Pandas

강사 소개

- 이우진
  - 한양대학교 sw교육전담교수
  - 소프트웨어공학 전공

Pandas 소개

- Pandas는 파이썬에서 사용하는 데이터 분석 라이브러리로 '판다스'라고 읽는다.

- Pandas는 다차원으로 구조화된 데이터를 뜻하는 계량 경제학 용어인 Panel data와 파이썬 데이터 분석인 Python data analysis에서 따온 이름이다.

- Pandas는 안정적으로 대용량의 데이터를 처리하는데 편리한 도구이다.

Pandas 소개

- Pandas는 NumPy의 고성능 배열 계산 기능과 스프레드시트, SQL과 같은 관계형 데이터베이스의 데이터 조작 기능을 조합한 것이다.

- Pandas는 series와 dataframe 자료구조를 제공한다.
  - ✓ Series : list와 dictionary의 장점을 섞어 놓은 듯한 자료구조
  - ✓ DataFrame : 행과 열로 이루어진 2차원 형태의 자료구조

- Pandas의 기능을 이용해 데이터의 재배치와 집계, 부분집합 구하기 등을 보다 쉽게 할 수 있다.

List (리스트)

- 요소들의 모음 … 집합과 비슷, but 순서 있음

- 예)
  - [1, 2, 3, 4, 5]
  - [5, 4, 3, 2, 1]
  - ['aa', 'b', 'cde', 'fghi']
  - [3, 8, 1, 3, 2]
  - [1, 'a', [1, 2], 'b']

List (리스트)

- 관련 연산자

연산자	사용 형태	의미	예
+	리스트 + 리스트	두 리스트 연결 시키기	[1, 2, 3] + ['a', 'b', 'c'] → [1, 2, 3, 'a', 'b', 'c']
*	리스트 * 숫자	리스트를 숫자 만큼 반복하여 연결	[1, 2] * 4 → [1, 2, 1, 2, 1, 2, 1, 2]

  - 우선 순위: * > +
  - * 사용 시 숫자
    - ✓ 정수형만 가능
    - ✓ 0이나 음수일 경우: [] (비어있는 리스트)

List (리스트) – 부분 정보 활용하기

- 리스트 인덱싱(indexing)
  - 리스트의 요소 하나를 선택
  - 인덱싱
    - ✓ 왼쪽부터 0, 1, 2, … 로 증가
    - ✓ 오른쪽에서 -1, -2, -3, …로 감소

```
 0    1    2    3    4
['a', 'b', 'c', 'd', 'e']
-5   -4   -3   -2   -1
```

List (리스트) – 부분 정보 활용하기

- 리스트 인덱싱(indexing)
  - 예시 – 왼쪽부터 증가하는 인덱스 사용

```
In [1]: a = [1, 2 ,3]

In [2]: a[0]
Out[2]: 1

In [3]: a[1]
Out[3]: 2

In [4]: a[2]
Out[4]: 3

In [5]: a[3]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-5-f75b6be7d8e3> in <module>
----> 1 a[3]

IndexError: list index out of range
```

List (리스트) – 부분 정보 활용하기

- 리스트 인덱싱(indexing)
  - 예시 – 오른쪽부터 감소하는 인덱스 사용

```
In [1]: a = [1, 2, 3]

In [2]: a[-1]
Out[2]: 3

In [3]: a[-2]
Out[3]: 2

In [4]: a[-3]
Out[4]: 1
```

List (리스트) – 부분 정보 활용하기

- 문자열 인덱싱(indexing)
  - 문자열도 리스트와 같은 방식으로 인덱싱 가능
  - 예시 – 왼쪽부터 증가하는 인덱스 사용

```
In [1]: a = '123'

In [2]: a[0]
Out[2]: '1'

In [3]: a[1]
Out[3]: '2'

In [4]: a[2]
Out[4]: '3'

In [5]: a[3]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-5-f75b6be7d8e3> in <module>
----> 1 a[3]

IndexError: string index out of range
```

List (리스트) – 부분 정보 활용하기

- 문자열 인덱싱(indexing)
  - 문자열도 리스트와 같은 방식으로 인덱싱 가능
  - 예시 – 오른쪽부터 감소하는 인덱스 사용

```
In [1]: a = '123'

In [2]: a[-1]
Out[2]: '3'

In [3]: a[-2]
Out[3]: '2'

In [4]: a[-3]
Out[4]: '1'
```

List (리스트) – 부분 정보 활용하기

- 리스트 슬라이싱(slicing)
  - 리스트의 일부분을 잘라 냄
    - ✓ 결과는 리스트
    - ✓ 부분 집합과 비슷
  - 사용하는 방법: *변수*[시작 인덱스:끝 인덱스:스텝]
    - ✓ 시작 인덱스: 범위의 시작, 생략 시 0
    - ✓ 끝 인덱스: 범위의 끝, 생략 시 리스트의 크기
      - ❖ 끝 인덱스는 미포함, 직전 값까지만 포함
    - ✓ 스텝: 자료를 취하는 간격, 생략 시 1

