IT창업개론

4주차 - 판다스 자료형, 문자열 처리하기

대용량 데이터 처리하기

- 뉴욕 택시 데이터는 13억 대의 뉴욕 택시에 대한 정보를 가지고 있음
  - 파일의 개수: 140개

```
import os
import urllib.request

# 네트워크 상태에 따라 5 ~ 15분이 소요됩니다.
with open('../data/raw_data_urls.txt', 'r') as data_urls:

    for line, url in enumerate(data_urls):
        if line == 5:
            break

        fn = url.split('/')[-1].strip()
        fp = os.path.join('', '../data', fn)
        print(url)
        print(fp)
        urllib.request.urlretrieve(url, fp)
```

```
https://s3.amazonaws.com/nyc-tlc/trip+data/fhv_tripdata_2015-01.csv

../data\fhv_tripdata_2015-01.csv
https://s3.amazonaws.com/nyc-tlc/trip+data/fhv_tripdata_2015-02.csv

../data\fhv_tripdata_2015-02.csv
https://s3.amazonaws.com/nyc-tlc/trip+data/fhv_tripdata_2015-03.csv

../data\fhv_tripdata_2015-03.csv
https://s3.amazonaws.com/nyc-tlc/trip+data/fhv_tripdata_2015-04.csv

../data\fhv_tripdata_2015-04.csv
https://s3.amazonaws.com/nyc-tlc/trip+data/fhv_tripdata_2015-05.csv

../data\fhv_tripdata_2015-05.csv
```

대용량 데이터 처리하기

- glob 라이브러리의 glob 함수는 특정한 패턴의 이름을 가진 파일을 한 번에 읽어 들일 수 있음

```
import glob
nyc_taxi_data = glob.glob('../data/fhv_*')
nyc_taxi_data
```

```
['../data₩₩fhv_tripdata_2015-01.csv',
 '../data₩₩fhv_tripdata_2015-02.csv',
 '../data₩₩fhv_tripdata_2015-03.csv',
 '../data₩₩fhv_tripdata_2015-04.csv',
 '../data₩₩fhv_tripdata_2015-05.csv']
```

```
taxi1 = pd.read_csv(nyc_taxi_data[0])
taxi2 = pd.read_csv(nyc_taxi_data[1])
taxi3 = pd.read_csv(nyc_taxi_data[2])
taxi4 = pd.read_csv(nyc_taxi_data[3])
taxi5 = pd.read_csv(nyc_taxi_data[4])
```

```
print(taxi1.head(n=2))
print(taxi2.head(n=2))
print(taxi3.head(n=2))
print(taxi4.head(n=2))
print(taxi5.head(n=2))
```

```
  Dispatching_base_num          Pickup_date  locationID
0               B00013  2015-01-01 00:30:00         NaN
1               B00013  2015-01-01 01:22:00         NaN
  Dispatching_base_num          Pickup_date  locationID
0               B00013  2015-02-01 00:00:00         NaN
1               B00013  2015-02-01 00:01:00         NaN
  Dispatching_base_num          Pickup_date  locationID
0               B00029  2015-03-01 00:02:00       213.0
1               B00029  2015-03-01 00:03:00        51.0
  Dispatching_base_num          Pickup_date  locationID
0               B00001  2015-04-01 04:30:00         NaN
1               B00001  2015-04-01 06:00:00         NaN
  Dispatching_base_num          Pickup_date  locationID
0               B00001  2015-05-01 04:30:00         NaN
1               B00001  2015-05-01 05:00:00         NaN
```

대용량 데이터 처리하기

```
print(taxi1.shape)
print(taxi2.shape)
print(taxi3.shape)
print(taxi4.shape)
print(taxi5.shape)
```

```
(2746033, 3)
(3126401, 3)
(3281427, 3)
(3917789, 3)
(4296067, 3)
```

```
taxi = pd.concat([taxi1, taxi2, taxi3, taxi4, taxi5])
taxi.shape
```

```
(17367717, 3)
```

