





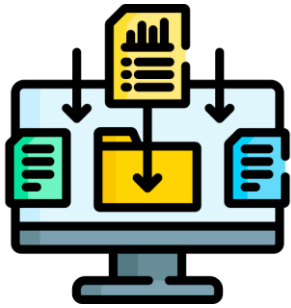
■ Contents

- 데이터
- Pandas
- Series
- DataFrame
- Index & slice
- 함수

데이터

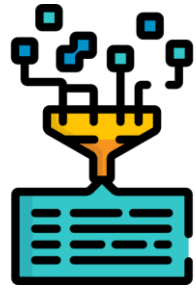


■ 데이터 분석



데이터 수집

다양한 채널을 통해 문서, 그림 등의 데이터를 수집



전처리 및 가공

지저분한 원시 데이터를 변환, 처리, 검사 등 하여 전처리



시각화

가공된 데이터를 그래프, 도표 등을 이용하여 시각화



인사이트 도출

시각화 된 그래프, 도표를 통하여 분석 결과 도출

데이터



- 데이터 자료구조

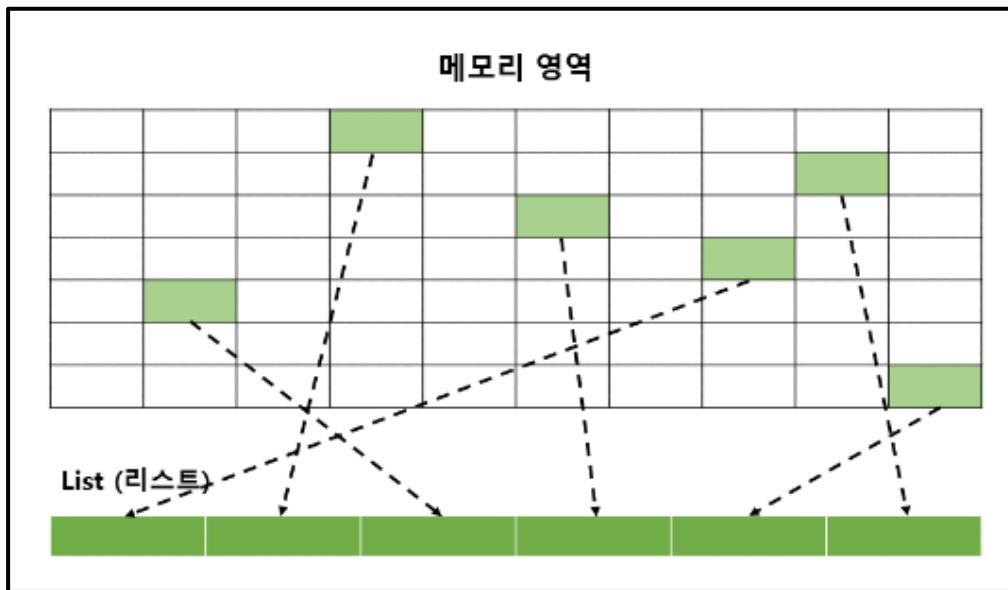
자료 구조	특 징	파이썬 코드
리스트	순서가 있고 수정 가능	<code>ex_list = [1,2,3]</code>
튜플	순서가 있고 수정 불가능	<code>ex_tuple = (1,2,3)</code>
딕셔너리	연관된 데이터 {key:value}	<code>ex_dic = {'coffee':5000}</code>
NdArray	반복문 없이 데이터 배열 처리 및 다양한 기능 제공	<code>ex_arr = np.array([1,2,3])</code>



