[Python] Pandas 사용법 - 다양한 함수 사용(데이터 입출력, 대소문자변환, 공백제거, 문자열 접근) — 나무늘보의 개발 블로그

노트북:	첫 번째 노트북
만든 날짜:	2020-11-05 오전 1:31
URL:	https://continuous-development.tistory.com/132?category=736681

Python

[Python] Pandas 사용법 - 다양한 함수 사용 (데이터 입출력, 대소문자변환, 공백제거, 문자열 접근)

2020. 10. 15. 23:48 수정 삭제 공개

데이터 입출력

```
In [81]: court_df = pd.read_csv('court_code.txt',sep='₩t',encoding='cp949')
         dfInfo(court_df)

         df shape : (46180, 3)
         df size : 138540
         df ndim : 2
         df index : RangeIndex(start=0, stop=46180, step=1)
         df index type : <class 'pandas.core.indexes.range.RangeIndex'>
         df columns : Index(['법정동코드', '법정동명', '폐지여부'], dtype='object')
         df columns type : <class 'pandas.core.indexes.base.Index'>
```

read_csv를 통해 해당 파일을 읽을 수 있다.

sep는 간격을 어떤거로 두냐는 뜻이다.
encoding 은 파일이 구성되어 있는 형식이라고 보면 된다.

#head()

```
In [84]: court_df.head()
```

Out[84]:

	법정동코드	법정동명	폐지여부
0	1100000000	서울특별시	존재
1	1111000000	서울특별시 종로구	존재
2	1111010100	서울특별시 종로구 청운동	존재
3	1111010200	서울특별시 종로구 신교동	존재
4	1111010300	서울특별시 종로구 궁정동	존재

맨앞에 5개의 값을 가져온다.

tail()

```
In [86]: court_df.tail()
```

Out[86]:

	법정동코드	법정동명	폐지여부
46175	5013032022	제주특별자치도 서귀포시 표선면 하천리	존재
46176	5013032023	제주특별자치도 서귀포시 표선면 성읍리	존재
46177	5013032024	제주특별자치도 서귀포시 표선면 가시리	존재
46178	5013032025	제주특별자치도 서귀포시 표선면 세화리	존재
46179	5013032026	제주특별자치도 서귀포시 표선면 토산리	존재

맨뒤에 5개의 값을 가져온다.

info()

```
In [87]: court_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46180 entries, 0 to 46179
Data columns (total 3 columns):
법정동코드    46180 non-null int64
법정동명     46180 non-null object
폐지여부     46180 non-null object
dtypes: int64(1), object(2)
memory usage: 1.1+ MB
```

데이터의 전반적인 내용을 보여준다.

예제

1. 폐지 여부가 존재인 것들만 데이터 프레임으로 만든다.

```
In [91]: # 1. 폐지 여부가 존재인 것들만 데이터 프레임으로 만든다.
court_df[court_df['폐지여부'] == '존재']
```

Out[91]:

	법정동코드	법정동명	폐지여부
0	1100000000	서울특별시	존재
1	1111000000	서울특별시 종로구	존재
2	1111010100	서울특별시 종로구 청운동	존재
3	1111010200	서울특별시 종로구 신교동	존재
4	1111010300	서울특별시 종로구 궁정동	존재
5	1111010400	서울특별시 종로구 효자동	존재
6	1111010500	서울특별시 종로구 창성동	존재
7	1111010600	서울특별시 종로구 통의동	존재

str 를 통한 문자열 접근

In [103]:
```
# 법정동명 앞 5자리까지만 추출
court_df['법정동명'].str[:5].head()
```

Out[103]:
```
0    서울특별시
1    서울특별시
2    서울특별시
3    서울특별시
4    서울특별시
Name: 법정동명, dtype: object
```

. str을 통해 해당하는 부분의 문자열에 접근해 인덱싱을 통해 내가 원하는 식으로 출력할 수 있다.