대용량 데이터 처리하기

- 반복문으로 데이터 준비하기

```
list_taxi_df = []

for csv_filename in nyc_taxi_data:
    # print(csv_filename)
    df = pd.read_csv(csv_filename)
    list_taxi_df.append(df)

len(list_taxi_df)
```

```
5
```

```
type(list_taxi_df[0])
```

```
pandas.core.frame.DataFrame
```

```
list_taxi_df[0].head()
```

	Dispatching_base_num	Pickup_date	locationID
0	B00013	2015-01-01 00:30:00	NaN
1	B00013	2015-01-01 01:22:00	NaN
2	B00013	2015-01-01 01:23:00	NaN
3	B00013	2015-01-01 01:44:00	NaN
4	B00013	2015-01-01 02:00:00	NaN

```
taxi_loop_concat = pd.concat(list_taxi_df)
taxi_loop_concat.shape
```

```
(17367717, 3)
```

판다스 자료형

자료형 변환하기

- 여러가지 자료형을 문자열로 변환하기
  - astype() 사용해 sex 열의 데이터를 문자열로 변환

```
tips['sex']
```

```
0       Female
1         Male
2         Male
3         Male
4       Female
         ...
239       Male
240     Female
241       Male
242       Male
243     Female
Name: sex, Length: 244, dtype: category
Categories (2, object): [Male, Female]
```

```
import pandas as pd
import seaborn as sns
tips = sns.load_dataset("tips")
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
tips['sex_str'] = tips['sex'].astype(str)
tips.dtypes
```

```
total_bill     float64
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
sex_str         object
dtype: object
```

자료형 변환하기

- total_bill 열의 자료형을 문자열로 변경 후 다시 실수로 변환

```
tips['total_bill'].dtype
```

```
dtype('float64')
```

```
tips['total_bill'] = tips['total_bill'].astype(str)
tips.dtypes
```

```
total_bill      object
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
dtype: object
```

```
tips['total_bill'] = tips['total_bill'].astype(float)
tips.dtypes
```

```
total_bill     float64
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
dtype: object
```

잘 못 입력한 데이터 처리하기

- total_bill 열의 1, 3, 5, 7 행의 데이터를 'missing'으로 변경

```
tips_sub_miss = tips.head(10)
tips_sub_miss.loc[[1, 3, 5, 7], 'total_bill'] = 'missing'
tips_sub_miss
```

	total_bill	tip	sex	smoker	day	time	size	sex_str
0	16.99	1.01	Female	No	Sun	Dinner	2	Female
1	missing	1.66	Male	No	Sun	Dinner	3	Male
2	21.01	3.50	Male	No	Sun	Dinner	3	Male
3	missing	3.31	Male	No	Sun	Dinner	2	Male
4	24.59	3.61	Female	No	Sun	Dinner	4	Female
5	missing	4.71	Male	No	Sun	Dinner	4	Male
6	8.77	2.00	Male	No	Sun	Dinner	2	Male
7	missing	3.12	Male	No	Sun	Dinner	4	Male
8	15.04	1.96	Male	No	Sun	Dinner	2	Male
9	14.78	3.23	Male	No	Sun	Dinner	2	Male

```
tips_sub_miss.dtypes
```

```
total_bill      object
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
sex_str         object
dtype: object
```

잘 못 입력한 데이터 처리하기

- total_bill 열의 데이터를 실수로 변환하려 하면 오류 발생

```
tips_sub_miss['total_bill'].astype(float)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-3aba35b22fb4> in <module>
----> 1 tips_sub_miss['total_bill'].astype(float)

ValueError: could not convert string to float: 'missing'
```

```
pd.to_numeric(tips_sub_miss['total_bill'])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
pandas/_libs/src\inference.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "missing"
```

