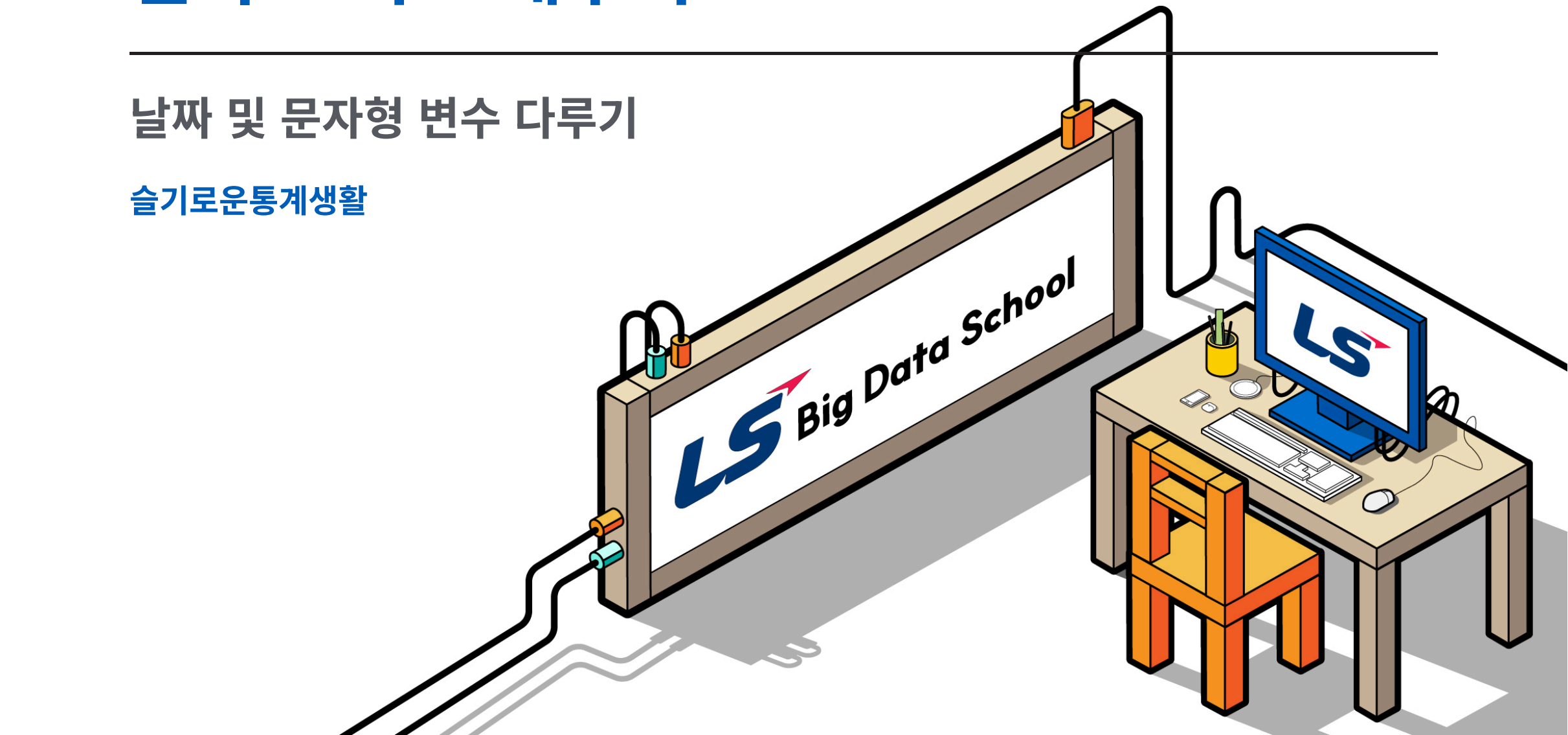
판다스 기초 배우기

날짜 및 문자형 변수 다루기

슬기로운통계생활



코스 훑어보기.

날짜형 변수 다루는 방법에 대해 학습합니다.

날짜와 시간 다루기

판다스는 날짜와 시간 데이터를 다루는 다양한 방법을 제공합니다. 주요 기능으로는 날짜 파싱, 날짜 정보 추출, 날짜 연산 등이 있습니다.

실습을 위해 예제 데이터를 생성해보겠습니다.

```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings("ignore", category=pd.errors.SettingWithCopyWarning)

data = {
    'date': ['2024-01-01 12:34:56', '2024-02-01 23:45:01', '2024-03-01 06:07:08', '2021-04-0
    'value': [100, 201, 302, 404]
}
df = pd.DataFrame(data)
```

데이터 타입 확인

각 칼럼별 데이터 타입을 확인해보겠습니다.

```
print(df.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 4 entries, 0 to 3
## Data columns (total 2 columns):
##  #   Column  Non-Null Count  Dtype 
## ---  ------  --------------  ----- 
##  0   date    4 non-null      object
##  1   value   4 non-null      int64 
## dtypes: int64(1), object(1)
## memory usage: 196.0+ bytes
## None
```

date 칼럼의 경우 object인 것을 확인할 수 있습니다.

