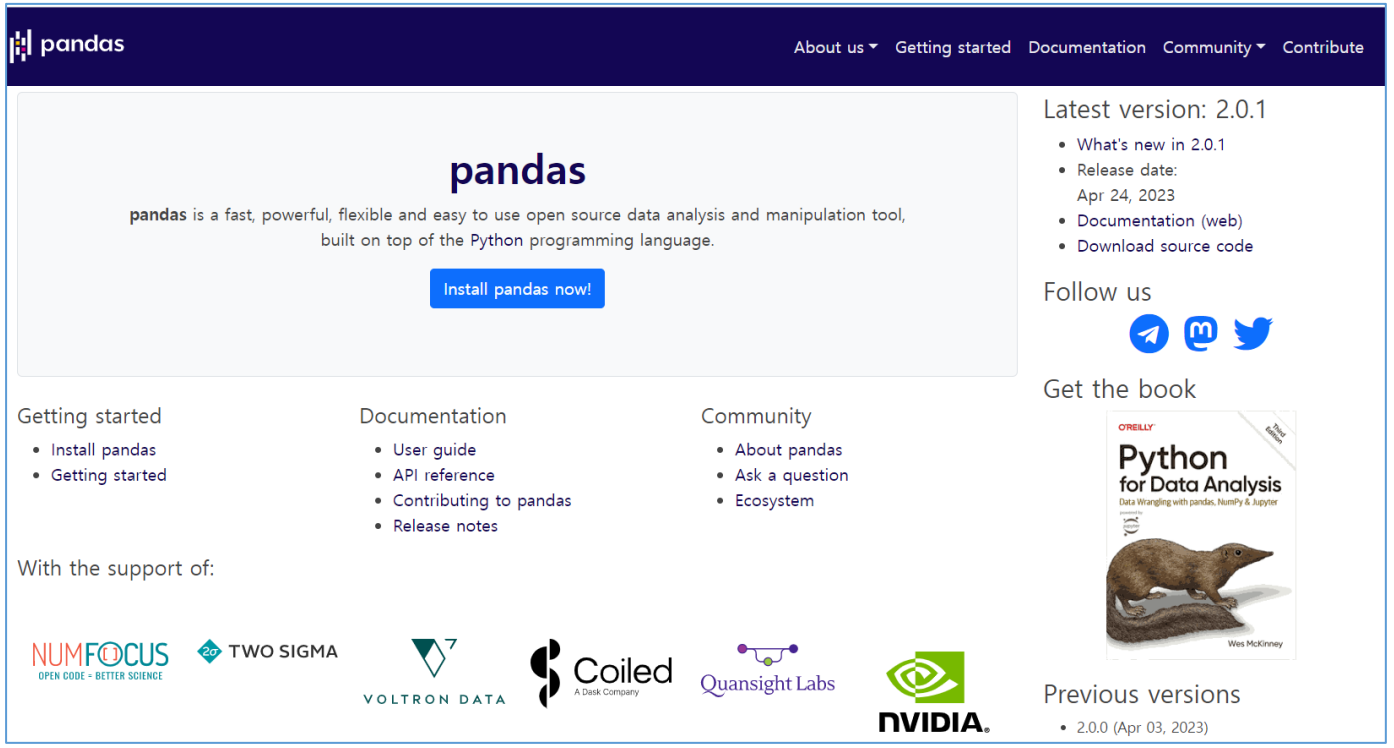
Python Library
Pandas

Pandas Data Structure

- 판다스(Pandas)
  - 파이썬에서 데이터 조작과 분석을 위한 라이브러리
  - 강력한 데이터 구조와 데이터 조작 기능을 제공하여 데이터를 쉽게 조작하고 분석할 수 있게 해줌
  - 데이터 구조로는 '시리즈(Series)'와 '데이터프레임(DataFrame)'이 있음

https://pandas.pydata.org/



Pandas Data Structure

- 시리즈(Series):
  - 시리즈는 1차원 배열과 같은 자료구조
  - 각 요소는 인덱스(Index)와 값(Value)으로 구성
  - 인덱스는 각 요소에 대한 고유한 이름 또는 레이블
  - 시리즈는 일반적으로 한 가지 데이터 유형(정수, 실수, 문자열 등)으로 구성된 열을 나타내는 데 사용

