Elixirr

파이썬으로 배우는 따릉이 데이터 분석과 시각화

3회차
데이터 가공

수업 일정

전체 수업은 13회로 구성된다.

1
- 따릉이 이용현황 파악
- 문제 정의
- 파이썬 및 사용할 라이브러리 소개

2
- 비주얼 스튜디오 코드 설치
- 따릉이 데이터 수집

3
- 파이썬 라이브러리
- 따릉이 데이터프레임 만들기

4
- 따릉이 데이터프레임 관찰하기

5
- 시간 개념에 따른 데이터 분석을 위한 컬럼 추가

6
- 장소적 특징에 따른 데이터 분석을 위한 컬럼 추가

7
- 시간 개념에 따른 데이터 분석 및 시각화-(1)

8
- 시간 개념에 따른 데이터 분석 및 시각화-(2)

9
- 장소 특징에 따른 데이터 분석 및 시각화-(1)

10
- 장소 특징에 따른 데이터 분석 및 시각화-(2)

수업 일정

전체 수업은 13회로 구성된다.

- 시간 개념 X 장소 특징에 따른 데이터 분석 및 시각화

- 주말과 평일에 이용건수가 많은 대여소 데이터 분석 및 시각화

- 문제 정의에 맞춘 해결방안 도출
- 총정리

Elixirr

1. 문제정의
2. 데이터 수집
3. **데이터 가공**
4. 데이터 분석
5. 시각화 및 탐색

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 현재 따릉이가 이용되고 있는 현황 조사
- 미래에 개선될 점에 대해서 분석주제 정의
- 파이썬과 판다스 소개

- 환경 설정 비주얼 스튜디오 코드(Visual Studio Code) 설치
- 따릉이 데이터 수집

- **확보한 따릉이 데이터 관찰 및 기초 데이터 분석**
- 분석에 필요한 데이터 컬럼 추가 및 가공

- 시간 개념에 따른 데이터 분석
- 장속적 특징에 따른 데이터 분석
- 두 개념을 동시에 고려한 데이터 분석
- 인기 대여점 Top50

- 4단계 데이터 분석을 가장 잘 반영할 수 있는 시각화
- 문제 정의에 맞춘 해결방안 도출

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

단계 1 : 분석할 데이터프레임 만들기

단계 2 : 데이터프레임 관찰하기

단계 3 : 분석주제에 맞는 새로운 컬럼 추가하기

따릉이 수업에서 사용되는 라이브러리

따릉이 분석에서는 다음의 라이브러리를 사용한다.

pandas 판다스
데이터 관리 주특기

column

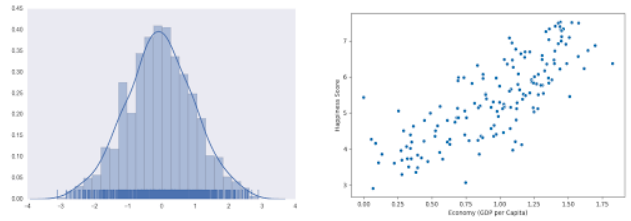
index

	Name	Age	Department
0	Jim	26	Sales
1	Dwight	28	Sales
2	Angela	27	Accounting
3	Tobi	32	Human Resources

DataFrame

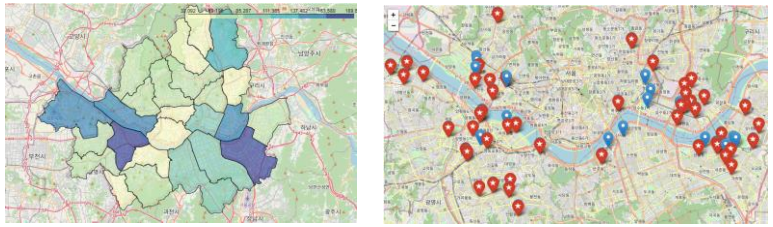
- 데이터 읽어 들이기, 가공 담당
- 데이터프레임(DataFrame)으로 표를 표현

matplotlib 맷플롯립 seaborn 시본
그래프



- 다양한 그래프 작성 기능을 제공하는 라이브러리
- **seaborn**은 자체적으로 분석해서 그래프를 제공

folium 폴리움
지도 그리기 (시각화)



- 대여소 위도, 경도 데이터를 활용하여 지도에 분석한 내용을 시각화

라이브러리 vs. 모듈

데이터 분석과 시각화에 필요한 라이브러리를 import 문을 이용하여 읽어 들인다.

```
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt

import folium
```