In [104]:
```
# 법정동명 마지막 한글자만 추출
court_df['법정동명'].str[-1].head()
```

Out[104]:
```
0    시
1    구
2    동
3    동
4    동
Name: 법정동명, dtype: object
```

In [111]:
```
subset_df['법정동명'].str.split(" ", expand=True).head()
```

Out[111]:

	0	1	2	3	4
0	서울특별시	None	None	None	None
1	서울특별시	종로구	None	None	None
2	서울특별시	종로구	청운동	None	None
3	서울특별시	종로구	신교동	None	None
4	서울특별시	종로구	궁정동	None	None

특정 글자로 시작하는 startswith()

In [112]:
```
# 특정 글자로 시작하는 startswith()
# 서울로 시작하는 데이터만 필터링 한다면?
```

In [117]:
```
subset_df[subset_df['법정동명'].str.startswith("서울")]
```

Out[117]:

	법정동코드	법정동명	폐지여부
0	1100000000	서울특별시	존재
1	1111000000	서울특별시 종로구	존재
2	1111010100	서울특별시 종로구 청운동	존재
3	1111010200	서울특별시 종로구 신교동	존재
4	1111010300	서울특별시 종로구 궁정동	존재
5	1111010400	서울특별시 종로구 효자동	존재
6	1111010500	서울특별시 종로구 창성동	존재
7	1111010600	서울특별시 종로구 통의동	존재
8	1111010700	서울특별시 종로구 적선동	존재
9	1111010800	서울특별시 종로구 통인동	존재
10	1111010900	서울특별시 종로구 누상동	존재
11	1111011000	서울특별시 종로구 누하동	존재

특정 글자로 끝나는 endswith()

```
In [118]: subset_df[subset_df['법정동명'].str.endswith("동")]
```

Out[118]:

	법정동코드	법정동명	폐지여부
2	1111010100	서울특별시 종로구 청운동	존재
3	1111010200	서울특별시 종로구 신교동	존재
4	1111010300	서울특별시 종로구 궁정동	존재
5	1111010400	서울특별시 종로구 효자동	존재
6	1111010500	서울특별시 종로구 창성동	존재
7	1111010600	서울특별시 종로구 통의동	존재
8	1111010700	서울특별시 종로구 적선동	존재
9	1111010800	서울특별시 종로구 통인동	존재
10	1111010900	서울특별시 종로구 누상동	존재
11	1111011000	서울특별시 종로구 누하동	존재
12	1111011100	서울특별시 종로구 옥인동	존재
13	1111011200	서울특별시 종로구 체부동	존재
14	1111011300	서울특별시 종로구 필운동	존재
15	1111011400	서울특별시 종로구 내자동	존재
16	1111011500	서울특별시 종로구 사직동	존재
17	1111011600	서울특별시 종로구 도렴동	존재
18	1111011700	서울특별시 종로구 당주동	존재
19	1111011800	서울특별시 종로구 내수동	존재
23	1111012200	서울특별시 종로구 정진동	존재

#특정 글자를 포함하는 데이터만 필터링 : str.contains()

```
In [121]: subset_df[subset_df['법정동명'].str.contains("강서구")]
```

Out[121]:

	법정동코드	법정동명	폐지여부
737	1150000000	서울특별시 강서구	존재
740	1150010100	서울특별시 강서구 염창동	존재
741	1150010200	서울특별시 강서구 등촌동	존재
742	1150010300	서울특별시 강서구 화곡동	존재
743	1150010400	서울특별시 강서구 가양동	존재
744	1150010500	서울특별시 강서구 마곡동	존재
745	1150010600	서울특별시 강서구 내발산동	존재
746	1150010700	서울특별시 강서구 외발산동	존재
747	1150010800	서울특별시 강서구 공항동	존재
748	1150010900	서울특별시 강서구 방화동	존재
749	1150011000	서울특별시 강서구 개화동	존재
750	1150011100	서울특별시 강서구 과해동	존재
751	1150011200	서울특별시 강서구 오곡동	존재
752	1150011300	서울특별시 강서구 오쇠동	존재
2792	2644000000	부산광역시 강서구	존재
2793	2644010100	부산광역시 강서구 대저1동	존재

특정 문자(공백)를 다른 문자(_)로 대체 : str.replace()

```
In [124]: subset_df['법정동명'].str.replace(" ","_")
```

Out[124]:

```
0                    서울특별시
1                 서울특별시_종로구
2              서울특별시_종로구_청운동
3              서울특별시_종로구_신교동
4              서울특별시_종로구_궁정동
5              서울특별시_종로구_효자동
6              서울특별시_종로구_창성동
7              서울특별시_종로구_통의동
8              서울특별시_종로구_적선동
9              서울특별시_종로구_통인동
10             서울특별시_종로구_누상동
11             서울특별시_종로구_누하동
12             서울특별시_종로구_옥인동
13             서울특별시_종로구_체부동
14             서울특별시_종로구_필운동
15             서울특별시_종로구_내자동
16             서울특별시_종로구_사직동
17             서울특별시_종로구_도렴동
18             서울특별시_종로구_당주동
19             서울특별시_종로구_내수동
20             서울특별시_종로구_세종로
21          서울특별시_종로구_신문로1가
22          서울특별시_종로구_신문로2가
23             서울특별시_종로구_청진동
24             서울특별시_종로구_서린동
25             서울특별시_종로구_수송동
26             서울특별시_종로구_중학동
27          서울특별시_종로구_종로1가
```

