333333 0.        ]
 [3.         1.        ]]
```

019 기본연산

- 다음은 랜덤한 데이터를 생성하고 데이터를 연산하는 코드입니다.
- 기본연산 함수를 사용하여 요구하는 값을 print하세요.
- 코드는 반드시 print문 안의 중괄호{}에 작성하세요.

In

```
data = np.array(np.random.randn(2, 5) * 10, dtype=int)
display(data)
print(f'전체 합계: {}') # 주석을 지우고 중괄호 안에 코드를 작성하세요.
print(f'행 합계: {}') # 주석을 지우고 중괄호 안에 코드를 작성하세요.
print(f'열 합계: {}') # 주석을 지우고 중괄호 안에 코드를 작성하세요.
```

Print

```
array([[ -3, -16,   8,   1, -13],
       [  0,  14,   1,  -1, -11]])
전체 합계: -20
행 합계: [ -3  -2   9   0 -24]
열 합계: [-23   3]
```

020 기본연산

- 다음은 랜덤한 데이터를 생성하고 데이터를 연산하는 코드입니다.
- 기본연산 함수를 사용하여 요구하는 값을 print하세요.
- 코드는 반드시 print문 안의 중괄호{}에 작성하세요.

In

```
data = np.array(np.random.randn(2, 5) * 10, dtype=int)
display(data)
print(f'평균: {}') # 주석을 지우고 중괄호 안에 코드를 작성하세요.
print(f'최대값: {}') # 주석을 지우고 중괄호 안에 코드를 작성하세요.
print(f'최소값: {}') # 주석을 지우고 중괄호 안에 코드를 작성하세요.
print(f'분산: {}') # 주석을 지우고 중괄호 안에 코드를 작성하세요.
```

Print

```
array([[  7, -12,   0, -11,  -2],
       [  4, -15,  14,   6,   6]])
평균: -0.3
최대값: 14
최소값: -15
분산: 82.61000000000001
```

2. pandas

021 pandas 모듈 불러오기 및 버전 확인

- pandas 모듈을 불러오고 버전을 확인해보세요.
- pandas 모듈을 pd 줄여 사용할 수 있도록 작성하세요.

022 Series 생성

- list 또는 tuple을 활용하여 보기의 출력값과 같은 Series를 생성하세요.

In

```
data = # 주석을 지우고 이곳에 코드를 작성하세요.
data
```

Out

```
0    5
1    3
2    1
3    2
4    4
dtype: int64
```

023 Series 생성

- numpy.array 활용하여 보기의 출력값과 같은 Series를 생성하세요.

In

```
data = # 주석을 지우고 이곳에 코드를 작성하세요.
data
```

Out

```
0    21
1    10
2    36
3    48
4    12
dtype: int32
```

024 Series 생성

- dict를 활용하여 보기의 출력값과 같은 Series를 생성하세요.

In

```
temp_dict = # 주석을 지우고 이곳에 코드를 작성하세요.
data = # 주석을 지우고 이곳에 코드를 작성하세요.
data
```

Out

```
일    1
이    2
삼    3
사    4
오    5
dtype: int64
```

025 데이터 확인

- 주어진 데이터의 타입을 예상해보고 확인하세요.

In

```
data1 = pd.Series(np.array([21, 10, 36, 48, 12]))
data2 = pd.Series(['A', 'B', 'C', 'D', 'E'])
print(data1.dtype)
print(data2.dtype)
data2.dtype
```

026 데이터 확인

- 다음의 출력문과 같이 주어진 Series의 index와 value를 확인하세요.

In

```
data = pd.Series({'일': 1, '이': 2, '삼': 3, '사': 4, '오': 5})
print() # 주석을 지우고 print 안에 코드를 작성하세요.
print() # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
Index(['일', '이', '삼', '사', '오'], dtype='object')
[1 2 3 4 5]
```

027 index 수정

- `data`의 Series 값을 다음의 출력값과 동일하도록 하는 코드를 작성하세요.

In

```
data = pd.Series({'일': 1, '이': 2, '삼': 3, '사': 4, '오': 5})
# 주석을 지우고 이곳에 코드를 작성하세요.
data
```

Out