[파이썬 명령어]

- 라이브러리와 모듈을 가져다 쓰겠다는 명령어

[라이브러리 (library)]

- 다양한 자료형, 함수, 모듈들을 모아놓은 곳
  예) DataFrame : Pandas 내에 있는 자료형
- Python 설치 시 기본적 제공 외에는 별도 설치

[모듈 (module)]

- 라이브러리 내 함수, 변수, 데이터 셋 등을 용도별 모아 놓은 곳
- 자주 활용할 함수, 변수, 데이터 셋 등 정의

6개의 데이터 파일 읽어 들이기

6개로 나누어진 csv 파일을 읽어 들여서 하나의 데이터프레임으로 합한다.

bike_rent_1.csv → ① bikes_temp[0] = pd.read_csv()

bike_rent_2.csv → bikes_temp[1] = pd.read_csv()

bike_rent_3.csv → bikes_temp[2] = pd.read_csv()

bike_rent_4.csv → bikes_temp[3] = pd.read_csv()

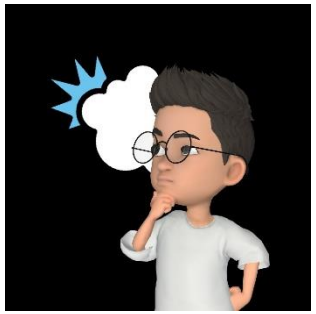
bike_rent_5.csv → bikes_temp[4] = pd.read_csv()

bike_rent_6.csv → bikes_temp[5] = pd.read_csv()

② **bikes** = pd.concat( )

	자전거번호	대여일시	대여 대여소번호	대여 대여소명	이용시간	이용거리
0	SPB-22040	2019-06-03 08:49:00	646	장한평역 1번출구 (국민은행앞)	27	1330
1	SPB-07446	2019-06-03 08:33:00	526	용답토속공원 앞	54	1180
2	SPB-20387	2019-06-05 08:27:00	646	장한평역 1번출구 (국민은행앞)	12	1930
3	SPB-16794	2019-06-05 08:46:00	646	장한평역 1번출구 (국민은행앞)	6	1340
4	SPB-18266	2019-06-10 08:27:00	529	장한평역 8번 출구 앞	5	1230

6개의 파일을 각각
읽어들일 때 반복을 피하는
방법이 없을까 ?



최종 데이터프레임 완성 과정

6개의 데이터 파일을 각각 읽어 들인 후, 합쳐진 최종 데이터프레임 : bikes

6개의 데이터 파일을 읽어 들일 딕셔너리 초기화	bikes_temp = {}
반복을 피하기 위한 순회가능한 목록 range() 함수	range(6)
for문으로 6개의 파일을 읽어 들여서 딕셔너리의 키와 값에 할당	for문
파일명을 쉽게 읽어 들이기 위해서 문자열과 format문 사용	문자열.format()
딕셔너리의 값을 하나로 연결해서 최종 데이터프레임 bikes 완성	pd.concat()

파이썬 문법 > 딕셔너리dictionary

정의 : 일상생활에서 사용하는 사전처럼 이름과 그 이름에 대응되는 내용이 하나의 항목으로 연결되어 있는 파이썬 자료구조이다. 이 때 이름을 키(key), 대응되는 내용을 값(value)라고 한다.

만들기 : 딕셔너리 이름 = { key1 : value1, key2 : value2, ... } 예) bikes_temp = { 0 : df_0, 1 : df_1, 2 : df_2 }

키(key) 값(value)

딕셔너리 이름 = {} -> 딕셔너리 초기화

항목 추출하기 : 딕셔너리 이름[항목의 키]

예) val0 = bikes_temp[0], val1 = bikes_temp[1], val2 = bikes_temp[2]

val0, val1, val2에 할당되는 값은 딕셔너리 bikes_temp 에서 키인 0, 1, 2에 대응되는 값인 데이터프레임 df_0, df_1, df_2가 할당된다.

딕셔너리의 값에 데이터프레임이 할당될 수 있구나

파이썬 문법 : 리스트list

정의 : 여러 데이터들을 잘 관리하기 위해서 묶어서 목록으로 관리할 수 있는 파이썬 자료구조

만들기 : 리스트의 이름 = **[** 항목1, 항목2, ...**]** or list(range(6)) 예) iter_nums = list(range(6)) = [0,1,2,3,4,5]

range()함수 : 연속된 정수를 생성하는 함수 예) range(6) -> range(0, 6) -> 0부터 5까지의 정수

인덱싱 : 리스트에 있는 여러 항목들은 모두 각각 그 위치가 0부터 시작하는 숫자로 매겨져 있다.