- 데이터프레임(DataFrame):
  - 데이터프레임은 2차원 테이블과 같은 자료구조
  - 여러 개의 시리즈를 모아서 열(column)로 구성되며, 각 열은 다른 데이터 유형을 가질 수 있음
  - 데이터프레임은 행(row)과 열(column)의 2차원 구조이므로 표 형태로 데이터를 저장하고 조작하는 데 유용
  - 열은 각각의 시리즈로 구성되며, 각 열은 고유한 이름을 가지고 있음
  - 행은 데이터프레임에서 하나의 데이터 레코드를 나타내며, 각 행은 인덱스를 가지고 있음

Series 주요 함수

- head(n): 시리즈의 처음 n개의 요소를 반환(기본값은 5)
- tail(n): 시리즈의 마지막 n개의 요소를 반환(기본값은 5)
- index: 시리즈의 인덱스를 반환
- values: 시리즈의 값들을 반환
- dtype: 시리즈의 데이터 타입을 반환
- size: 시리즈의 크기, 즉 요소의 개수를 반환
- shape: 시리즈의 형태를 반환(크기를 튜플 형태로 표현)
- describe(): 시리즈의 기술 통계량을 요약하여 반환(평균, 표준편차, 최솟값, 최댓값 등을 제공)
- mean(): 시리즈의 평균을 계산하여 반환
- sum(): 시리즈의 합계를 계산하여 반환

Series 주요 함수

- min(): 시리즈의 최솟값을 반환
- max(): 시리즈의 최댓값을 반환
- std(): 시리즈의 표준편차를 계산하여 반환
- median(): 시리즈의 중앙값을 계산하여 반환
- value_counts(): 시리즈 내의 각 값의 빈도를 계산하여 반환
- sort_values(): 시리즈의 값을 기준으로 정렬된 시리즈를 반환
- iloc(): (인덱스)를 기반으로 데이터에 접근하는 함수(정수 인덱싱을 사용하여 데이터를 선택하고 추출)
- isnull(): 시리즈에서 누락된 값을 나타내는 불리언 시리즈를 반환
- fillna(value): 시리즈의 누락된 값을 지정한 값으로 채움

Example(Series)

```python
import pandas as pd

# 리스트 데이터 생성
data = [10, 20, 30, 40, 50]

# 판다스 시리즈 생성
series = pd.Series(data)

# 시리즈 출력
print(series)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

Example(Series)

```python
import pandas as pd

# 학생 점수 데이터
scores = pd.Series([85, 92, 78, 90, 88])

# 시리즈 출력
print("학생 점수:")
print(scores)
print()

# 특정 인덱스의 점수 조회
print("첫 번째 학생의 점수:", scores[0])
print("세 번째 학생의 점수:", scores[2])
print()

# 최소값, 최대값, 평균값 계산
print("최소 점수:", scores.min())
print("최대 점수:", scores.max())
print("평균 점수:", scores.mean())
print()

# 점수가 90 이상인 학생들의 인덱스 조회
high_scores_indices = scores[scores >= 90].index
print("점수가 90 이상인 학생들의 인덱스:", high_scores_indices)
```

```
학생 점수:
0    85
1    92
2    78
3    90
4    88
dtype: int64

첫 번째 학생의 점수: 85
세 번째 학생의 점수: 78

최소 점수: 78
최대 점수: 92
평균 점수: 86.6

점수가 90 이상인 학생들의 인덱스: Int64Index([1, 3], dtype='int64')
```

Example(Series_Stack)

```
import pandas as pd

class Stack:
    def __init__(self):
        self.stack = pd.Series()

    def push(self, value):
        self.stack = self.stack.append(pd.Series([value]))

    def pop(self):
        if self.is_empty():
            return None
        popped_item = self.stack.iloc[-1]
        self.stack = self.stack.iloc[:-1]
        return popped_item

    def is_empty(self):
        return len(self.stack) == 0
```

Example(Series_Stack)