날짜 형식으로 변환

날짜 및 시간을 나타내는 칼럼은 날짜 및 시간 처리를 손쉽게 하기 위해 날짜 형식으로 변환하는 것이 좋습니다. `to_datetime()` 을 활용하여 날짜 형식으로 변환해보겠습니다.

```
# 문자열을 날짜 형식으로 변환
df['date'] = pd.to_datetime(df['date'])
print(df.dtypes)
```

```
## date     datetime64[ns]
## value             int64
## dtype: object
```

`datetime64[ns]`으로 변환된 것을 확인할 수 있습니다.

날짜 형식으로 변환

datetime64[ns] 의미

- datetime64: Numpy와 Pandas에서 사용하는 기본 날짜 및 시간 데이터 타입으로 날짜와 시간 정보를 포함합니다.
- ns : ns는 나노초(nanosecond)를 의미하며, 나노초 단위까지 정밀하게 날짜와 시간을 표현할 수 있습니다.

`to_datetime()`은 일반적으로 ISO 8601 형식을 자동으로 인식합니다. 자동 변환되는 표준화된 날짜 형식을 확인해보겠습니다.

날짜 형식으로 변환

입력 날짜 형식	설명	변환된 결과
`2024-01-01`	ISO 8601 기본 형식	`Timestamp('2024-01-01 00:00:00')`
`2024-01-01 12:34:56`	ISO 8601 날짜 및 시간	`Timestamp('2024-01-01 12:34:56')`
`2024-01-01T12:34:56`	ISO 8601 날짜 및 시간 (T 포함)	`Timestamp('2024-01-01 12:34:56')`
`2024-01-01T12:34:56Z`	ISO 8601 날짜, 시간, UTC	`Timestamp('2024-01-01 12:34:56+0000', tz='UTC')`
`01/01/2024`	MM/DD/YYYY 형식	`Timestamp('2024-01-01 00:00:00')`
`01-01-2024`	MM-DD-YYYY 형식	`Timestamp('2024-01-01 00:00:00')`
`31-01-2024`	DD-MM-YYYY 형식	`Timestamp('2024-01-31 00:00:00')`
`January 1, 2024`	월 이름 포함 형식	`Timestamp('2024-01-01 00:00:00')`

날짜 형식 변환시 주의 사항

`to_datetime()`은 비표준화 형식의 날짜 문자열의 경우 날짜 형식으로 자동 변환되지 않습니다. 예시를 통해 확인해보겠습니다.

```
pd.to_datetime('02-2024-01')
```

```
## pandas._libs.tslibs.parsing.DateParseError: day is out of range for month: 02-2024-01, at
```

월-년-일 형식이므로, 자동으로 변환되지 않는 것을 확인할 수 있습니다. `format='%m-%Y-%d'` 옵션을 지정해보겠습니다.

날짜 형식 변환시 주의 사항

```
pd.to_datetime('02-2024-01', format='%m-%Y-%d')
```

```
## Timestamp('2024-02-01 00:00:00')
```

원하는 날짜 형식으로 변환된 것을 확인할 수 있습니다.

날짜 형식 변환시 주의 사항

날짜형 문자열에 한글이 포함된 경우는 어떨까요?

```
pd.to_datetime('2024년 01월 01일')
```

```
## pandas._libs.tslibs.parsing.DateParseError: Unknown datetime string format, unable to pars
```

이전과 마찬가지로, 자동으로 변환되지 않는 것을 확인할 수 있습니다. `format='%Y년 %m월 %d일'` 옵션을 지정해보겠습니다.

날짜 형식 변환시 주의 사항

```
pd.to_datetime('2024년 01월 01일', format='%Y년 %m월 %d일')
```

```
## Timestamp('2024-01-01 00:00:00')
```

원하는 날짜 형식으로 변환된 것을 확인할 수 있습니다.

날짜 정보 추출

date 칼럼을 날짜 형식으로 변환했으므로, 연도, 월, 일, 요일 등 시간 정보를 손쉽게 추출할 수 있습니다.

```
# 연도 추출
df['year'] = df['date'].dt.year

# 월 추출
df['month'] = df['date'].dt.month

# 일 추출
df['day'] = df['date'].dt.day

# 요일 추출
df['wday'] = df['date'].dt.day_name()
df['wday2'] = df['date'].dt.weekday
```

날짜 정보 추출

```
# 시간 추출
df['hour'] = df['date'].dt.hour

# 분 추출
df['minute'] = df['date'].dt.minute

# 초 추출
df['second'] = df['date'].dt.second

# 년-월-일 추출
print(df['date'].dt.date)
```

```
## 0    2024-01-01
## 1    2024-02-01
## 2    2024-03-01
## 3    2021-04-01
## Name: date, dtype: object
```