예) nums = [5, 4, 3, 2, 1, 0]
인덱스 -> 0 1 2 3 4 5

for문의 순회가능한 목록을 만들 때
리스트가 많이 사용되는구나.

파이썬 문법 > for 문

for 문은 실행구문들을 목록안의 항목 수 만큼 반복하는 제어구조다.

```
# 목록의 값을 차례로 변수로 받아 실행한다.

for 변수 in 순회가능한 자료구조 또는 목록 :
    실행구문1
    실행구문2
    …
```

```
# 3.5.2.3.1 for문 연습1
# 숫자 1,2,3 목록에서 차례로 a 의 값이 변하여 전달 되고 그 값이 출력된다.
for a in [1,2,3] :
    print (a)
    print (a*a)
```

```
1
1
2
4
3
9
```

1. 키워드 **in** 뒤에 기재된 자료 구조 또는 목록으로부터 항목을 처음부터 마지막까지 하나씩 꺼내어 변수에 넣고, for 블록의 실행구문들을 순서대로 실행한다.
2. 변수에 들어갈 값이 없게 되면 for 블록은 더 이상 실행되지 않고 빠져 나온다.

파이썬 자료 구조 -> 리스트 (List)

순회가능한 자료구조

- [1, 2, 3] ≠ [3, 2, 1]

순회 가능한 목록 : 파이썬 함수 > range(숫자)

range는 '범위'라는 뜻을 가진 영어 단어로, 파이썬에서 range 함수를 사용하면 간단히 정수 범위를 표현할 수 있다. 예를 들어, range(0, 10)은 0부터 9까지의 숫자 범위를 나타낸다.

```
range(10)
✓ 0.4s
range(0, 10)
```

0부터 10사이의 정수 범위를 나타낸다. 시작값 0은 생략 가능하다.

```
list(range(10))
✓ 0.4s
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

리스트로 나타내면 끝값은 포함되지 않는 것을 알 수 있다.

```
for i in range(10) :
    print(i)
✓ 0.4s
0
1
2
3
4
5
6
7
8
9
```

- For문과 함께 사용해서 반복 횟수를 정한다.
- 0부터 9까지 10번 반복할 수 있고 마지막값 10은 포함되지 않음을 유념한다.

파이썬과 판다스 데이터 타입

파이썬과 판다스는 각각 문자, 정수, 실수, 날짜, 불린 등의 데이터 타입을 가지고 있다.

파이썬 데이터 타입 (자료형)

str	• 문자열 • 따옴표로 둘러싸여 있으면 문자열이다.
int	• 정수
float	• 실수
bool	• 참 또는 거짓을 나타내는 Boolean • 참 = 1, 거짓 = 0
datetime	• 날짜 (datetime 패키지)

판다스 데이터 타입 (자료형)

object	• 문자열 • 따옴표로 둘러싸여 있으면 문자열이다.
int64	• 정수
float64	• 실수
bool	• 참 또는 거짓을 나타내는 Boolean • 참 = 1, 거짓 = 0
datetime64	• 날짜

파이썬 문법 > format 함수

정의 : 문자열 중간에 특정 변수값을 넣어 주기 위해서 사용하는 것이다.

만들기 : 문자열 내에 {}를 여러 개 넣은 후에, {} 순서에 대응되는 변수값을 추가할 수 있다.
문자열.format(변수) 형식으로 쓰며 문자열안에 {}를 여러 개 쓸 수 있고, 이 경우에 {}의 개수만큼 변수를 쓸 수 있다.

```
i = 10

print('./data/bike_rent_{}.csv'.format(i))
```

✓ 0.4s

```
./data/bike_rent_10.csv
```

문자열 내에 있는 {}에 format 함수의 변수값이 출력된다.