```
A    1
B    2
C    3
D    4
E    5
dtype: int64
```

028 Series 인덱싱

- `data`의 Series중 출력값과 같이 영어~국사의 데이터에 접근하세요.

In

```
data = pd.Series({'국어': 77, '영어': 88, '수학': 81, '국사': 63, '과학': 97})
print() # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
국어    77
영어    88
수학    81
국사    63
dtype: int64
```

029 Series 인덱싱

- 다음의 출력값과 같이 `data` 변수에서 과학 데이터를 조회하세요.
- 반드시 `at`을 사용하세요.

In

```
data = pd.Series({'국어': 77, '영어': 88, '수학': 81, '국사': 63, '과학': 97})
print() # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
97
```

030 Series 연산

- 다음 세 Series 데이터를 연산하여 보기의 출력값과 같도록 코드를 작성하세요.

In

```
data1 = pd.Series([1, 2, 3, 4, 5])
data2 = pd.Series([10, 20, 30, 40, 50])

print(data1 * data2) # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
0     10
1     40
2     90
3    160
4    250
dtype: int64
```

031 NaN처리

- 다음의 출력값과 같이 변수 `data`의 NaN을 0으로 처리하는 코드를 작성하세요.
- 반드시 `fillna`를 사용하세요.

In

```
data = pd.Series([10, 20, 30, 40, 50]) * pd.Series([1, 2])
# 주석을 지우고 이곳에 코드를 작성하세요.
```

Out

```
0    10.0
1    40.0
2     0.0
3     0.0
4     0.0
dtype: float64
```

032 Series 데이터 확인

- 다음의 출력값과 같이 Series 데이터의 정보를 확인하는 코드를 작성하세요.

In

```
data = pd.Series([1, 2, 3, 4, 5]) * pd.Series([10, 20, 30, 40, 50])
print('mean:', ) # 주석을 지우고 print 안에 코드를 작성하세요.
print('std:', ) # 주석을 지우고 print 안에 코드를 작성하세요.
print('count:', ) # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
mean: 110.0
std: 96.69539802906858
count: 5
```

033 Series 데이터 확인

- 다음의 출력값과 같이 Series 데이터의 정보를 확인하는 코드를 작성하세요.

In

```
data = pd.Series([1, 2, 3, 4, 5]) * pd.Series([10, 20, 30, 40, 50])
print() # 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
count      5.000000
mean     110.000000
std       96.695398
min       10.000000
25%       40.000000
50%       90.000000
75%      160.000000
max      250.000000
dtype: float64
```

034 DataFrame 생성

- list를 활용해 보기의 출력값과 같은 DataFrame을 생성하세요.

In

```
df = # 주석을 지우고 이곳에 코드를 작성하세요.
df
```

Out

	0	1	2	3
0	0	7	8	1
1	1	4	7	3

035 DataFrame 생성

- dict를 활용해 보기의 출력값과 같은 DataFrame을 생성하세요.

In

```
df = # 주석을 지우고 이곳에 코드를 작성하세요.
df
```

Out

	학년	수학	영어
0	0	33	98
1	1	79	67
2	3	81	31
3	2	50	49

036 DataFrame 생성

- 아래의 출력값과 같이 index와 column이 출력 될 수 있도록 DataFrame을 생성하세요.

In

```
df = # 주석을 지우고 이곳에 DataFrame을 생성하세요.
df
```

Out

	남	여
김xx	1	0
이xx	0	1
박xx	0	1
최xx	1	0

037 index, column 변경

- 다음 데이터 프레임의 index와 column을 보기와 같이 변경하세요.

In

```
# 주석을 지우고 이곳에 코드를 작성하세요.
```

Out

	위도	경도
A	37.58945628	127.0168496
B	37.60935675	127.0530356
C	37.59167275	127.0122079
D	37.60935675	127.0530356
E	37.60963055	127.0530544

038 데이터 조회

- 다음 `df` 데이터의 1번 행의 값을 조회하세요.

In