날짜 정보 추출

```
pd.set_option('display.max_columns', None) # 모든 칼럼 출력
print(df.head(2))
```

```
##                  date  value  year  month  day      wday  wday2  hour  minute  \
## 0 2024-01-01 12:34:56    100  2024      1    1    Monday      0    12      34
## 1 2024-02-01 23:45:01    201  2024      2    1  Thursday      3    23      45
##
##    second
## 0      56
## 1       1
```

날짜 연산

날짜 형식의 데이터를 사용하여 날짜 간의 차이를 계산하거나 날짜를 조작할 수 있습니다.

```
# 현재 날짜 생성
current_date = pd.to_datetime('2024-05-01')

# 날짜 차이 계산
df['days_diff'] = (current_date - df['date']).dt.days

print(df.head(2))
```

```
##                  date  value  year  month  day      wday  wday2  hour  minute  \
## 0 2024-01-01 12:34:56    100  2024      1    1    Monday      0    12      34
## 1 2024-02-01 23:45:01    201  2024      2    1  Thursday      3    23      45
##
##    second  days_diff
## 0      56        120
## 1       1         89
```

날짜 범위 생성

`date_range()`를 사용하여 일정한 간격의 날짜 범위를 생성할 수 있습니다.

```
# 날짜 범위 생성
date_range = pd.date_range(start='2021-01-01', end='2021-01-10', freq='D')

print(date_range)
```

```
## DatetimeIndex(['2021-01-01', '2021-01-02', '2021-01-03', '2021-01-04',
##                '2021-01-05', '2021-01-06', '2021-01-07', '2021-01-08',
##                '2021-01-09', '2021-01-10'],
##               dtype='datetime64[ns]', freq='D')
```

날짜 합치기

년-월-일 칼럼을 활용해서 새롭게 날짜 칼럼을 생성할 수도 있습니다.

```
df['date2'] = pd.to_datetime(dict(year=df.year, month=df.month, day=df.day))

print(df[['date', 'date2']])
```

```
##                  date      date2
## 0 2024-01-01 12:34:56 2024-01-01
## 1 2024-02-01 23:45:01 2024-02-01
## 2 2024-03-01 06:07:08 2024-03-01
## 3 2021-04-01 14:15:16 2021-04-01
```

코스 훑어보기.

문자형 변수 다루는 방법에 대해 학습합니다.

문자 다루기

판다스는 str 접근자와 함께 다양한 메서드를 사용하여 문자열 데이터를 처리할 수 있습니다. 판다스에서 문자열을 다루는 방법을 알아보겠습니다.

실습을 위해 예제 데이터를 생성해보겠습니다.

```
import pandas as pd

data = {
    '가전제품': ['냉장고', '세탁기', '전자레인지', '에어컨', '청소기'],
    '브랜드': ['LG', 'Samsung', 'Panasonic', 'Daikin', 'Dyson']
}
df = pd.DataFrame(data)
```

문자열 길이 확인

`str.len()`을 통해 문자열 길이를 확인할 수 있습니다.

```
# 문자열 길이
df['제품명_길이'] = df['가전제품'].str.len()
df['브랜드_길이'] = df['브랜드'].str.len()

print(df.head(2))
```

```
##    가전제품       브랜드  제품명_길이  브랜드_길이
## 0   냉장고        LG        3         2
## 1   세탁기   Samsung        3         7
```

문자열 대소문자 변환

`str.lower()`, `str.upper()`를 통해 문자열의 대소문자를 자유롭게 변환할 수 있습니다.

```
# 대소문자 변환
df['브랜드_소문자'] = df['브랜드'].str.lower()
df['브랜드_대문자'] = df['브랜드'].str.upper()
df['브랜드_타이틀'] = df['브랜드'].str.title()

print(df[['브랜드', '브랜드_소문자', '브랜드_대문자', '브랜드_타이틀']].head(2))
```

```
##         브랜드  브랜드_소문자  브랜드_대문자  브랜드_타이틀
## 0        LG        lg        LG        Lg
## 1   Samsung   samsung   SAMSUNG   Samsung
```

특정 문자 포함 여부 확인

`str.contains()`을 통해 특정 문자 포함 여부를 확인할 수 있습니다. 특정 문자가 포함된 경우 True, 포함되지 않을 경우 False로 인식합니다.

```
# 문자열 포함 여부
df['브랜드에_a포함'] = df['브랜드'].str.contains('a')

print(df[['브랜드', '브랜드에_a포함']].head(2))
```

```
##         브랜드  브랜드에_a포함
## 0        LG     False
## 1   Samsung      True
```

특정 문자열 교체

`str.replace()`을 통해 특정 문자열을 교체할 수 있습니다.

```
# 문자열 교체
df['브랜드_언더스코어'] = df['브랜드'].str.replace('L', 'HHHHG')

print(df[['브랜드', '브랜드_언더스코어']].head(2))
```

```
##          브랜드 브랜드_언더스코어
## 0         LG     HHHHGG
## 1   Samsung    Samsung
```