공백제거(strip, lstrip, rstrip)

```
In [129]: empty_df
```

Out[129]:

	col01	col02
0	abcd	fgHAij
1	FFFght	fhhij
2	abCCe	lmnop

strip() - 전체 공백 제거

```
In [134]: empty_df['col01'].str.strip()
Out[134]: 0      abcd
          1    FFFght
          2     abCCe
          Name: col01, dtype: object
```

lstrip() - 왼쪽 공백 제거

```
In [137]: empty_df['col01'].str.lstrip()
Out[137]: 0    abcd
          1     FFFght
          2    abCCe
          Name: col01, dtype: object
```

rstrip() - 오른쪽 공백 제거

```
In [137]: empty_df['col01'].str.lstrip()
Out[137]: 0    abcd
          1     FFFght
          2    abCCe
          Name: col01, dtype: object
```

대문자 , 소문자 변환 (upper, lower, swapcase)

lower() - 소문자 변환

```
In [140]: empty_df['col02'].str.lower()
Out[140]: 0      fghaij
          1       fhhij
          2       imnop
          Name: col02, dtype: object
```

lower() - 대문자 변환

```
In [142]: empty_df['col02'].str.upper()
Out[142]: 0      FGHAIJ
          1       FHHIJ
          2       IMNOP
          Name: col02, dtype: object
```

lower() - 대소문자 변환

```
In [144]: empty_df['col02'].str.swapcase()
Out[144]: 0      FGhaIJ
          1       FHHIJ
          2       iMNOP
          Name: col02, dtype: object
```

실습

```
In [146]: court_df = pd.read_csv('weather_20201012.csv',sep=',',encoding='cp949')
```

```
In [148]: court_df
```

Out[148]:

	날짜	지점	평균기온(℃)	최저기온(℃)	최고기온(℃)
0	1907-10-01	108	13.5	7.9	20.7
1	1907-10-02	108	16.2	7.9	22.0
2	1907-10-03	108	16.2	13.1	21.3
3	1907-10-04	108	16.5	11.2	22.0
4	1907-10-05	108	17.6	10.9	25.4
5	1907-10-06	108	13.0	11.2	21.3
6	1907-10-07	108	11.3	6.3	16.1
7	1907-10-08	108	8.9	3.9	14.9

In [149]:
```
# 위 데이터에서 기온이 가장 높았던 날은 언제이고 몇도인지를 데이터 프레임으로 출력해보자
```

In [156]:
```
court_df[court_df['최고기온(℃)'] == court_df['최고기온(℃)'].max()]
```

Out[156]:

	날짜	지점	평균기온(℃)	최저기온(℃)	최고기온(℃)
40051	2018-08-01	108	33.6	27.8	39.6

정렬 : sort_values(by= , ascending=)

타입변환 : astype(type)

In [162]:
```
# 오름차순 정렬
sort_df = name_df.sort_values(by='COUNT', ascending=True).head(10)
sort_df
```

Out[162]:

	NAME	GENDER	COUNT
16918	Adwoa	F	5
18460	Lakisha	F	5
18461	Lalena	F	5
18462	Lalla	F	5
18463	Lamarria	F	5
18464	Lamoni	F	5
18465	Lamonica	F	5
18459	Lakhia	F	5
18466	Lamora	F	5
18468	Lanayja	F	5

In [163]:
```
# 내림차순 정렬
sort_df = name_df.sort_values(by='COUNT', ascending=False).head(10)
sort_df
```

Out[163]:

	NAME	GENDER	COUNT
0	Isabella	F	22731
19698	Jacob	M	21875
1	Sophia	F	20477
19699	Ethan	M	17866
2	Emma	F	17179
19700	Michael	M	17133
19701	Jayden	M	17030
19702	William	M	16870
3	Olivia	F	16860
19703	Alexander	M	16634