```
df = pd.DataFrame({'구분':['일자리','창업','창업','창업','소공인'],
                   '시설명':['일자리 플러스 센터','벤처창업지원센터','1인 창조기업 비즈니스센터','시니어 기술창업센터','성북구 패션봉제지원센터(소공인특화지원센터)'],
                   '행정동':['삼선동','장위2동','성북동','장위2동','장위2동']})
# 주석을 지우고 이곳에 코드를 작성하세요.
```

Out

```
구분          창업
시설명    벤처창업지원센터
행정동        장위2동
Name: 1, dtype: object
```

039 데이터 조회

- 다음 `df` 데이터의 1~3번을 포함한 행의 값을 조회하세요.

In

```
df = pd.DataFrame({'구분':['일자리','창업','창업','창업','소공인'],
                   '시설명':['일자리 플러스 센터','벤처창업지원센터','1인 창조기업 비즈니스센터','시니어 기술창업센터','성북구 패션봉제지원센터(소공인특화지원센터)'],
                   '행정동':['삼선동','장위2동','성북동','장위2동','장위2동']})

print()# 주석을 지우고 print 안에 코드를 작성하세요.
```

Print

```
    구분                        시설명    행정동
1   창업                  벤처창업지원센터   장위2동
2   창업            1인 창조기업 비즈니스센터    성북동
3   창업                시니어 기술창업센터   장위2동
4  소공인  성북구 패션봉제지원센터(소공인특화지원센터)   장위2동
```

040 데이터 조회

- 다음의 출력값 처럼 `df` 의 데이터를 조회하세요.
- 반드시 인덱스와 컬럼을 동시에 사용하세요.

In

```
df = pd.DataFrame({'구분':['일자리','창업','창업','창업','소공인'],
                   '시설명':['일자리 플러스 센터','벤처창업지원센터','1인 창조기업 비즈니스센터','시니어 기술창업센터','성북구 패션봉제지원센터(소공인특화지원센터)'],
                   '행정동':['삼선동','장위2동','성북동','장위2동','장위2동']})

# 주석을 지우고 이곳에 코드를 작성하세요.
```

Out

	구분	행정동
2	창업	성북동
3	창업	장위2동
4	소공인	장위2동

041 데이터 조회

- 다음 출력값과 같이 위도가 37.6 보다 높은 데이터만 조회하세요.

In

```
df = pd.DataFrame({'위도':[37.58945628, 37.60935675, 37.59167275, 37.60935675, 37.60963055],
                   '경도':[127.0168496, 127.0530356, 127.0122079, 127.0530356, 127.0530544]})

# 주석을 지우고 이곳에 코드를 작성하세요.
```

Out

	위도	경도
1	37.609357	127.053036
3	37.609357	127.053036
4	37.609631	127.053054

042 데이터 합치기

- 다음의 두 데이터를 index 기준으로 합치세요

In

```
df1 = pd.DataFrame({'위도':[37.58945628, 37.60935675, 37.59167275, 37.60935675, 37.60963055],
                    '경도':[127.0168496, 127.0530356, 127.0122079, 127.0530356, 127.0530544]})
df2 = pd.DataFrame({'구분':['일자리','창업','창업','창업','소공인'],
                    '시설명':['일자리 플러스 센터','벤처창업지원센터','1인 창조기업 비즈니스센터','시니어 기술창업센터','성북구 패션봉제지원센터(소공인특화지원센터)'],
                    '행정동':['삼선동','장위2동','성북동','장위2동','장위2동']})

# 주석을 지우고 이곳에 코드를 작성하세요.
```

Out

	위도	경도	구분	시설명	행정동
0	37.589456	127.016850	일자리	일자리 플러스 센터	삼선동
1	37.609357	127.053036	창업	벤처창업지원센터	장위2동
2	37.591673	127.012208	창업	1인 창조기업 비즈니스센터	성북동
3	37.609357	127.053036	창업	시니어 기술창업센터	장위2동
4	37.609631	127.053054	소공인	성북구 패션봉제지원센터(소공인특화지원센터)	장위2동

043 데이터프레임 연산

- 두 데이터프레임의 연산 결과를 예상해보고 확인하세요.

In