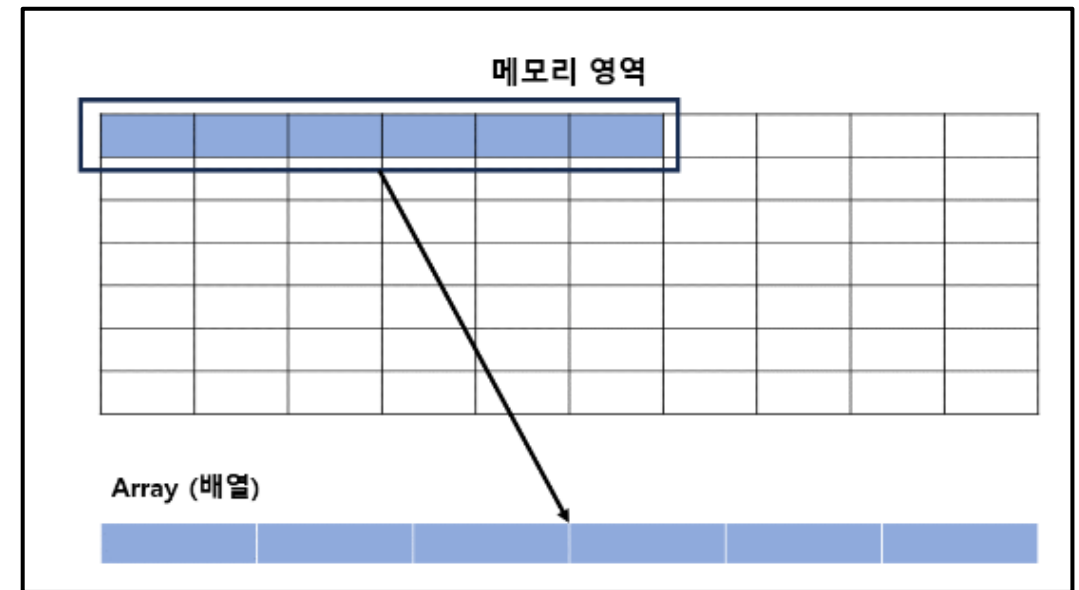
데이터



■ List(리스트) & array(배열)



List는 크기가 정해져 있지 않으며
메모리에 무작위로 할당되어 있음



Array(배열)은 크기가 정해져있으며
메모리에 연속적으로 할당되어 있음



데이터



- List(리스트) & array(배열)

	파이썬	연산	속도	사용예
List (리스트)	○	기본적인 연산지원	삽입/삭제:느림 데이터조회:빠름	웹 개발, 데이터 구조
Array (배열)	x	다양한 연산지원	삽입/삭제:빠름 데이터조회:느림	데이터분석, 과학 계 산



데이터



- Numpy

	A	B
1	1	10
2	2	20
3	3	30
4	4	40

A=[1,2,3,4]의 데이터에
각 항목에 x10하여
B=[10,20,30,40]
만들 때,

Python

```
data = [1, 2, 3, 4]
result = []
for i in data:
    result.append(i * 10)

print(result)
```

np.array

```
1 import numpy as np
2
3 data = [1, 2, 3, 4]
4 arr = np.array(data)
5 arr10 = arr * 10
6 print(arr10)
7
```



Pandas



■ Pandas

- 데이터 처리와 분석을 위한 파이썬 라이브러리
- 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있음
- 대용량의 데이터들을 처리하는데 매우 편리

	점수
재욱	90
철수	50
제니	80
지수	85

Series(1차원)

	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
제니	80	60	70	88
지수	85	90	99	70

DataFrame(2차원)



Pandas



■ Pandas 기본 자료구조

자료구조	특징
Series	NumPy를 기반으로 만들어진 1차원 데이터를 위한 자료구조
DataFrame	NumPy를 기반으로 만들어진 2차원 데이터를 위한 자료구조

	점수
재욱	90
철수	50
제니	80
지수	85

Series(1차원)

	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
제니	80	60	70	88
지수	85	90	99	70

DataFrame(2차원)



Pandas



- Data 구조

	점수
재욱	90
철수	50
세니	80
지수	85

index

Series(1차원)

	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
세니	80	60	70	88
지수	85	90	99	70

index

DataFrame(2차원)