문자열 분할

`str.split()`을 통해 문자열을 분할할 수 있습니다. `expand = True`로 설정하면 분할 결과를 여러 칼럼으로 확장합니다. `expand = False`로 설정할 경우 리스트의 시리즈를 반환합니다.

```
# 문자열 분할
df[['브랜드_첫부분', '브랜드_두번째', '브랜드_세번째']] = df['브랜드'].str.split('a', expand=True)

print(df[['브랜드', '브랜드_첫부분', '브랜드_두번째', '브랜드_세번째']].head(2))
```

```
##         브랜드 브랜드_첫부분 브랜드_두번째 브랜드_세번째
## 0        LG       LG     None     None
## 1   Samsung        S    msung     None
```

문자열 결합

`str.cat()`을 통해 문자열을 결합할 수 있습니다.

```
# 문자열 결합
df['제품_브랜드'] = df['가전제품'].str.cat(df['브랜드'], sep=', ')

print(df[['가전제품', '제품_브랜드']].head(2))
```

```
##    가전제품          제품_브랜드
## 0   냉장고        냉장고, LG
## 1   세탁기   세탁기, Samsung
```

문자열 공백 제거

`str.strip()`을 통해 문자열 앞 뒤 공백을 제거할 수 있습니다.

```
# 문자열 앞뒤 공백 제거
df['가전제품'] = df['가전제품'].str.replace('전자레인지', '  전자 레인지   ')

df['가전제품_공백제거'] = df['가전제품'].str.strip()

print(df[['가전제품', '가전제품_공백제거']].head(2))
```

```
##    가전제품 가전제품_공백제거
## 0   냉장고         냉장고
## 1   세탁기         세탁기
```

정규표현식을 통한 문자열 추출

정규 표현식(Regular Expressions, regex)은 특정한 규칙을 가진 문자열의 패턴을 정의하는 데 사용되며, 문자열 검색, 치환, 추출 등 다양한 문자열 처리 작업에 활용됩니다.

대표적인 정규표현식은 다음과 같습니다.

정규표현식	설명	매칭 예시
`[]`	대괄호 안에 있는 문자 중 하나와 매칭	`[aeiou]`는 'a', 'e', 'i', 'o', 'u' 중 하나에 매칭
`()`	소괄호 안에 있는 패턴을 하나의 그룹으로 취급	`(abc)`는 'abc'라는 문자열에 매칭
`.`	임의의 한 문자에 매칭	`a.c`는 'abc', 'a1c', 'a-c' 등
`^`	문자열의 시작에 매칭	`^Hello`는 'Hello world'에서 'Hello'와 매칭
`$`	문자열의 끝에 매칭	`world$`는 'Hello world'에서 'world'와 매칭

정규표현식을 통한 문자열 추출

정규표현식	설명	매칭 예시
`\d`	숫자에 매칭	`\d`는 '0', '1', '2', ... '9'와 매칭
`\D`	숫자가 아닌 문자에 매칭	`\D`는 'a', 'b', '!', ' ' 등 숫자가 아닌 문자
`\w`	단어 문자 (알파벳, 숫자, 밑줄)에 매칭	`\w`는 'a', '1', '_'와 매칭
`\W`	단어 문자가 아닌 문자에 매칭	`\W`는 '!', '\@', ' ' 등
`\s`	공백 문자 (스페이스, 탭, 개행 문자 등)에 매칭	`\s`는 ' ', '\\t', '\\n'와 매칭
`\S`	공백 문자가 아닌 문자에 매칭	`\S`는 'a', '1', '!', '\@' 등

정규표현식을 통한 문자열 추출

정규표현식	설명	매칭.예시
\b	단어 경계 (word boundary)에 매칭	\bHello\b는 'Hello world'에서 'Hello'와 매칭
\B	단어 경계가 아닌 곳에 매칭	\Bend는 'end'에서 'end'와 매칭
[Hh]i	Hi나 hi에 매칭되는 문자열 반환	Hi there, hi there
(Hi|Hello)	Hi나 Hello에 매칭되는 문자열 반환	Hi there, Hello world
(Hi|Hello)|(And)	Hi와 Hello는 Group1에, And는 Group2에 매칭됨	Hi there, Hello world, And also
bl(e|u)e	blee나 blue 모두 매칭되어 반환	blee, blue
[a-zA-Z0-9]	a~z, A~Z, 0~9 사이의 문자나 숫자	Hi there, Hello world, blue, bllue 등

정규표현식을 통한 문자열 추출

정규표현식	설명	매칭 예시
`[^a-zA-Z0-9]`	영어와 숫자 빼고 모든 문자를 반환	`'!'`, `'@'`, `' '` 등 특수 문자
`[가-힣]`	모든 한글 문자에 매칭	가, 핫, 힣
`bl?ue`	`blue` 또는 `bue` 모두 반환	`blue`, `bue`
`bl*ue`	`blue`, `bllee`, `bue` 모두 반환	`blue`, `bllee`, `bue`
`bl+ue`	`blue`, `bllue`, `blllue` 모두 반환	`blue`, `bllue`, `blllue`
`bl{2}ue`	`bllue`만 반환	`bllue`
`bl{2,3}ue`	`bllue`, `blllue`를 반환	`bllue`, `blllue`
`bl{2,}ue`	`bllue`, `blllue`를 반환 (최소 2개 이상)	`bllue`, `blllue`