#rank

rank를 통해 순위를 매겨준다.

```
In [171]: rank_df = name_df.sort_values(by='COUNT', ascending=False)['COUNT'].rank(ascending=False)
```

```
In [175]: rank = rank_df.astype('int64')
          rank

          sort_df['RANK'] = rank
```

```
In [176]: sort_df
```

Out[176]:

	NAME	GENDER	COUNT	RANK
0	Isabella	F	22731	1
19698	Jacob	M	21875	2
1	Sophia	F	20477	3
19699	Ethan	M	17866	4
2	Emma	F	17179	5
19700	Michael	M	17133	6
19701	Jayden	M	17030	7
19702	William	M	16870	8
3	Olivia	F	16860	9
19703	Alexander	M	16634	10

rank 줬던 것의 타입을 astype 을통해 바꿔주고 해당 rank를 기존에 있던 sort_df에 새로운 칼럼명으로
값을 넣어준다. 그러면 위에화 같이 count에 따라 순위가 매겨진 걸 볼 수 있다.

#실습

```
In [179]: # gender 를 기준으로 M 데이터 프레임을 만들어라
          # gender 를 기준으로 F 데이터 프레임을 만들어라
```

```
In [184]: m_df = sort_df[sort_df['GENDER']=='M']
```

```
In [185]: f_df = sort_df[sort_df['GENDER']=='F']
```

In [187]:
```
m_df
```

Out[187]:

	NAME	GENDER	COUNT	RANK
19698	Jacob	M	21875	2
19699	Ethan	M	17866	4
19700	Michael	M	17133	6
19701	Jayden	M	17030	7
19702	William	M	16870	8
19703	Alexander	M	16634	10

In [188]:
```
f_df
```

Out[188]:

	NAME	GENDER	COUNT	RANK
0	Isabella	F	22731	1
1	Sophia	F	20477	3
2	Emma	F	17179	5
3	Olivia	F	16860	9

In [189]:
```
gender_df = pd.merge(m_df,f_df)
gender_df
```

Out[189]:

NAME	GENDER	COUNT	RANK

이 상황에서는 합쳐지지 않는다. 배열의 인덱스가 다르기 때문이다.

In [192]:
```
m_df = m_df.reset_index(drop=True)
```

그래서 m_df의 index를 초기화해준다

In [193]:
```
gender_df = pd.merge(m_df, f_df, left_index = True, right_index=True)
gender_df
```

Out[193]:

	NAME_x	GENDER_x	COUNT_x	RANK_x	NAME_y	GENDER_y	COUNT_y	RANK_y
0	Jacob	M	21875	2	Isabella	F	22731	1
1	Ethan	M	17866	4	Sophia	F	20477	3
2	Michael	M	17133	6	Emma	F	17179	5
3	Jayden	M	17030	7	Olivia	F	16860	9

그 후 다 시 합치면 정상적으로 합쳐진다.

In [195]:
```
rank = gender_df['COUNT_x'].rank(ascending=False)
rank = rank.astype('int64')
gender_df['RANK'] = rank
```

In [196]:
```
gender_df
```

Out[196]:

	NAME_x	GENDER_x	COUNT_x	RANK_x	NAME_y	GENDER_y	COUNT_y	RANK_y	RANK
0	Jacob	M	21875	2	Isabella	F	22731	1	1
1	Ethan	M	17866	4	Sophia	F	20477	3	2
2	Michael	M	17133	6	Emma	F	17179	5	3
3	Jayden	M	17030	7	Olivia	F	16860	9	4

마지막으로 rank를 통해 순위를 매기고 그것을 새로운 변수에 넣어준다.

'Python' 카테고리의 다른 글□

[Python] Pandas 사용법 - 다양한 인덱스 함수(reset_index,set_index,sort_index)□

[Python] Pandas 사용법 - 인덱싱 접근,데이터 조작, 인덱스조작(loc,iloc)□

[Python] Pandas 사용법 - 다양한 함수 사용(데이터 입출력, 대소문자변환, 공...

[Python] Pandas 사용법 - DataFrame 생성, 추가 , 수정, 삭제, indexing□

[Python] Pandas 사용법 - series 에 대한 추가 , 수정, 삭제, 연산, 결측치□

[Python] Pandas의 이론과 기초적인 사용법□

pandas 공백제거 pandas 대소문자 pandas 데이터 입출력 pandas 문자열 접근

나무늘보스

혼자 끄적끄적하는 블로그 입니다.