학습데이터처리

Pandas 기초

학습 목차

- 교재의 다양한 예제를 통해 데이터를 분석하기에 앞써,
  - 1. 판단스 입문(기초)
  - 2. 데이터 입출력 (CSV, Excel, Json, etc.)
  - 3. 데이터 분석/시각화
- 위 3가지에 대해서 기초 수업 진행 예정

판다스 입문

- 데이터과학자가 판다스를 배우는 이유

빅데이터의 시대. 데이터 과학이라는 새로운 영역의 출현.

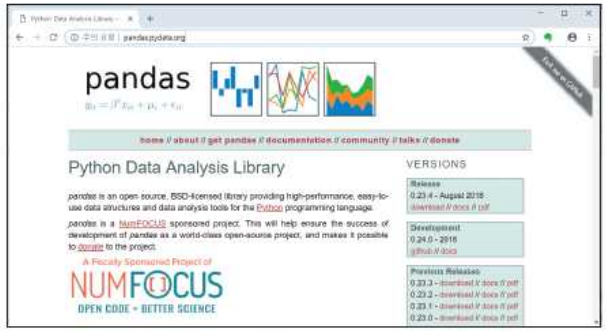
- 클라우드 컴퓨팅의 확산. 빅데이터 저장, 분석에 필요한 컴퓨팅 자원이 매우 저렴해짐.
- 컴퓨팅 파워의 대중화는 최적의 학습환경과 연구 인프라를 제공.

데이터과학은 데이터를 연구하는 분야이고, 데이터 자체가 가장 중요한 자원

- 데이터 분석 업무의 **80~90%는 데이터를 수집하고 정리하는 일**이 차지.
- 나머지 10~20%는 알고리즘을 선택하고, 모델링 결과를 분석하여 데이터로부터 유용한 정보(information)을 뽑아내는 분석 프로세스의 몫.
- 데이터과학자가 하는 가장 중요한 일이 **데이터를 수집하고 분석이 가능한 형태로 정리**하는 것.

판다스는 데이터를 수집하고 정리하는데 최적화된 도구.

- 가장 배우기 쉬운 프로그래밍 언어, 파이썬(Python) 기반.
- 오픈소스(open source)로 무료로 이용 가능.



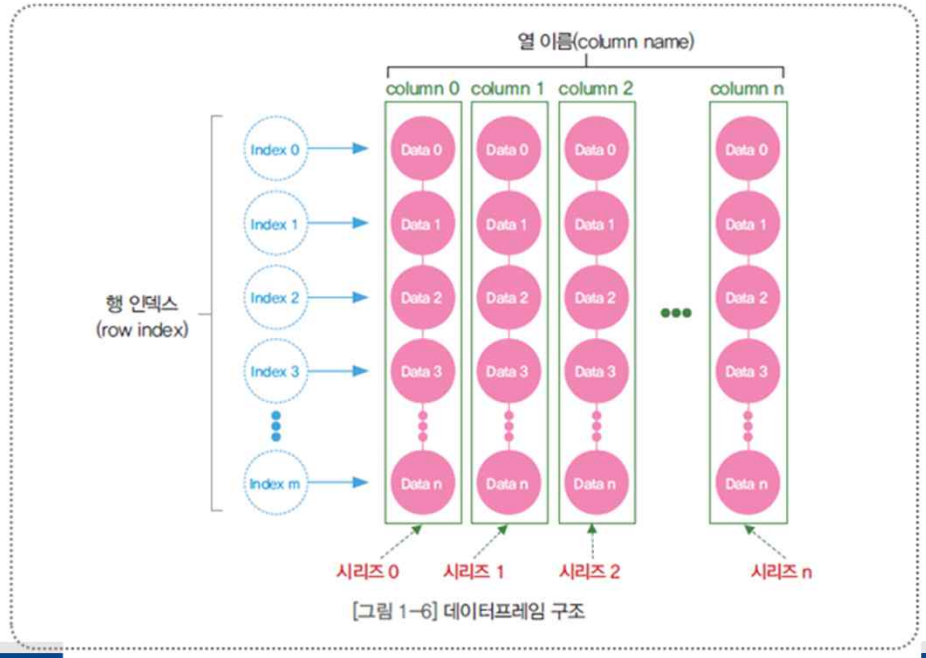
[그림 1-1] 판다스 공식 홈페이지(http://pandas.pydata.org/)