실습 데이터 생성

실습을 위해 예제 데이터를 생성하겠습니다.

```
data = {
    '주소': ['서울특별시 강남구 테헤란로 123', '부산광역시 해운대구 센텀중앙로 45', '대구광역시 수성구 동대구로 77-
}
df = pd.DataFrame(data)

print(df.head(2))
```

```
##                          주소
## 0    서울특별시 강남구 테헤란로 123
## 1  부산광역시 해운대구 센텀중앙로 45
```

특정 문자열 추출

주소에서 특정 도시명(예 : 광역시, 특별시)을 추출해보겠습니다.

`([가-힣]+광역시|[가-힣]+특별시)` 구성 요소

- `[가-힣]` : 모든 한글 문자
- `[가-힣]+` : 한글 문자가 1회 이상 반복
    - 예시 : 서울, 부산, 대구 등
- `[가-힣]+광역시` : 한글 문자가 1회 이상 반복되고, 광역시로 끝나는 문자열 반환

`([가-힣]+광역시|[가-힣]+특별시)`은 광역시 또는 특별시로 끝나는 모든 문자열을 의미합니다.

특정 문자열 추출

`str.extract()`를 활용하면 정규표현식으로 매칭되는 패턴 중 첫 번째 값을 추출할 수 있습니다.

```
df['도시'] = df['주소'].str.extract(r'([가-힣]+광역시|[가-힣]+특별시)', expand=False)
print(df.head(2))
```

```
##                          주소      도시
## 0   서울특별시 강남구 테헤란로 123  서울특별시
## 1  부산광역시 해운대구 센텀중앙로 45  부산광역시
```

특수 문자 추출

주소에서 모든 특수 문자를 추출해보겠습니다. `str.extractall()`을 통해 정규표현식으로 매칭되는 패턴 중 모든 값을 추출할 수 있습니다.

```
# 모든 특수 문자 추출
special_chars = df['주소'].str.extractall(r'([^a-zA-Z0-9가-힣\s])')
print(special_chars)
```

```
##           0
##    match
## 2 0       -
##   1       @
##   2       @
##   3       #
##   4       #
## 3 0       &
##   1       &
##   2       ,
```

특수 문자 제거

정규표현식을 활용하면 특수 문자를 손쉽게 제거할 수 있습니다. `str.replace()` 메서드에 `regex=True` 옵션을 활용하면 정규 표현식을 적용할 수 있습니다.

`[^a-zA-Z0-9가-힣\s]` 구성 요소

- `^` : 대괄호 [] 내에서 사용되면 “부정”을 의미하며, 해당 패턴에 포함되지 않는 문자를 의미함
- `a-z` : 소문자 알파벳 a부터 z까지
- `A-Z` : 대문자 알파벳 A부터 Z까지
- `0-9` : 숫자 0부터 9까지
- `가-힣` : 한글 음절을 나타내는 문자 범위
- `\s` : 공백 문자(스페이스, 탭, 개행 문자 등)

`[^a-zA-Z0-9가-힣\s]`은 알파벳 소문자와 대문자, 숫자, 한글, 공백 문자가 아닌 모든 문자를 의미합니다.

특수 문자 제거

```
# 특수 문자 제거
df['주소_특수문자제거'] = df['주소'].str.replace(r'[^a-zA-Z0-9가-힣\s]', '', regex=True)
print(df.head(2))
```

```
##                          주소      도시              주소_특수문자제거
## 0   서울특별시 강남구 테헤란로 123  서울특별시  서울특별시 강남구 테헤란로 123
## 1  부산광역시 해운대구 센텀중앙로 45  부산광역시  부산광역시 해운대구 센텀중앙로 45
```

정규표현식 파이썬 문법

- 정규표현식 앞 부분에 삽입하는 `r`은 "raw string" 을 나타냅니다.
- 파이썬에서는 백슬래시(\)가 문자열에서 특별한 의미를 가지기 때문에, 이를 회피하기 위해 raw string을 사용합니다.
- r을 앞에 붙이면, 백슬래시를 그대로 해석합니다.
    - r'\n'은 줄바꿈 문자가 아닌 두 개의 문자 \와 n을 나타냄

연습 문제

공유 자전거 데이터

데이터셋은 런던 공유 자전거 시스템의 대여 기록을 다루고 있으며, 대여 및 반납 정보, 날씨 정보, 시간대 등의 다양한 특성(features)을 포함하고 있습니다.