잘 못 입력한 데이터 처리하기

- to_numeric()의 errors 인자
  - 기본 설정 값은 raise

raise	숫자로 변환할 수 없는 값이 있으면 오류 발생
coerce	숫자로 변환할 수 없는 값을 누락값으로 지정
ignore	아무 작업도 하지 않음

```
tips_sub_miss.loc[:, 'total_bill'] = pd.to_numeric(tips_sub_miss['total_bill'], errors='ignore')
tips_sub_miss.dtypes
```

```
total_bill      object
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
sex_str         object
dtype: object
```

잘 못 입력한 데이터 처리하기

- to_numeric()의 error 인자

```
tips_sub_miss.loc[:, 'total_bill'] = pd.to_numeric( tips_sub_miss['total_bill'], errors='coerce')
tips_sub_miss.dtypes
```

```
total_bill     float64
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
sex_str         object
dtype: object
```

```
tips_sub_miss
```

	total_bill	tip	sex	smoker	day	time	size	sex_str
0	16.99	1.01	Female	No	Sun	Dinner	2	Female
1	NaN	1.66	Male	No	Sun	Dinner	3	Male
2	21.01	3.50	Male	No	Sun	Dinner	3	Male
3	NaN	3.31	Male	No	Sun	Dinner	2	Male
4	24.59	3.61	Female	No	Sun	Dinner	4	Female
5	NaN	4.71	Male	No	Sun	Dinner	4	Male
6	8.77	2.00	Male	No	Sun	Dinner	2	Male
7	NaN	3.12	Male	No	Sun	Dinner	4	Male
8	15.04	1.96	Male	No	Sun	Dinner	2	Male
9	14.78	3.23	Male	No	Sun	Dinner	2	Male

자료형 변환하기

- to_numeric()의 downcast 인자
  - integer, signed, unsigned, float
  - 정수, 실수와 같은 자료형을 더 작은 형태로 만들 때 사용

```
tips_sub_miss.loc[:, 'total_bill'] = pd.to_numeric( tips_sub_miss['total_bill'], errors='coerce', downcast='float')
tips_sub_miss.dtypes
```

```
total_bill     float32
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
sex_str         object
dtype: object
```

카테고리 자료형

- 용량과 속도 면에서 매우 효율적
- 주로 동일한 문자열이 반복되어 데이터를 구성하는 경우에 사용

sex 열의 데이터는 남자 또는 여자만으로 구성

```
tips['sex'] = tips['sex'].astype('str')
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 8 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null object
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
sex_str       244 non-null object
dtypes: category(3), float64(2), int64(1), object(2)
memory usage: 10.7+ KB
```

```
tips['sex'] = tips['sex'].astype('category')
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 8 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null category
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
sex_str       244 non-null object
dtypes: category(4), float64(2), int64(1), object(1)
memory usage: 9.2+ KB
```

문자열 처리하기

문자열 다루기

- 인덱스로 문자열 추출하기

문자열 grail과 인덱스

인덱스	0	1	2	3	4
문자열	g	r	a	i	l
음수 인덱스	-5	-4	-3	-2	-1

문자열 a scratch와 인덱스

인덱스	0	1	2	3	4	5	6	7	8
문자열	a		s	c	r	a	t	c	h
음수 인덱스	-9	-8	-7	-6	-5	-4	-3	-2	-1

문자열 다루기

- 인덱스로 문자열 추출하기

```
word = 'grail'
sent = 'a scratch'
```

```
word[0]
```

```
'g'
```

```
sent[0]
```

```
'a'
```

```
word[0:3]
```

```
'gra'
```

```
sent[-1]
```

```
h
```

```
sent[-9:-8]
```

```
a
```

```
sent[0:-8]
```

```
a
```

문자열 다루기

- 전체 문자열을 추출할 때 음수를 사용하면?
  - 인덱스 슬라이싱의 오른쪽 범위로 지정한 값은 추출 범위에서 제거됨

```
word = 'grail'
sent = 'a scratch'
```

```
sent[2:-1]
```

```
'scratc'
```

```
sent[-7:-1]
```

```
'scratc'
```

```
s_len = len(sent)
s_len
```

```
9
```

```
sent[2:s_len]
```

```
'scratch'
```