판다스 입문

- 판다스 자료구조

  - **시리즈(Series)와 데이터프레임(DataFrame)이라는 구조화된 데이터 형식**을 제공
  - 서로 다른 종류의 데이터를 한곳에 담는 컨테이너
  - 시리즈는 1차원 배열이고, 데이터프레임이 2차원 배열이라는 점에서 차이가 있음
  - 특히, **행과 열로 이루어진 2차원 구조의 데이터프레임은 데이터 분석 실무에서 자주 사용**

[그림 1-2] 시리즈 구조



[그림 1-6] 데이터프레임 구조

판다스 입문

- 판다스 자료구조 (시리즈)

  - 시리즈 만들기
    1) 딕셔너리와 시리즈의 구조가 비슷하기 때문에, **딕셔너리를 시리즈로 변환하는 방법을 많이 사용**
    2) 판다스 내장 함수인 **Series()를 이용**하고, 딕셔너리를 함수의 매개변수(인자)로 전달
    3) 딕셔너리의 키(k)는 시리즈의 인덱스에 대응하고, 딕셔너리의 각 키에 매칭되는 값(v)이 시리즈의 데이터 값(원소)로 변환.

딕셔너리 → 시리즈 변환: `pandas.Series( 딕셔너리 )`

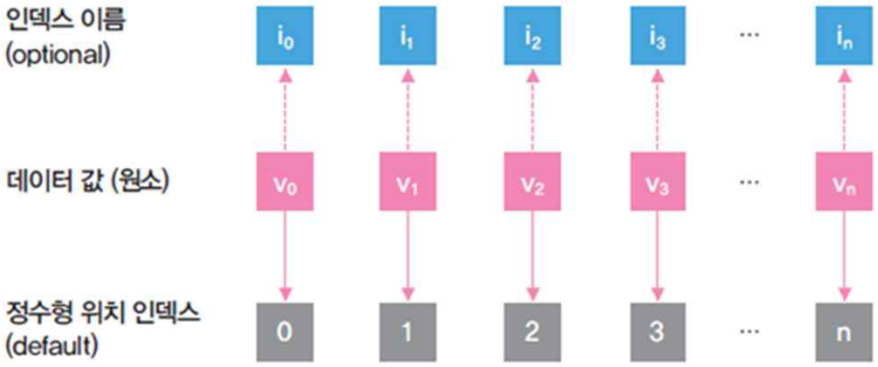
[그림 1-3] 딕셔너리 → 시리즈 변환

```python
import pandas as pd

# k:v 구조를 갖는 딕셔너리를 만들고, 변수 dict_data에 저장
dict_data = {'a': 1, 'b': 2, 'c': 3}

# 판다스 Series() 함수로 딕셔너리(dict_data)를 시리즈로 변환. 변수 sr에 저장
sr = pd.Series(dict_data)

# 변수 sr의 자료형 출력
print(type(sr))
print('\n')

# 변수 sr에 저장되어 있는 시리즈 객체를 출력
print(sr)
```

판다스 입문

- 판다스 자료구조 (시리즈)

  - 인덱스 구조
    1) 인덱스는 자기와 짝을 이루는 원소(데이터 값)의 순서와 주소를 저장
    2) 인덱스를 잘 활용하면 데이터 값의 탐색, 정렬, 선택, 결합 등 데이터 조작
    3) 인덱스의 종류 (2가지)
       - 정수형 위치 인덱스(integer position): Default
       - 인덱스 이름(index name) 또는 인덱스 라벨(index label)