- datetime: 날짜 및 시간 정보
- season: 계절 (1: 봄, 2: 여름, 3: 가을, 4: 겨울)
- holiday: 공휴일 여부 (0: 공휴일 아님, 1: 공휴일)
- workingday: 평일 여부 (0: 주말 또는 공휴일, 1: 평일)
- weather: 날씨 상황 (1: 맑음, 2: 흐림, 3: 약간의 눈/비, 4: 폭우/폭설)
- temp: 기온 (섭씨)
- atemp: 체감 온도 (섭씨)
- humidity: 습도 (%)
- windspeed: 풍속 (m/s)
- casual: 비회원 대여 수
- registered: 회원 대여 수
- count: 총 대여 수

데이터 불러오기

```
df = pd.read_csv(r'~/Desktop/슬기로운통계생활/lsbigdata/본강의자료/data/bike_data.csv')
print(df.head())
```

```
##              datetime  season  holiday  workingday  weather   temp   atemp  \
## 0 2011-09-05 17:00:00       3        1           0        2  27.06  29.545
## 1 2011-05-17 11:00:00       2        0           1        2  22.96  26.515
## 2 2011-11-10 09:00:00       4        0           1        2  17.22  21.210
## 3 2011-10-13 07:00:00       4        0           1        3  22.14  25.760
## 4 2011-10-15 14:00:00       4        0           0        1  24.60  31.060
##
##    humidity  windspeed  casual  registered  count
## 0        89     7.0015      37          77    114
## 1        83    27.9993      26         104    130
## 2        94     7.0015      23         188    211
## 3       100     8.9981       5          76     81
## 4        33    31.0009     242         230    472
```

데이터 속성 변환

칼럼에 대한 설명을 확인한 후 데이터 형식을 적절하게 변경하겠습니다.

```
df = df.astype({'datetime' : 'datetime64[ns]', 'weather' : 'int64', 'season' : 'object', 'wo
```

계절(season) == 1일 때, 가장 대여량이 많은 시간대(hour)을 구하시오.

```
## count가 가장 큰 hour는 17시이며, 대여량은 970입니다.
```

각 계절(season)별 평균 대여량(count)을 구하시오.

```
##    season       count
## 0       1  103.169811
## 1       2  218.803922
## 2       3  265.500000
## 3       4  218.581197
```

특정 달(month) 동안의 총 대여량(count)을 구하시오.

```
## 1월 동안의 총 대여량은 2567입니다.
```

가장 대여량이 많은 날짜를 구하시오.

```
## 가장 대여량이 많은 날짜는 2012-05-11이며, 대여량은 1398입니다.
```

시간대(hour)별 평균 대여량(count)을 구하시오.

```
##    hour       count
## 0     0  43.500000
## 1     1  52.714286
## 2     2  32.842105
## 3     3  12.000000
## 4     4   6.687500
```

특정 요일(weekday) 동안의 총 대여량(count)을 구하시오.

```
## 월요일 동안의 총 대여량은 10191입니다.
```

주어진 Bike Sharing 데이터를 사용하여 넓은 형식(wide format)에서 긴 형식(long format)으로 변환하시오. casual과 registered 열을 하나의 열로 변환하고, 각 기록의 대여 유형과 대여 수를 포함하는 긴 형식 데이터프레임을 만드시오.

```
##
## melt를 사용하여 변환된 데이터프레임:
```

```
##                 datetime season   user_type  user_count
## 0    2011-09-05 17:00:00      3      casual          37
## 1    2011-05-17 11:00:00      2      casual          26
## 2    2011-11-10 09:00:00      4      casual          23
## 3    2011-10-13 07:00:00      4      casual           5
## 4    2011-10-15 14:00:00      4      casual         242
## ..                   ...    ...         ...         ...
## 865 2011-04-07 16:00:00      2  registered         161
## 866 2011-09-03 22:00:00      3  registered          96
## 867 2011-11-12 22:00:00      4  registered          88
## 868 2012-04-11 23:00:00      2  registered          52
## 869 2012-01-06 09:00:00      1  registered         237
##
```

이전에 생성한 긴 형식 데이터프레임을 활용하여 각 계절(season)별로 casual과 registered 사용자의 평균 대여 수(count)를 구하시오.

```
##
## 각 계절별 user_type의 평균 대여 수:
```

```
##    season   user_type  user_count
## 0       1      casual   14.122642
## 1       1  registered   89.047170
## 2       2      casual   48.990196
## 3       2  registered  169.813725
## 4       3      casual   55.127273
## 5       3  registered  210.372727
## 6       4      casual   29.709402
## 7       4  registered  188.871795
```

앱 로그 데이터

데이터셋은 앱 로그에 대한 정보를 포함하고 있습니다.