```
df1 = pd.DataFrame([[12, 6, 3], [4, 12, 9]])
df2 = pd.DataFrame([[3, 6], [1, 6]])

df1 / df2
```

044 csv 불러오기

- 다음의 링크로 들어가 공공데이터를 다운받으세요.[https://www.data.go.kr/data/15040314/fileData.do#/tab-layer-file]
- 다운 받은 데이터를 현재 작업경로로 옮기세요.
- csv 데이터를 불러와 df 변수에 저장하세요.

In

```
df = # 주석을 지우고 이곳에 코드를 작성하세요.
df.head()
```

045 DataFrame 통계

- 다음의 링크로 들어가 공공데이터를 다운받으세요.[공공데이터]
- 다운 받은 데이터를 현재 작업경로로 옮기세요.
- `경기도 여주시_보호수_20220803.csv` 파일을 불러와 `df` 변수에 저장하세요.
- `df` 의 통계 요약을 출력하세요.

In

```
df = pd.read_csv('경기도 여주시_보호수_20220803.csv', encoding='cp949')
# 주석을 지우고 이곳에 코드를 작성하세요.
```

046 DataFrame 통계

- `경기도 여주시_보호수_20220803.csv` 파일을 불러와 `df` 변수에 저장하세요.
- `df` 의 나무나이, 나무높이, 가슴높이둘레, 나무갓지름의 상관계수를 출력하세요.

In

```
df = pd.read_csv('경기도 여주시_보호수_20220803.csv', encoding='cp949')
# 주석을 지우고 이곳에 코드를 작성하세요.
```

047 DataFrame 통계

- `경기도 여주시_보호수_20220803.csv` 파일을 불러와 `df` 변수에 저장하세요.
- `df` 의 그루수, 나무나이 컬럼의 합을 출력하세요

In

```
df = pd.read_csv('경기도 여주시_보호수_20220803.csv', encoding='cp949')
# 주석을 지우고 이곳에 코드를 작성하세요.
```

048 DataFrame 통계

- `경기도 여주시_보호수_20220803.csv` 파일을 불러와 `df` 변수에 저장하세요.
- `df` 를 나무종류 순서로 정렬하세요.
- 내림차순으로 정렬하세요.
- 변경된 데이터를 원본 데이터에 덮어 쓸 수 있는 파라미터를 사용하세요.

In

```
df = pd.read_csv('경기도 여주시_보호수_20220803.csv', encoding='cp949')
# 주석을 지우고 이곳에 코드를 작성하세요.
df.head()
```

049 DataFrame 통계

- `경기도 여주시_보호수_20220803.csv` 파일을 불러와 `df` 변수에 저장하세요.
- `df` 를 컬럼의 이름 순서로 정렬하세요.

In

```
df = pd.read_csv('경기도 여주시_보호수_20220803.csv', encoding='cp949')
# 주석을 지우고 이곳에 코드를 작성하세요.
```

050 결측치 처리

- `경기도 여주시_보호수_20220803.csv` 파일을 불러와 `df` 변수에 저장하세요.
- `df` 의 컬럼 중 나무갓지름의 빈 데이터가 포함된 곳에 나무갓지름의 평균값을 채워넣으세요.

In

```
# 주석을 지우고 이곳에 코드를 작성하세요.
```

051 데이터 조회 및 변경

- `경기도 여주시_보호수_20220803.csv` 파일을 불러와 `df` 변수에 저장하세요.
- `df` 의 컬럼 중 학명의 빈 데이터가 포함된 알맞는 데이터를 채워 넣으세요.

- `느티나무` 의 학명은 `Zelkova serrata (Thunb.) Makino` 입니다.
- `물푸레나무` 의 학명은 `Fraxinus rhynchophylla` 입니다.
- `아까시나무` 의 학명은 `Robinia pseudoacacia` 입니다.
- `오리나무` 의 학명은 `Alnus japonica (Thunb.) Steudel` 입니다.
- `은행나무` 의 학명은 `Ginkgo biloba L.` 입니다.
- `향나무` 의 학명은 `Juniperus chinensis` 입니다.

In

```
df = pd.read_csv('경기도 여주시_보호수_20220803.csv', encoding='cp949')
# 주석을 지우고 이곳에 코드를 작성하세요.
df
```