List (리스트) – 부분 정보 활용하기

- 리스트 슬라이싱(slicing)
  - 예시

```
In [1]: a = [0, 1, 2, 3, 4, 5]

In [2]: a[0:2]
Out[2]: [0, 1]

In [3]: a[2:]
Out[3]: [2, 3, 4, 5]

In [4]: a[:4:2]
Out[4]: [0, 2]

In [5]: a[-3:]
Out[5]: [3, 4, 5]

In [6]: a[:-3]
Out[6]: [0, 1, 2]
```

List (리스트) – 부분 정보 활용하기

- 문자열 슬라이싱(slicing)
  - 문자열도 리스트와 같은 방식으로 슬라이싱 가능
  - 예시

```
In [1]: a = '012345'

In [2]: a[0:2]
Out[2]: '01'

In [3]: a[2:]
Out[3]: '2345'

In [4]: a[:3]
Out[4]: '012'

In [5]: a[-3:]
Out[5]: '345'

In [6]: a[:-3]
Out[6]: '012'
```

List (리스트) - 생성하기

- 리스트 생성 관련 함수
  - list(), split(), range(끝), range(시작, 끝), range(시작, 끝, 스텝)

```
list('12345')  →  ['1', '2', '3', '4', '5']
```

```
a = '1 2 3 4 5‘
b = a.split()     → b = ['1', '2', '3', '4', '5']
```

```
a = '1:2:3:4:5'
b = a.split(':')    → b = ['1', '2', '3', '4', '5']
```

```
list(range(4)) → [0, 1, 2, 3]

list(range(3, 5)) → [3, 4]

list(range(2, 11, 2)) → [2, 4, 6, 8, 10]

list(range(9, 1, -2)) → [9, 7, 5, 3]
```

List (리스트) - 수정하기

- 리스트에 추가, 삭제 관련 함수

사용 방법	의미	예시(a = [1, 2, 3]일 때)
리스트.append(요소)	리스트의 마지막에 요소를 추가	a.append(4) → a = [1, 2, 3, 4]
리스트.extend(리스트2)	리스트의 마지막에 리스트2를 추가	a.extend([4, 5]) → a = [1, 2, 3, 4, 5]
리스트.insert(index, 요소)	리스트의 index 위치에 요소를 추가	a.insert(1, 4) → a = [1, 4, 2, 3]
del 리스트[index]	리스트의 index에 위치한 요소를 삭제	del a[1] → a = [1, 3]
리스트.remove(요소)	리스트에서 첫 번째로 나오는 요소를 삭제	a.remove(1) → a = [2, 3]

실습

- 생년월일을 입력 받아 홀수 번째 글자들로만 이루어진 문자열을 출력하는 프로그램을 작성하시오.
  - 문자열 슬라이싱 이용하기

실행결과(밑줄 친 부분은 키보드로 입력 받는 부분)

```
생년월일 입력(yyyymmdd): 20190101
2100
```

실습

- 문자열을 입력 받아 거꾸로 출력하는 프로그램을 작성하시오.
  - 문자열 슬라이싱 이용하기

실행결과(밑줄 친 부분은 키보드로 입력 받는 부분)

```
문자열을 입력하시오.
Hello
거꾸로 문자열:
olleH
```

Dictionary

- Key와 value 쌍들의 모음
- {} 사용
- 예:
  - {'name':'gdhong', 'phone':'0222200001', 'addr':['Seoul', 'Wangsimni']}
- Key
  - 중복 X, list형 불가
- Value
  - 숫자, 문자열, list, dictionary 등 대부분의 자료형 가능

Dictionary

- Value의 선택
  - List와 비교하였을 때 index 번호(0부터 시작) 대신 key 값으로 value를 선택
- 예시

```
In [1]: a = {1:'a', 2:'b', 'three':'c'}

In [2]: a[1]
Out[2]: 'a'

In [3]: a[2]
Out[3]: 'b'

In [4]: a['three']
Out[4]: 'c'
```

```
In [5]: a.get(1)
Out[5]: 'a'

In [6]: a.get(2)
Out[6]: 'b'

In [7]: a.get('three')
Out[7]: 'c'
```

Dictionary - 수정하기

- 요소 수정, 추가, 삭제
  - 수정: 기존에 있는 쌍에서 value만 수정
  - 추가: 기존에 없는 key에 대한 value를 대입
  - 삭제: del dictionary[키]
  - 모두 삭제: clear()