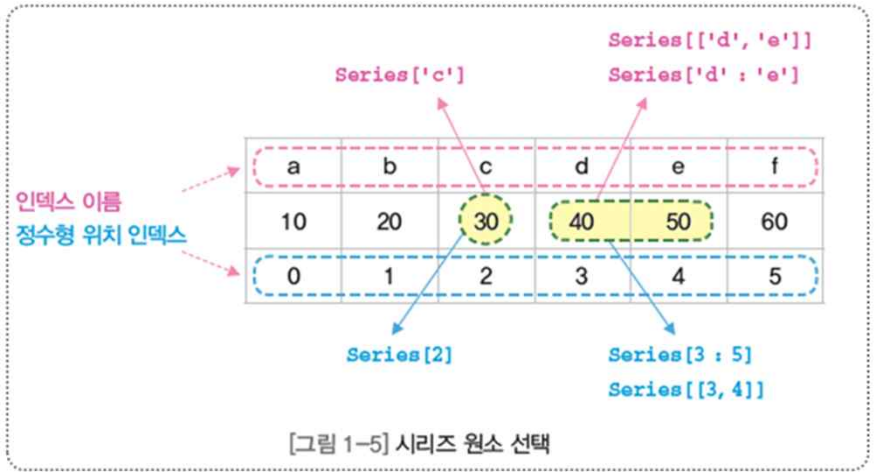
[그림 1-4] 시리즈 인덱스의 유형

```python
import pandas as pd

# 리스트를 시리즈로 변환하여 변수 sr에 저장
list_data = ['2019-01-02', 3.14, 'ABC', 100, True]
sr = pd.Series(list_data)
print(sr)
print('\n')

# 인덱스 배열은 변수 idx에 저장. 데이터 값 배열은 변수 val에 저장
idx = sr.index
val = sr.values
print(idx)
print('\n')
print(val)
```

판다스 입문

- 판다스 자료구조 (시리즈)

  - 원소 선택
    1) 인덱스를 이용하여 시리즈의 원소 선택
    2) 하나의 원소를 선택하거나, 여러 원소를 한꺼번에 선택 가능 e.g.) sr[0], sr[[1, 2]]
    3) 인덱스의 범위를 지정하여 여러 개의 원소 선택 가능 (슬라이싱, e.g.) sr[0:3]
    4) 인덱스의 유형에 따라 사용법이 조금 다름
       - 정수형 인덱스: [ ] 안에 숫자 입력 (Indexing)
       - 인덱스 이름(라벨): [ ] 안에 이름을 입력



[그림 1-5] 시리즈 원소 선택

판다스 입문

- 판다스 자료구조 (시리즈)

  - 원소 선택
    1) 인덱스를 이용하여 시리즈의 원소 선택
    2) 하나의 원소를 선택하거나, 여러 원소를 한꺼번에 선택 가능 e.g.) sr[0], sr[[1, 2]]
    3) 인덱스의 범위를 지정하여 여러 개의 원소 선택 가능 (슬라이싱, e.g.) sr[0:3]

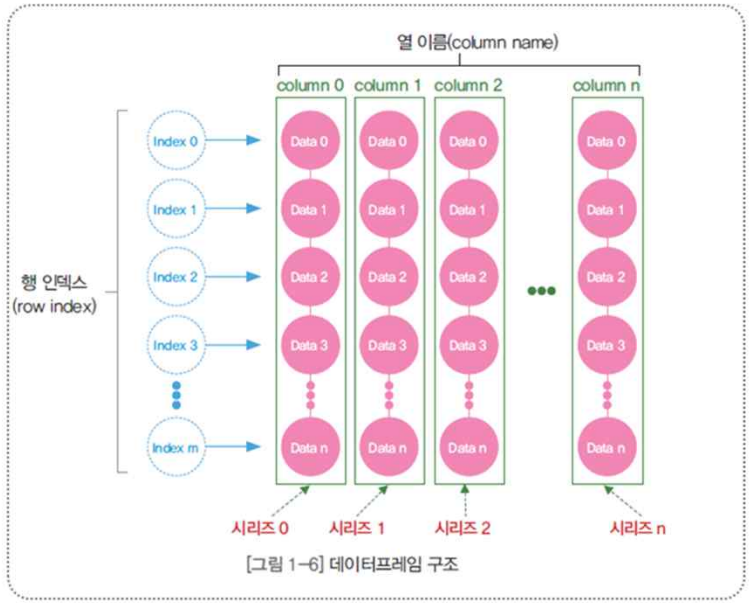
```
import pandas as pd

# 튜플을 시리즈로 변환(index 옵션에 인덱스 이름을 지정)
tup_data = ('이근혁', '19xx-xx-xx', '남', True)
sr = pd.Series(tup_data, index=['이름', '생년월일', '성별', '학생여부'])
print(sr)
print('\n')

# 원소를 1개 선택
print(sr[0])        # sr의 1 번째 원소를 선택 (정수형 위치 인덱스를 활용)
print(sr['이름'])  # '이름' 라벨을 가진 원소를 선택 (인덱스 이름을 활용)
print('\n')

# 여러 개의 원소를 선택 (인덱스 리스트 활용)
print(sr[[1, 2]])
print('\n')
print(sr[['생년월일', '성별']])
print('\n')

# 여러 개의 원소를 선택 (인덱스 범위 지정, 1~1: 따라서 하나만)
print(sr[1:1])
print('\n')
print(sr['생년월일':'성별'])
```