Pandas



- Data 구조

	점수
재욱	90
철수	50
세니	80
지수	85

index Values

Series(1차원)

	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
세니	80	60	70	88
지수	85	90	99	70

index Values

DataFrame(2차원)



Pandas



- Data 구조

	점수
재욱	90
철수	50
세니	80
지수	85

index Values

Series(1차원)

	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
세니	80	60	70	88
지수	85	90	99	70

index Values

DataFrame(2차원)



Pandas



- Data 구조

	점수
재욱	90
철수	50
제니	80
지수	85

Series(1차원)

```
score_list = [90, 50, 80, 85]
```

	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
제니	80	60	70	88
지수	85	90	99	70

DataFrame(2차원)

```
score_dic = {'국어':[90,50,80,85], '수학':[40,50,60,90], ...  
....}
```

즉, 여러 개의 series (columns) 모아 하나의 큰 dataframe 된다.



Pandas



■ 간단 예제

```
[1]: import pandas as pd

•[6]: df1 = pd.DataFrame([[3,2,5],[10,0,2],[6,5,3]],
                        columns=["사과", "자두", "포도"], index=["이성계", "김유신", "이순신"])

[7]: s1 = df["사과"]
```

df1			
	사과	자두	포도
이성계	3	2	5
김유신	10	0	2
이순신	6	5	3

Dataframe

s1	
이성계	3
김유신	10
이순신	6
Name: 사과, dtype: int64	

Series



Pandas



■ 간단 예제

	사과	자두	포도
이성계	3	2	5
김유신	10	0	2
이순신	6	5	3

- `df1.info()`
- `df1.columns`
- `df1.index`
- `df1.values`
- `df1.sum()`
- `df1.mean()`



Pandas



■ 간단 예제

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 3 entries, 이성계 to 이순신  
Data columns (total 3 columns):  
#   Column  Non-Null Count  Dtype  
---  ---  
0   사과      3 non-null    int64  
1   자두      3 non-null    int64  
2   포도      3 non-null    int64  
dtypes: int64(3)  
memory usage: 96.0+ bytes
```

```
df1.columns
```

```
Index(['사과', '자두', '포도'], dtype='object')
```

```
df1.index
```

```
Index(['이성계', '김유신', '이순신'], dtype='object')
```

```
df1.values
```

```
array([[ 3,  2,  5],  
       [10,  0,  2],  
       [ 6,  5,  3]], dtype=int64)
```



Series



- Series 인덱싱

1) iloc 활용

2) loc 활용

3) [] 활용

```
1  import pandas as pd
2
3  data = [100, 200, 300]
4  index = ["월", "화", "수"]
5  s = pd.Series(data, index)
6
7  # iloc 활용
8  print(s.iloc[2])
9  # loc 활용
10 print(s.loc["월"])
11 # [ ] 활용
12 print(s["월"])
13 |
```



Series



- Series 슬라이싱

1) iloc 활용

2) loc 활용

3) [] 활용

```
import pandas as pd

data = [100, 200, 300]
index = ["월", "화", "수"]
s = pd.Series(data, index)

#iloc 활용
print(s.iloc[0:2])
#loc 활용
print(s.loc["월":"화"])
#[ ] 활용
print(s[0:2])
```

```
월      100
화      200
dtype: int64

월      100
화      200
dtype: int64

월      100
화      200
dtype: int64
```



Series



- Series 멀티 인덱싱

1) iloc 활용

2) loc 활용

3) [] 활용

```
import pandas as pd

data = [100, 200, 300]
index = ["월", "화", "수"]
s = pd.Series(data, index)

iloc_target = [0, 2]
loc_target = ["월", "수"]

#iloc 활용
print(s.iloc[iloc_target])
print(s.iloc[[0, 2]])

#loc 활용
print(s.loc[loc_target])
print(s.loc[["월", "수"]])
```

```
월    100
수    300
dtype: int64
월    100
수    300
dtype: int64
월    100
수    300
dtype: int64
월    100
수    300
dtype: int64
```



DataFrame



- DataFrame 인덱싱 & 슬라이싱

- 1) iloc 활용

- 2) loc 활용