Dictionary - 수정하기

- 요소 수정, 추가, 삭제
  - 예시

```
In [1]: a = {1:'a', 2:'b', 'three':'c'}

In [2]: a[1] = 'abc'

In [3]: a
Out[3]: {1: 'abc', 2: 'b', 'three': 'c'}

In [4]: a[4] = 'd'

In [5]: a
Out[5]: {1: 'abc', 2: 'b', 'three': 'c', 4: 'd'}
```

```
In [6]: del a['three']

In [7]: a
Out[7]: {1: 'abc', 2: 'b', 4: 'd'}

In [8]: a.clear()

In [9]: a
Out[9]: {}
```

Dictionary – 정보 추출하기

- Key만 얻기, value만 얻기
  - keys(): dictionary의 key만 모아서 반환
  - values(): dictionary의 value만 모아서 반환
  - 예시

```
In [1]: a = {1:'a', 2:'b', 'three':'c'}

In [2]: a.keys()
Out[2]: dict_keys([1, 2, 'three'])

In [3]: list(a.keys())
Out[3]: [1, 2, 'three']

In [4]: a.values()
Out[4]: dict_values(['a', 'b', 'c'])

In [5]: list(a.values())
Out[5]: ['a', 'b', 'c']
```

실습

- n, m을 입력 받아 'nXm'을 key로, n*1, n*2, …, n*m을 요소로 갖는 list를 value로 갖는 dictionary를 출력하시오.

실행결과(밑줄 친 부분은 키보드로 입력 받는 부분)

```
Input a number: 5
Input a number: 8
{'5X8': [5, 10, 15, 20, 25, 30, 35, 40]}
```

```
Input a number: 2
Input a number: 7
{'2X7': [2, 4, 6, 8, 10, 12, 14]}
```

실습

- Dictionary를 이용하여 다음과 같이 실행되는 프로그램을 작성하시오.

실행결과(밑줄 친 부분은 키보드로 입력 받는 부분)

```
학생 수: 3
학번: 11
이름: 홍길동
학번: 22
이름: 김철수
학번: 33
이름: 이영미
입력된 값:
{'11': '홍길동', '22': '김철수', '33': '이영미'}
검색할 학번: 22
학번 22에 해당하는 학생의 이름은 김철수입니다.
계속 검색하시겠습니까?(y/n) y
검색할 학번: 11
학번 11에 해당하는 학생의 이름은 홍길동입니다.
계속 검색하시겠습니까?(y/n) n
```

실습 - 답안

Array

- 배열은 리스트와 비슷하지만 다음과 같은 점에서 다르다.
  - 모든 원소가 같은 자료형이어야 한다.
  - 원소의 개수를 바꿀 수 없다.

- 파이썬은 자체적으로 배열 자료형을 제공하지 않는다.

- 배열은 NumPy 라이브러리에서 제공한다.

Array - 생성

- List를 array로 만들기
  - 생성방법: 배열 = np.array(리스트)
  - 1차원 vs. 2차원

```
In [1]: import numpy as np

In [2]: a = np.array([1, 2, 3])

In [3]: a
Out[3]: array([1, 2, 3])
```

```
In [1]: import numpy as np

In [2]: a = np.array([[1, 2, 3], [4, 5, 6]])

In [3]: a
Out[3]: array([[1, 2, 3],
               [4, 5, 6]])
```

Array - 생성

- 동일 간격으로 등분한 array 생성
  - 배열 = np.linspace(시작, 끝, 숫자개수)
  - 예시

```
In [1]: import numpy as np

In [2]: a = np.linspace(0, 15, 4)

In [3]: a
Out[3]: array([ 0.,  5., 10., 15.])
```

```
In [1]: import numpy as np

In [2]: b = np.linspace(0, 1, 5)

In [3]: print(b)
[0.   0.25 0.5  0.75 1.  ]
```

Array - 생성

- 수열로 구성된 array 생성
  - range + array
  - 배열 = np.arange(시작, 끝, 증감)
  - 예시

```
In [1]: import numpy as np

In [4]: a = np.arange(10)

In [5]: print(a)
[0 1 2 3 4 5 6 7 8 9]
```

```
In [6]: b = np.arange(3, 10)

In [7]: print(b)
[3 4 5 6 7 8 9]

In [8]: c = np.arange(3, 10, 2)

In [9]: print(c)
[3 5 7 9]
```

Array - list에 비해 편리한 점

- 다수의 값에 동일한 연산하기
  - List의 기본 연산으로는 해결 X

```
In [1]: a = [1, 2, 3]
In [2]: b = a * 2
In [3]: b
Out[3]: [1, 2, 3, 1, 2, 3]
```

```
In [1]: a = [1, 2, 3]
In [2]: b = []
In [3]: for n in a:
            b.append(n*2)
In [4]: b
Out[4]: [2, 4, 6]
```