```
# 스택 예제
stack = Stack()
stack.push(10)
stack.push(20)
stack.push(30)

print("스택:")
print(stack.stack)
print()

popped_item = stack.pop()
print("스택에서 제거된 요소:", popped_item)
print()

print("스택:")
print(stack.stack)
```

```
스택:
0    10
0    20
0    30
dtype: int64

스택에서 제거된 요소:
30

스택:
0    10
0    20
dtype: int64
```

DataFrame 주요 함수

- 판다스 데이터프레임(DataFrame)은 테이블 형식의 데이터를 다루기 위한 2차원 데이터 구조
- pd.DataFrame(data, index, columns):
  - 데이터프레임을 생성하는 함수
  - data는 데이터를 입력하는 인자로, 리스트, 배열, 사전 등 다양한 형태의 데이터를 사용할 수 있음
  - index는 행 인덱스를, columns는 열 인덱스를 지정하는 인자
- df.head(n): 데이터프레임의 첫 n개의 행을 반환(기본값은 5)
- df.tail(n): 데이터프레임의 마지막 n개의 행을 반환(기본값은 5)
- df.info(): 데이터프레임의 기본 정보를 출력(열의 데이터 타입, 결측치 유무, 메모리 사용량 등)
- df.describe(): 숫자형 열에 대한 기술 통계 정보를 출력(평균, 표준편차, 최소값, 최대값, 사분위수 등)
- df.shape: 데이터프레임의 크기를 반환(행의 개수, 열의 개수)
- df.columns: 데이터프레임의 열 인덱스를 반환

DataFrame 주요 함수

- df.index: 데이터프레임의 행 인덱스를 반환
- df.values: 데이터프레임의 값을 2차원 배열 형태로 반환
- df.loc[row_label, col_label]: 라벨을 이용하여 행과 열을 선택
- df.iloc[row_index, col_index]: 위치(정수)를 이용하여 행과 열을 선택
- df.isnull(): 결측치(null)가 있는 위치를 확인하여 불리언 형태로 반환
- df.dropna(): 결측치(null)가 있는 행 또는 열을 삭제
- df.fillna(value): 결측치(null)를 지정한 값으로 채움
- df.sort_values(column): 지정한 열을 기준으로 데이터프레임을 정렬
  - 기본적으로 오름차순으로 정렬되며, ascending=False 옵션을 사용하여 내림차순으로 정렬

Example(DataFrame)

```python
import pandas as pd

# 데이터프레임 생성
data = {
    'Name': ['John', 'Emma', 'Liam', 'Olivia', 'Noah'],
    'Age': [25, 28, 24, 27, 26],
    'Score': [85, 92, 78, 90, 88]
}

df = pd.DataFrame(data)
print("데이터프레임:")
print(df)
print()

# head() - 첫 3개의 행 출력
print("첫 3개의 행:")
print(df.head(3))
print()

# describe() - 기술 통계 정보 출력
print("기술 통계 정보:")
print(df.describe())
print()
```

```
데이터프레임:
      Name  Age  Score
0     John   25     85
1     Emma   28     92
2     Liam   24     78
3   Olivia   27     90
4     Noah   26     88

첫 3개의 행:
   Name  Age  Score
0  John   25     85
1  Emma   28     92
2  Liam   24     78

기술 통계 정보:
             Age      Score
count   5.000000   5.000000
mean   26.000000  86.600000
std     1.581139   5.458938
min    24.000000  78.000000
25%    25.000000  85.000000
50%    26.000000  88.000000
75%    27.000000  90.000000
max    28.000000  92.000000
```

Example(DataFrame)

```python
# shape - 데이터프레임의 크기 출력
print("데이터프레임의 크기:")
print(df.shape)
print()

# columns - 열 이름 출력
print("열 이름:")
print(df.columns)
print()

# index - 행 인덱스 출력
print("행 인덱스:")
print(df.index)
print()

# loc - 특정 행과 열 선택
print("특정 행과 열 선택:")
print(df.loc[2:3, 'Name':'Score'])
print()

# sort_values() - 점수를 기준으로 내림차순 정렬
df_sorted = df.sort_values('Score', ascending=False)
print("점수 기준 내림차순 정렬:")
print(df_sorted)
```

```
데이터프레임의 크기:
(5, 3)

열 이름:
Index(['Name', 'Age', 'Score'], dtype='object')

행 인덱스:
RangeIndex(start=0, stop=5, step=1)

특정 행과 열 선택:
     Name  Age  Score
2    Liam   24     78
3  Olivia   27     90

점수 기준 내림차순 정렬:
     Name  Age  Score
1    Emma   28     92
3  Olivia   27     90
4    Noah   22     88
0    John   25     85
2    Liam   24     78
```

Example(Grade)

```python
import pandas as pd
import random

# 학과, 학번, 성명, 과목 리스트 생성
departments = ['컴퓨터공학', '화학공학', '생명공학', '반도체공학']
student_ids = [202301, 202302, 202303, 202304]
names = ['김학생', '이학생', '박학생', '최학생']
subjects = ['프로그래밍', '미적분', '물리', '화학']