판다스 입문

- 판다스 자료구조 (데이터프레임)

  - 개요

    1) 2차원 배열 (R의 데이터프레임에서 유래)
    2) 데이터프레임의 열은 시리즈 객체. 시리즈를 열벡터(vector)라고 하면, 데이터프레임은 여러 개의 열벡터들이 같은 행 인덱스를 기준으로 줄지어 결합된 **행렬(matrix)**
    3) 데이터프레임은 행과 열을 나타내기 위해 두 가지 종류의 주소를 사용.
       - 행 인덱스(row index)와 열 이름(column name 또는 column label)으로 구분



[그림 1-6] 데이터프레임 구조

- 데이터프레임의 각 열은 공통의 속성을 갖는 일련의 데이터를 나타냄.
- 각 행은 개별 관측대상에 대한 다양한 속성 데이터들의 모음인 레코드(record).

열

종목 코드	회사 이름	액면가	총 주식수
005930	삼성전자	100원	5,970백만 주
017670	SK텔레콤	500원	81백만 주
005380	현대자동차	5000원	214백만 주

행

[표 1-1] 주식 종목 리스트

판다스 입문

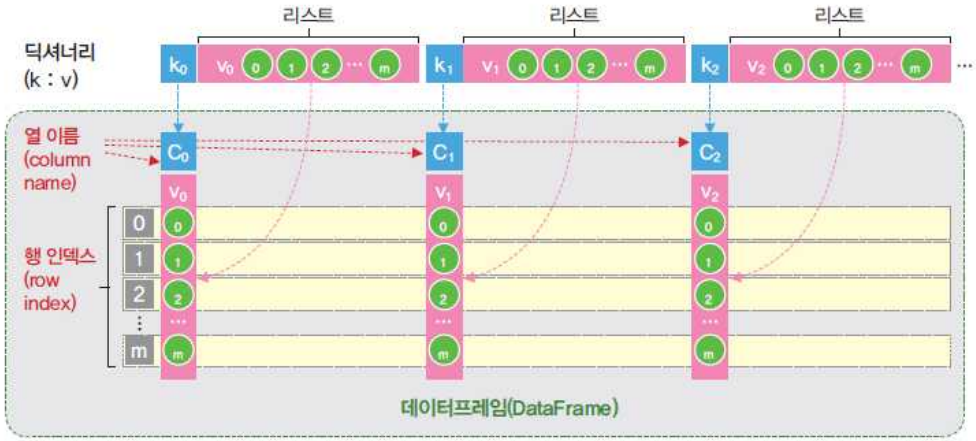
- 판다스 자료구조 (데이터프레임)

  - 데이터프레임 만들기

    1) 같은 길이(원소의 개수가 동일한)의 배열 여러 개가 필요. 데이터프레임은 여러 개의 시리즈(열, column)를 모아 놓은 집합
    2) 판다스 DataFrame() 함수를 사용. 여러 개의 리스트를 원소로 갖는 딕셔너리를 함수에 전달하는 방식을 주로 활용

    딕셔너리 → 데이터프레임 변환: `pandas.DataFrame( 딕셔너리 객체 )`

    3) 딕셔너리의 값(v)에 해당하는 각 리스트가 시리즈로 변환되어 데이터프레임의 각 열이 됨
    4) 딕셔너리의 키(k)는 각 시리즈의 이름으로 변환되어, 최종적으로 데이터프레임의 열 이름이 됨