문자열 다루기

- 전체 문자열 추출하기

```
word[0:3]
```

```
gra
```

```
word[ :3]
```

```
gra
```

```
sent[2:len(sent)]
```

```
scratch
```

```
sent[2: ]
```

```
scratch
```

```
sent[ : ]
```

```
a scratch
```

```
sent[::2]
```

```
asrth
```

문자열 함수

capitalize	첫 문자를 대문자로 변환
count	문자열의 개수 반환
startswith	문자열이 특정 문자로 시작하면 참을 반환
endswith	문자열이 특정 문자로 끝나면 참을 반환
find	찾을 문자열의 첫 번째 인덱스 반환, 실패 시 -1 반환
index	find와 같은 역할을 수행하지만, 실패 시 ValueError
isalpha	모든 문자가 알파벳이면 참을 반환
isdecimal	모든 문자가 숫자면 참을 반환
isalnum	모든 문자가 알파벳이거나 숫자면 참을 반환

문자열 함수

lower	모든 문자를 소문자로 변환
upper	모든 문자를 대문자로 변환
replace	문자열의 문자를 다른 문자로 교체
strip	문자열의 맨 앞과 맨 뒤에 있는 빈 칸 제거
split	구분자를 지정하여 문자열을 나누고, 나눈 값들의 리스트 반환
partition	split과 비슷한 역할을 수행하지만 구분자도 반환
center	지정한 너비로 문자열을 늘이고 문자열을 가운데 정렬
zfill	문자열의 빈 칸을 '0'으로 채움

문자열 함수

"black knight".capitalize()	'Black knight'
"It's just a flesh wound!".count('u')	2
"Halt! Who goes there?".startswith('Halt')	True
"coconut".endswith('nut')	True
"It's just a flesh wound!".find('u')	6
"It's just a flesh wound!".index('scratch')	ValueError
"old woman".isalpha()	False
"37".isdecimal()	True
"I'm 37".isalnum()	False

문자열 함수

"Black knight".lower()	'black knight'
"Black knight".upper()	'BLACK KNIGHT'
"flesh wound!".replace('flesh wound', 'scratch')	'scratch!'
" I'm not dead. ".strip()	"I'm not dead."
"NI! NI! NI! NI!".split(sep=' ')	['NI!', 'NI!', 'NI!', 'NI!']
"3,4,5".partition(',')	('3', ',', '4,5')
"nine".center(20)	' nine '
"9".zfill(5)	'00009'

문자열 함수

- join()
  - 문자열을 연결하여 새로운 문자열을 반환
  - 해당 문자열을 단어 사이에 넣어 연결

```
d1 = '40° '
m1 = "46'"
s1 = '52.837"'
u1 = 'N'

d2 = '73° '
m2 = "58'"
s2 = '26.302"'
u2 = 'W'
```

```
' '.join([d1, m1, s1, u1, d2, m2, s2, u2])
```

```
'40°  46′ 52.837" N 73°  58′ 26.302" W'
```

```
'-'.join([d1, m1, s1, u1, d2, m2, s2, u2])
```

```
'40° -46′-52.837"-N-73° -58′-26.302"-W'
```

문자열 함수

- splitlines()
  - 여러 행을 가진 문자열을 분리한 다음 리스트로 반환

```
multi_str = """Guard: What? Ridden on a horse?
King Arthur: Yes!
Guard: You're using coconuts!
King Arthur: What?
Guard: You've got ... coconut[s] and you're bangin' 'em together.
"""
print(multi_str)
```

```
Guard: What? Ridden on a horse?
King Arthur: Yes!
Guard: You're using coconuts!
King Arthur: What?
Guard: You've got ... coconut[s] and you're bangin' 'em together.
```

```
multi_str_split = multi_str.splitlines()
print(multi_str_split)
```

```
['Guard: What? Ridden on a horse?', 'King Arthur: Yes!', "Guard: You're using coconu
ts!", 'King Arthur: What?', "Guard: You've got ... coconut[s] and you're bangin' 'em
together. "]
```

```
guard = multi_str_split[::2]
print(guard)
```

```
['Guard: What? Ridden on a horse?', "Guard: You're using coconuts!", "Guard: You've
got ... coconut[s] and you're bangin' 'em together. "]
```