# 데이터프레임 생성을 위한 빈 리스트 생성
data = []

# 학생별 랜덤 점수 생성 및 학점 부여
for i in range(len(names)):
    scores = [random.randint(80, 100) for _ in range(len(subjects))]
    grades = []
    for score in scores:
        if score >= 90:
            grades.append('A')
        elif score >= 80:
            grades.append('B')
        elif score >= 70:
            grades.append('C')
        else:
            grades.append('F')

    # 데이터 리스트에 학생 정보 추가
    for j in range(len(subjects)):
        data.append([departments[i], student_ids[i], names[i], subjects[j], scores[j], grades[j]])

# 데이터프레임 생성
df = pd.DataFrame(data, columns=['학과', '학번', '성명', '과목', '점수', '학점'])

# 결과 출력
print(df)
```

	학과	학번	성명	과목	점수	학점
0	컴퓨터공학	202301	김학생	프로그래밍	85	B
1	컴퓨터공학	202301	김학생	미적분	87	B
2	컴퓨터공학	202301	김학생	물리	84	B
3	컴퓨터공학	202301	김학생	화학	92	A
4	화학공학	202302	이학생	프로그래밍	97	A
5	화학공학	202302	이학생	미적분	96	A
6	화학공학	202302	이학생	물리	86	B
7	화학공학	202302	이학생	화학	94	A
8	생명공학	202303	박학생	프로그래밍	85	B
9	생명공학	202303	박학생	미적분	85	B
10	생명공학	202303	박학생	물리	99	A
11	생명공학	202303	박학생	화학	96	A
12	반도체공학	202304	최학생	프로그래밍	95	A
13	반도체공학	202304	최학생	미적분	84	B
14	반도체공학	202304	최학생	물리	98	A
15	반도체공학	202304	최학생	화학	100	A

Example(StudentManagement)

```python
import pandas as pd

class StudentManagement:
    def __init__(self):
        self.df = pd.DataFrame(columns=['학번', '성명', '학과', '학점'])

    def add_student(self, 학번, 성명, 학과, 학점):
        self.df = self.df.append({'학번': 학번, '성명': 성명, '학과': 학과, '학점': 학점}, ignore_index=True)
        print(f"{성명} 학생이 추가되었습니다.")

    def remove_student(self, 학번):
        if 학번 in self.df['학번'].values:
            self.df = self.df[self.df['학번'] != 학번]
            print(f"학번 {학번} 학생이 삭제되었습니다.")
        else:
            print(f"학번 {학번} 학생이 존재하지 않습니다.")

    def search_student(self, 학번):
        if 학번 in self.df['학번'].values:
            student = self.df.loc[self.df['학번'] == 학번]
            print(f"학번: {student['학번'].values[0]}")
            print(f"성명: {student['성명'].values[0]}")
            print(f"학과: {student['학과'].values[0]}")
            print(f"학점: {student['학점'].values[0]}")
        else:
            print(f"학번 {학번} 학생이 존재하지 않습니다.")

    def display_students(self):
        if len(self.df) > 0:
            print("전체 학생 목록:")
            print(self.df)
        else:
            print("등록된 학생이 없습니다.")
```

Example(StudentManagement)

```
# 학생 관리 시스템 예제
management = StudentManagement()

management.add_student(20231001, '김철수', '컴퓨터공학', 'A+')
management.add_student(20231002, '이영희', '화학공학', 'B+')
management.add_student(20231003, '박민지', '생명공학', 'A-')
management.add_student(20231004, '최성준', '반도체공학', 'A-')

management.display_students()

management.search_student(20231002)

management.remove_student(20231003)

management.display_students()
```

```
김철수 학생이 추가되었습니다.
이영희 학생이 추가되었습니다.
박민지 학생이 추가되었습니다.
최성준 학생이 추가되었습니다.
전체 학생 목록:
          학번   성명      학과  학점
0  20231001  김철수  컴퓨터공학   A+
1  20231002  이영희   화학공학   B+
2  20231003  박민지   생명공학   A-
3  20231004  최성준  반도체공학   A-
학번: 20231002
성명: 이영희
학과: 화학공학
학점: B+
학번 20231003 학생이 삭제되었습니다.
전체 학생 목록:
          학번   성명      학과  학점
0  20231001  김철수  컴퓨터공학   A+
1  20231002  이영희   화학공학   B+
3  20231004  최성준  반도체공학   A-
```