- 로그 : 로그 정보

```
pd.set_option('display.max_columns', None) # 전체 칼럼 정보 프린트 옵션
df = pd.read_csv(r'~/Desktop/슬기로운통계생활/lsbigdata/본강의자료/data/logdata.csv')
print(df.head(2))
```

```
##                                                  로그
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...
```

로그 칼럼에서 숫자 정보만 추출하시오.

```
## 숫자 정보 추출:
```

```
##                                                    로그   \
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...
##
##                          숫자 정보
## 0  2024 07 18 12 34 56 12345
## 1   2024 07 18 12 35 00 2000
```

로그 칼럼에서 모든 시간 정보를 추출하시오.

```
##                                                    로그      시간 정보
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...  12:34:56
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...  12:35:00
## 2  2024-07-18 12:36:10 User: 이영희 Action: Logout T...  12:36:10
## 3  2024-07-18 12:37:22 User: 박지성 Action: Login ID...  12:37:22
## 4  2024-07-18 12:38:44 User: 최강타 Action: Purchase...  12:38:44
```

로그 칼럼에서 한글 정보만 추출하시오.

```
##
## 한글 정보 필터링:
```

```
##                                                    로그   \
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...
##
##                          숫자 정보      시간 정보 한글 정보
## 0  2024 07 18 12 34 56 12345  12:34:56    홍길동
## 1   2024 07 18 12 35 00 2000  12:35:00    김철수
```

로그 칼럼에서 특수 문자를 제거하시오.

```
##
## 특수 문자 제거:
```

```
##                                                    로그   \
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...
##
##                          숫자 정보      시간 정보 한글 정보  \
## 0  2024 07 18 12 34 56 12345  12:34:56    홍길동
## 1   2024 07 18 12 35 00 2000  12:35:00    김철수
##
##                                                  특수 문자 제거
## 0      20240718 123456 User 홍길동 Action Login ID12345
## 1  20240718 123500 User 김철수 Action Purchase Amoun...
```

로그 칼럼에서 유저, Amount 값을 추출한 후 각 유저별 Amount의 평균값을 계산하시오.

```
##
## 그룹별 평균 Amount 계산:
```

```
##    User        Amount
## 0  김철수  3666.666667
## 1  박지성  5750.000000
## 2  이영희  4250.000000
## 3  장보고  5750.000000
## 4  최강타  3750.000000
## 5  홍길동  4250.000000
```

연습 문제 해답

```
df = pd.read_csv(r'~/Desktop/슬기로운통계생활/lsbigdata/본강의자료/data/bike_data.csv')
df = df.astype({'datetime' : 'datetime64[ns]', 'weather' : 'int64', 'season' : 'object', 'wo
```

계절(season) == 1일 때, 가장 대여량이 많은 시간대(hour)을 구하시오.

```
df_sub = df.loc[df.season == 1, ]

# 시간 정보 추출
df_sub.loc[:, 'hour'] = df_sub['datetime'].dt.hour

# 계절별 및 시간대별 대여량 합계 계산
summary_data = df_sub.groupby(['season', 'hour']).agg({'count': 'sum'}).reset_index()

# count가 가장 큰 hour 찾기
max_count_hour = df_sub.loc[df_sub['count'].idxmax(), 'hour']
max_count = df['count'].max()

print(f"count가 가장 큰 hour는 {max_count_hour}시이며, 대여량은 {max_count}입니다.")
```

```
## count가 가장 큰 hour는 17시이며, 대여량은 970입니다.
```

각 계절(season)별 평균 대여량(count)을 구하시오.

```
# 계절별로 그룹화하여 평균 대여량 계산
season_avg = df.groupby('season')['count'].mean().reset_index()
print(season_avg)
```

```
##    season       count
## 0       1  103.169811
## 1       2  218.803922
## 2       3  265.500000
## 3       4  218.581197
```

특정 달(month) 동안의 총 대여량(count)을 구하시오.

```
# 월 정보 추출
df['month'] = df['datetime'].dt.month

# 특정 월 (예: 1월) 필터링
january_rentals = df[df['month'] == 1]['count'].sum()
print(f"1월 동안의 총 대여량은 {january_rentals}입니다.")
```

```
## 1월 동안의 총 대여량은 2567입니다.
```

가장 대여량이 많은 날짜를 구하시오.

```
# 날짜 정보 추출
df['date'] = df['datetime'].dt.date

# 날짜별로 대여량 합계 계산
date_rentals = df.groupby('date')['count'].sum()

# 가장 대여량이 많은 날짜 찾기
max_rental_date = date_rentals.idxmax()
max_rental_count = date_rentals.max()
print(f"가장 대여량이 많은 날짜는 {max_rental_date}이며, 대여량은 {max_rental_count}입니다.")
```

```
## 가장 대여량이 많은 날짜는 2012-05-11이며, 대여량은 1398입니다.
```

시간대(hour)별 평균 대여량(count)을 구하시오.

```
# 시간 정보 추출
df['hour'] = df['datetime'].dt.hour

# 시간대별로 그룹화하여 평균 대여량 계산
hourly_avg = df.groupby('hour')['count'].mean().reset_index()
print(hourly_avg.head())
```

```
##    hour       count
## 0     0   43.500000
## 1     1   52.714286
## 2     2   32.842105
## 3     3   12.000000
## 4     4    6.687500
```

특정 요일(weekday) 동안의 총 대여량(count)을 구하시오.

```
# 요일 정보 추출 (0=Monday, 6=Sunday)
df['weekday'] = df['datetime'].dt.weekday

# 특정 요일 (예: Monday) 필터링
monday_rentals = df[df['weekday'] == 0]['count'].sum()
print(f"월요일 동안의 총 대여량은 {monday_rentals}입니다.")
```

```
## 월요일 동안의 총 대여량은 10191입니다.
```