[그림 1-7] 딕셔너리 → 데이터프레임 변환

```python
import pandas as pd

# 열이름을 key로 하고, 리스트를 value로 갖는 딕셔너리 정의(2차원 배열)
dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9],
             'c3':[10,11,12], 'c4':[13,14,15]}

# 판다스 DataFrame() 함수로 딕셔너리를 데이터프레임으로 변환. 변수 df에 저장.
df = pd.DataFrame(dict_data)

# df의 자료형 출력
print(type(df))
print('\n')
# 변수 df에 저장되어 있는 데이터프레임 객체를 출력
print(df)
```

판다스 입문

- 판다스 자료구조 (데이터프레임)

  - 행 인덱스/열 이름 설정: 데이터 프레임의 행 인덱스와 열 이름을 사용자가 지정 가능

    1) 데이터프레임을 만들때 설정

```
행 인덱스/열 이름 설정: pandas.DataFrame( 2차원 배열,
                                        index=행 인덱스 배열,
                                        columns=열 이름 배열 )
```

    2) 속성을 지정하여 변경: df.index, df.column으로 행 인덱스/열 이름 배열에 접근 가능(수정 가능)

```
• 행 인덱스 변경: DataFrame 객체.index = 새로운 행 인덱스 배열
• 열 이름 변경: DataFrame 객체.columns = 새로운 열 이름 배열
```

    3) rename 함수 사용: 새로운 데이터프레임 객체를 리턴, **원본 객체를 변경하려면, inplace=True 옵션을 사용**

```
• 행 인덱스 변경: DataFrame 객체.rename(index={기존 인덱스:새 인덱스, … })
• 열 이름 변경: DataFrame 객체.rename(columns={기존 이름:새 이름, … })
```

판다스 입문

- 판다스 자료구조 (데이터프레임)
  - 행 인덱스/열 이름 설정

```python
import pandas as pd

# 행 인덱스/열 이름 지정하여, 데이터프레임 만들기
df = pd.DataFrame([[23, '남', '대구'], [24, '여', '부산']],
                  index=['이근혁', '최윤정'],
                  columns=['나이', '성별', '지역'])

df_cpy = df[:]
print(df_cpy)

# 행 인덱스, 열 이름 확인하기
print(df)            #데이터프레임
print(df.index)      #행 인덱스
print(df.columns)    #열 이름

# 행 인덱스, 열 이름 변경하기
df.index=['학생1', '학생2']
df.columns=['연령', '남녀', '주소지']
print(df)            #데이터프레임

# df_cpy.rename(index={'이근혁': '학생1', '최윤정': '학생2'}, inplace=True)
df_cpy = df_cpy.rename(index={'이근혁': '학생1', '최윤정': '학생2'})
print(df_cpy)
```

판다스 입문

- 판다스 자료구조 (데이터프레임)

  - 행/열 삭제
    1) drop() 메소드에 행을 삭제할 때는 축(axis) 옵션으로 axis=0을 입력 (Default: axis = 0)
    2) axis = 1 입력 시: 열 삭제
    3) 여러 개의 행 또는 열을 삭제하려면, 리스트 형태로 입력
    4) drop() 메소드는 기존 객체를 변경하지 않고, 새로운 객체를 반환, **원본 객체를 변경하려면, inplace=True 옵션을 사용**

- 행 삭제: `DataFrame 객체.drop(행 인덱스 또는 배열, axis=0)`
- 열 삭제: `DataFrame 객체.drop(열 이름 또는 배열, axis=1)`

```python
import pandas as pd

...
exam_data = {'인공지능기초': [90, 80, 70], '자료구조': [98, 89, 95],
             '자바': [85, 95, 100], '기계학습프로그래밍': [100, 90, 90]}

df = pd.DataFrame(exam_data, index=['노영하', '김근형', '정경임'])
print(df)

# # 데이터프레임 df를 복제하여 변수 df2에 저장. df2의 1개 행(row)을 삭제
df2 = df[:]
# df2 = df.copy()
df2.drop('정경임', axis=0, inplace=True) # axis=0 default
print(df2)
#
# # 데이터프레임 df를 복제하여 변수 df3에 저장. df3의 2개 행(row)을 삭제
```

판다스 입문

- 판다스 자료구조 (데이터프레임)