Array - list에 비해 편리한 점

- 다수의 값에 동일한 연산하기
  - Array 사용

```
In [1]: a = [1, 2, 3]

In [2]: b = []

In [3]: for n in a:
            b.append(n*2)

In [4]: b
Out[4]: [2, 4, 6]
```

```
In [1]: import numpy as np

In [2]: a = np.array([1, 2, 3])

In [3]: b = a * 2

In [4]: print(b)
[2 4 6]
```

Array - list에 비해 편리한 점

- 다수의 값에 동일한 연산하기
  - List의 기본 연산으로는 해결 X

```
In [5]: a = [1, 2, 3]
In [6]: b = [4, 5, 6]
In [7]: c = a + b
In [8]: c
Out[8]: [1, 2, 3, 4, 5, 6]
```

```
In [5]: a = [1, 2, 3]
In [6]: b = [4, 5, 6]
In [9]: c = []
In [10]: for i in range(3):
             c.append(a[i] + b[i])
In [11]: c
Out[11]: [5, 7, 9]
```

Array - list에 비해 편리한 점

- 다수의 값에 동일한 연산하기
  - Array 사용

```
In [5]: a = [1, 2, 3]

In [6]: b = [4, 5, 6]

In [9]: c = []

In [10]: for i in range(3):
             c.append(a[i] + b[i])

In [11]: c
Out[11]: [5, 7, 9]
```

```
In [1]: import numpy as np

In [2]: a = np.array([1, 2, 3])

In [3]: b = np.array([4, 5, 6])

In [4]: c = a + b

In [5]: print(c)
[5 7 9]
```

Array - list에 비해 편리한 점

- 합 구하기
  - List의 기본 연산으로는 해결 X

```
In [1]: a = [[1,2,3], [4,5,6]]

In [2]: b = []

In [3]: for i in range(3):
            b.append(a[0][i] + a[1][i])

In [4]: b
Out[4]: [5, 7, 9]
```

```
In [1]: import numpy as np

In [2]: a = np.array([[1,2,3], [4,5,6]])

In [3]: b = a.sum(axis=0)

In [4]: print(b)
[5 7 9]
```

Array - list에 비해 편리한 점

- 합 구하기
  - List의 기본 연산으로는 해결 X

```
In [1]: a = [[1,2,3], [4,5,6]]

In [2]: b = []

In [3]: for i in range(2):
            b.append(a[i][0] + a[i][1] + a[i][2])

In [4]: b
Out[4]: [6, 15]
```

```
In [1]: import numpy as np

In [2]: a = np.array([[1,2,3], [4,5,6]])

In [3]: b = a.sum(axis=1)

In [4]: print(b)
[ 6 15]
```

Array - list와의 유사점

- Indexing/slicing 방법이 비슷
  - 예시

```
In [1]: import numpy as np

In [6]: a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [7]: a
Out[7]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [8]: a[3]
Out[8]: 3

In [9]: a[3:]
Out[9]: array([3, 4, 5, 6, 7, 8, 9])
```

```
In [10]: a[:3]
Out[10]: array([0, 1, 2])

In [11]: a[::2]
Out[11]: array([0, 2, 4, 6, 8])

In [12]: a[::-1]
Out[12]: array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])

In [13]: a[-2:2:-1]
Out[13]: array([8, 7, 6, 5, 4, 3])
```

Array - list와의 유사점

- Indexing/slicing 방법이 비슷
  - 예시

```
In [1]: import numpy as np

In [2]: a = np.array([[3, 0, 5, 5, 1, 0], [9, 7, 0, 3, 5, 8], [1, 1, 9, 7, 8, 0]])

In [3]: print(a)
[[3 0 5 5 1 0]
 [9 7 0 3 5 8]
 [1 1 9 7 8 0]]

In [4]: a[2]
Out[4]: array([1, 1, 9, 7, 8, 0])

In [5]: a[2, 3]
Out[5]: 7

In [7]: a[1:, 3:]
Out[7]: array([[3, 5, 8],
               [7, 8, 0]])
```

실습

- Array를 이용하여 다음과 같이 실행되는 프로그램을 작성하시오.

실행결과(밑줄 친 부분은 키보드로 입력 받는 부분)

```
학생 수: 3
1번째 학생의 국어,영어,수학 성적(,로 구분): 50,60,80
2번째 학생의 국어,영어,수학 성적(,로 구분): 80,70,90
3번째 학생의 국어,영어,수학 성적(,로 구분): 70,80,80
학생들의 성적:
[[50 60 80]
 [80 70 90]
 [70 80 80]]
각 학생의 총점:
[190 240 230]
각 과목의 평균:
[66.66666667 70.         83.33333333]
```

실습 - 답안