```
for i in range(3) :
    print('./data/bike_rent_{}.csv'.format(i+1))
```

✓ 0.7s

```
./data/bike_rent_1.csv
./data/bike_rent_2.csv
./data/bike_rent_3.csv
```

for문의 변수 i 값이 {}에 순서대로 들어가서 출력된다.

데이터프레임으로 읽어 들이기 : pd.read_csv( )

판다스 라이브러리를 불러온 후 CSV 파일을 읽어서 데이터프레임으로 만든 후, bikes_temp라는 딕셔너리의 값(value)에 할당된다.

bikes_temp[i] = pd.read_csv(파일명, encoding='cp949', parse_date=['대여일시'])

실제 코딩

```
bikes_temp = {}

for i in range(6):
    bikes_temp[i] = pd.read_csv('./data/bike_rent_{}.csv'.format(i+1), \
                                encoding = 'cp949', parse_dates=['대여일시'])
```

작은 따옴표 ' ' 안에 디렉토리를 포함한 파일명

※ bikes_temp = { 0 : bike_rent_1.csv, 1 : bike_rent_2.csv,
2 : bike_rent_3.csv, 3 : bike_rent_4.csv,
4 : bike_rent_5.csv, 5 : bike_rent_6.csv }

- pd. : 판다스 (as pd) 라이브러리에 속한
- read_csv( ) : 명령으로 CSV 파일을 읽고
- bikes_temp[i] 에 저장됨

데이터프레임 둘러보기

읽어 들인 데이터프레임을 살펴본다.

bikes_temp[0].head()

1 데이터프레임 : 행과 컬럼으로 이루어진 이차원 데이터

열 : 컬럼 **3**

행 : 인덱스 **2**

	자전거번호	대여일시	대여 대여소번호	대여 대여소명	이용시간	이용거리
0	SPB-22040	2019-06-03 08:49:00	646	장한평역 1번출구 (국민은행앞)	27	1330
1	SPB-07446	2019-06-03 08:33:00	526	용답토속공원 앞	54	1180
2	SPB-20387	2019-06-05 08:27:00	646	장한평역 1번출구 (국민은행앞)	12	1930
3	SPB-16794	2019-06-05 08:46:00	646	장한평역 1번출구 (국민은행앞)	6	1340
4	SPB-18266	2019-06-10 08:27:00	529	장한평역 8번 출구 앞	5	1230

4 값(value)

최종 데이터프레임으로 연결 : pd.concat()

bikes = pd.concat(bikes_temp, axis='index', ignore_index=True)

1 **2** **3** **4**

1. **pd.concat** : 복수 개의 시리즈 또는 데이터프레임을 수직 또는 수평으로 연결하는 함수
2. **bikes_temp** : 연결할 데이터프레임들이 있는 딕셔너리
3. **axis='index'** : 수직(인덱스방향) 연결시는 'index', 수평(컬럼방향) 연결시는 'columns'를 입력
4. **ignore_index=True** : 인덱스를 리셋하고 다시 0부터 순서대로 정한다.

```
# 6개의 파일을 이어붙여 최종 bikes 데이터프레임을 만든다.

bikes = pd.concat(bikes_temp, axis='index', ignore_index=True)
```

```
bikes.head()
```

	자전거번호	대여일시	대여 대여소번호	대여 대여소명	이용시간	이용거리
0	SPB-22040	2019-06-03 08:49:00	646	장한평역 1번출구 (국민은행앞)	27	1330
1	SPB-07446	2019-06-03 08:33:00	526	용답토속공원 앞	54	1180
2	SPB-20387	2019-06-05 08:27:00	646	장한평역 1번출구 (국민은행앞)	12	1930

나 지금 어느 단계를 공부하는 거지?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

단계 1 : 분석할 데이터프레임 만들기

데이터프레임 읽어들이기 -> **pd.read_csv()**

데이터프레임 연결하기 -> **pd.concat()**

퀴즈

퀴즈를
풀어봅시다

1. 이름과 이름에 대응되는 내용이 하나의 항목으로 연결되어 키 : 값의 구조를 갖는 파이썬 자료구조는 ?

2. 시리즈와 데이터프레임이라는 데이터 형식을 제공하여 데이터 분석에 유용한 라이브러리는 ?

3. CSV 파일을 읽어들여서 데이터프레임으로 만들어주는 명령어는 ?

4. 복수 개의 시리즈 또는 데이터프레임을 수직 또는 수평으로 연결하는 함수는 ?

실습 시작

GD쌤

이제부터 Visual Studio Code 실습 환경에서 지금까지 배운 내용을 실습해 보겠습니다.

앞에서 배웠던 내용을 Visual Studio Code에서 직접 실습해보면 더욱 이해하기 편리할 것입니다.

수업 마무리

GD쌤

지금까지 3회차 수업내용을 배워 보았습니다.

다음 시간에는 4회차 수업내용으로 따릉이 데이터를 관찰해 보겠습니다.

수고 많으셨어요. 다음 시간에 만나요.