문자열 함수

- replace()

```
guard = multi_str_split[::2]
print(guard)
```

```
['Guard: What? Ridden on a horse?', "Guard: You're using coconuts!", "Guard: You've got ... coconut[s] and you're bangin' 'em together. "]
```

```
guard = multi_str.replace("Guard: ", "").splitlines()[::2]
print(guard)
```

```
['What? Ridden on a horse?', "You're using coconuts!", "You've got ... coconut[s] and you're bangin' 'em together. "]
```

문자열 Formatting

- I can swim, I can fly, I can run
  - I can이라는 문자열에 swim, fly, run과 같은 단어만 바꿔 넣어 출력하는 것이 편리함
  - 출력할 문자열의 형식을 지정하거나 변수를 조합하여 출력하는 방법

```
var = 'flesh wound'
s = "It's just a {}!"

s.format(var)
```

```
"It's just a flesh wound!"
```

```
s.format('scratch')
```

```
"It's just a scratch!"
```

문자열 Formatting

- 인덱스를 지정한 플레이스 홀더

```
s = """Black Knight: 'Tis but a {0}.
King Arthur: A {0}? Your arm's off!
"""
print(s.format('scratch'))
```

```
Black Knight: 'Tis but a scratch.
King Arthur: A scratch? Your arm's off!
```

```
s = 'I can {0}, I can {1}, I can {2}'
s.format('swim', 'fly', 'run')
```

```
'I can swim, I can fly, I can run'
```

문자열 Formatting

- 변수를 지정한 플레이스 홀더

```
s = 'Hayden Planetarium Coordinates: {lat}, {lon}'
print(s.format(lat='40.7815° N', lon='73.9733° W'))
```

```
Hayden Planetarium Coordinates: 40.7815° N, 73.9733° W
```

숫자 데이터 Formatting

```
print('Some digits of pi: {}'.format(3.14159265359))
```

```
Some digits of pi: 3.14159265359
```

- 플레이스 홀더에 :,를 넣으면 쉼표를 넣어 숫자를 표현할 수도 있음

```
print("In 2005, Lu Chao of China recited {:,} digits of pi".format(67890))
```

```
In 2005, Lu Chao of China recited 67,890 digits of pi
```

숫자 데이터 Formatting

- {0:.4}, {0:.4%}
  - 0: 인덱스
  - .4: 소수점 이하의 숫자를 4개까지 출력
  - %: 백분율로 환산

```
print("I remember {0:.4} or {0:.4%} of what Lu Chao recited".format(0.11223344556677))
```

```
I remember 0.1122 or 11.2233% of what Lu Chao recited
```

숫자 데이터 Formatting

- 5자리의 숫자로 표현되어야 한다면?
  - d는 10진수(decimal)을 의미

```
print("My ID number is {0:05d}".format(42))
```

```
My ID number is 00042
```

% 연산자로 Formatting

```
s = 'I only know %d digits of pi' % 7
print(s)
```

```
I only know 7 digits of pi
```

```
print('Some digits of %(cont)s: %(value).2f' % {'cont': 'e', 'value': 2.718})
```

```
Some digits of e: 2.72
```

f-strings Formatting

- 파이썬 3.6 버전에서 도입
- 문자열 앞에 f를 붙임

```
var = 'flesh wound'
s = f"It's just a {var}!"
print(s)
```

```
It's just a flesh wound!
```

```
lat='40.7815° N'
lon='73.9733° W'
s = f'Hayden Planetarium Coordinates: {lat}, {lon}'
print(s)
```

```
Hayden Planetarium Coordinates: 40.7815° N, 73.9733° W
```

Thank You!