```
import pandas as pd

data = {"국어": [90, 50, 80, 85], "수학": [40, 50, 60, 90],
        "영어": [20, 70, 70, 99], "코딩": [90, 50, 88, 70]}

index = ["재욱", "철수", "제니", "지수"]
df = pd.DataFrame(data, index)

#iloc 활용
print(df.iloc[0:2])
#loc 활용
print(df.loc["재욱": "철수"])
```

✓ 0.0s

	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50



DataFrame



- DataFrame 멀티 인덱싱 & 슬라이싱

- 1) iloc 활용

- 2) loc 활용

```
import pandas as pd

data = {"국어": [90, 50, 80, 85], "수학": [40, 50, 60, 90],
        "영어": [20, 70, 70, 99], "코딩": [90, 50, 88, 70]}

index = ["재욱", "철수", "제니", "지수"]
df = pd.DataFrame(data, index)

#iloc 활용
print(df.iloc[0:2, 1:3])
#loc 활용
print(df.loc["재욱": "철수", "수학": "영어"])
```

	수학	영어
재욱	40	20
철수	50	70

	수학	영어
재욱	40	20
철수	50	70



DataFrame



- DataFrame 멀티 인덱싱 & 슬라이싱

- 1) iloc 활용

- 2) loc 활용

```
import pandas as pd

data = {"국어": [90, 50, 80, 85], "수학": [40, 50, 60, 90],
        |      "영어": [20, 70, 70, 99], "코딩": [90, 50, 88, 70]}

index = ["재욱", "철수", "제니", "지수"]
df = pd.DataFrame(data, index)

#iloc 활용
print(df.iloc[0:2, 1:3])
#loc 활용
print(df.loc["재욱": "철수", "수학": "영어"])
```

	수학	영어
재욱	40	20
철수	50	70

	수학	영어
재욱	40	20
철수	50	70



DataFrame



- DataFrame 데이터 추가, 삭제

1) 추가

```
import pandas as pd

data = {"국어": [90, 50, 80, 85], "수학": [40, 50, 60, 90],
        "영어": [20, 70, 70, 99], "코딩": [90, 50, 88, 70]}

index = ["재욱", "철수", "제니", "지수"]
df = pd.DataFrame(data, index)

#추가
df["sum"] = df["국어"] + df["수학"] + df["영어"] + df["코딩"]
df
```

	국어	수학	영어	코딩	sum
재욱	90	40	20	90	240
철수	50	50	70	50	220
제니	80	60	70	88	298
지수	85	90	99	70	344



DataFrame



- DataFrame 데이터 추가, 삭제

1) 삭제

```
import pandas as pd

data = {"국어": [90, 50, 80, 85], "수학": [40, 50, 60, 90],
        "영어": [20, 70, 70, 99], "코딩": [90, 50, 88, 70]}

index = ["재욱", "철수", "제니", "지수"]
df = pd.DataFrame(data, index)

#추가
df["sum"] = df["국어"] + df["수학"] + df["영어"] + df["코딩"]
#삭제
df = df.drop("sum", axis=1)
```













	국어	수학	영어	코딩
재욱	90	40	20	90
철수	50	50	70	50
제니	80	60	70	88
지수	85	90	99	70



Index & slice



- 1차원 인덱싱 & 슬라이싱

0	1	2	3	
				[0:2]
				[1:3]
				[1:0]



Index & slice



- 2차원 인덱싱 & 슬라이싱

0:2

0:2

[0:2, 0:3]





- DataFrame 함수

1. `.head()` & `.tail()`
2. `.shape`, `index`, `columns`
3. `.copy()`
4. `.value_counts()`
5. `.fillna()`
6. `.drop()`
7. `.rename()`
8. `.iterrows()`



감사합니다