  - 행 선택
    1) loc, iloc 인덱서를 사용
       - loc: 인덱스 이름을 기준으로 행을 선택할 때
       - iloc: 정수형 위치 인덱스를 사용할 때

구분	loc	iloc
탐색 대상	인덱스 이름(index label)	정수형 위치 인덱스(integer position)
범위 지정	가능(범위의 끝 포함) 예) `['a':'c']` → `'a',' b', 'c'`	가능(범위의 끝 제외) 예) `[3:7]` → `3, 4, 5, 6` (* 7 제외)

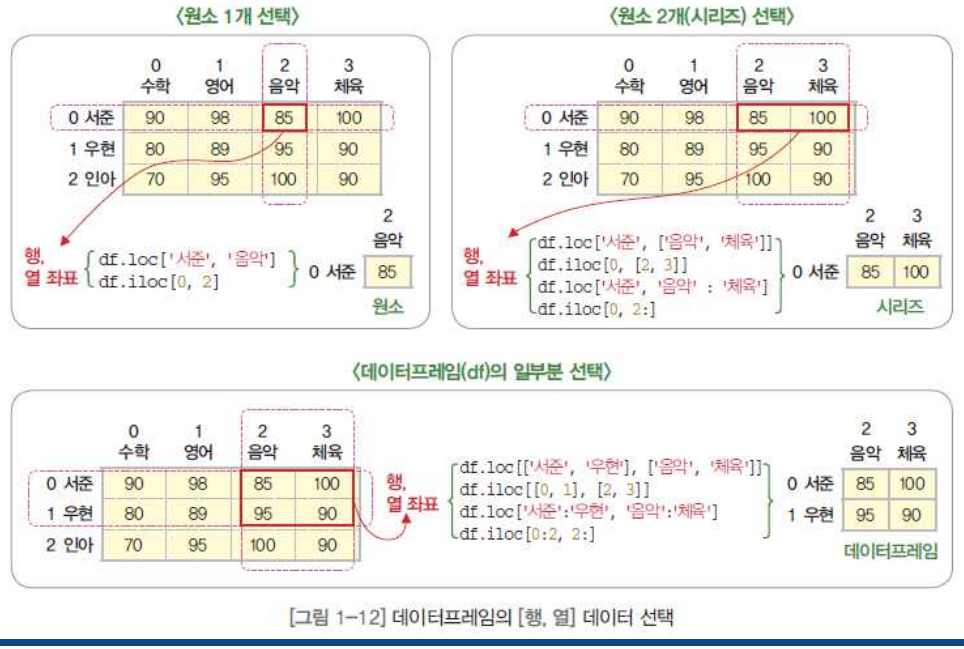
[표 1-2] loc과 iloc

  - 열 선택

열 1개 선택(시리즈 생성): `DataFrame 객체["열 이름"]` 또는 `DataFrame 객체.열 이름`

열 n개 선택(데이터프레임 생성): `DataFrame 객체[ [ 열1, 열2 , … , 열n ] ]`

  - 원소 선택
    - 인덱스 이름: DataFrame 객체.loc[행 인덱스, 열 이름]
    - 정수 위치 인덱스: DataFrame 객체.iloc[행번호, 열번호]



[그림 1-12] 데이터프레임의 [행, 열] 데이터 선택

판다스 입문

- 판다스 자료구조 (데이터프레임)
  - 행 선택, 열 선택, 원소 선택

```
import pandas as pd

# DataFrame() 함수로 데이터프레임 변환. 변수 df에 저장
exam_data = {'이름' : ['노영하', '김근형', '정경임'],
             'AI기초' : [90, 80, 70],
             '자료구조' : [98, 89, 95],
             '영어' : [85, 95, 100],
             '웹프로그래밍' : [100, 90, 90]}
df = pd.DataFrame(exam_data)
print(df)
print(type(df))

# '자료구조' 점수 데이터만 선택. 변수 datastructure에 저장
datastructure = df['자료구조']
print(datastructure)
print(type(datastructure)) # Series

# '영어' 점수 데이터만 선택. 변수 english에 저장
english = df.영어
print(english)
```

Thank you

Q&A

www.kopo.ac.kr
jsshin7@kopo.ac.kr