주어진 Bike Sharing 데이터를 사용하여 넓은 형식(wide format)에서 긴 형식(long format)으로 변환하시오. casual과 registered 열을 하나의 열로 변환하고, 각 기록의 대여 유형과 대여 수를 포함하는 긴 형식 데이터프레임을 만드시오.

```
# melt를 사용하여 긴 형식으로 변환
melted_df = pd.melt(
    df,
    id_vars=['datetime', 'season'],  # 고정할 열
    value_vars=['casual', 'registered'],  # 녹일 열
    var_name='user_type',  # 새로 생성될 열의 이름
    value_name='user_count'  # 새로 생성될 값 열의 이름
)

print("\nmelt를 사용하여 변환된 데이터프레임:")
```

```
##
## melt를 사용하여 변환된 데이터프레임:
```

```
print(melted_df.head())
```

이전에 생성한 긴 형식 데이터프레임을 활용하여 각 계절(season)별로 casual과 registered 사용자의 평균 대여 수(count)를 구하시오.

```
# season과 user_type별로 그룹화하여 평균 대여 수 계산
avg_rentals = melted_df.groupby(['season', 'user_type'])['user_count'].mean().reset_index()
print("\n각 계절별 user_type의 평균 대여 수:")
```

```
##
## 각 계절별 user_type의 평균 대여 수:
```

```
print(avg_rentals.head())
```

```
##    season   user_type  user_count
## 0       1      casual   14.122642
## 1       1  registered   89.047170
## 2       2      casual   48.990196
## 3       2  registered  169.813725
## 4       3      casual   55.127273
```

앱 로그 데이터

데이터셋은 앱 로그에 대한 정보를 포함하고 있습니다.

- 로그 : 로그 정보

```
pd.set_option('display.max_columns', None) # 전체 칼럼 정보 프린트 옵션
df = pd.read_csv(r'~/Desktop/슬기로운통계생활/lsbigdata/본강의자료/data/logdata.csv')
print(df.head(2))
```

```
##                                                  로그
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...
```

로그 칼럼에서 연도 정보만 추출하시오.

```
df['연도 정보'] = df['로그'].str.extract(r'(\d+)')
print(df.head())
```

```
##                                                  로그 연도 정보
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...  2024
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...  2024
## 2  2024-07-18 12:36:10 User: 이영희 Action: Logout T...  2024
## 3  2024-07-18 12:37:22 User: 박지성 Action: Login ID...  2024
## 4  2024-07-18 12:38:44 User: 최강타 Action: Purchase...  2024
```

로그 칼럼에서 모든 시간 정보를 추출하시오.

```
df['시간 정보'] = df['로그'].str.extract(r'(\d{2}:\d{2}:\d{2})')
print(df[['로그', '시간 정보']].head())
```

```
##                                                  로그      시간 정보
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...  12:34:56
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...  12:35:00
## 2  2024-07-18 12:36:10 User: 이영희 Action: Logout T...  12:36:10
## 3  2024-07-18 12:37:22 User: 박지성 Action: Login ID...  12:37:22
## 4  2024-07-18 12:38:44 User: 최강타 Action: Purchase...  12:38:44
```

로그 칼럼에서 한글 정보만 추출하시오.

```
df['한글 정보'] = df['로그'].str.extract(r'([가-힣]+)')
print(df[['로그', '한글 정보']].head())
```

```
##                                                  로그 한글 정보
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...   홍길동
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...   김철수
## 2  2024-07-18 12:36:10 User: 이영희 Action: Logout T...   이영희
## 3  2024-07-18 12:37:22 User: 박지성 Action: Login ID...   박지성
## 4  2024-07-18 12:38:44 User: 최강타 Action: Purchase...   최강타
```

로그 칼럼에서 특수 문자를 제거하시오.

```
df['특수 문자 제거'] = df['로그'].str.replace(r'[^a-zA-Z0-9가-힣\s]', '', regex=True)
print("\n특수 문자 제거:")
```

```
##
## 특수 문자 제거:
```

```
print(df.head(2))
```

```
##                                                로그 연도 정보      시간 정보 한글 정보  \
## 0  2024-07-18 12:34:56 User: 홍길동 Action: Login ID...  2024  12:34:56   홍길동
## 1  2024-07-18 12:35:00 User: 김철수 Action: Purchase...  2024  12:35:00   김철수
##
##                                                 특수 문자 제거
## 0      20240718 123456 User 홍길동 Action Login ID12345
## 1  20240718 123500 User 김철수 Action Purchase Amoun...
```

로그 칼럼에서 유저, Amount 값을 추출한 후 각 유저별 Amount의 평균값을 계산하시오.

```
df['Amount'] = df['로그'].str.extract(r'Amount:\s*(\d+)').astype(float)
df['User'] = df['로그'].str.extract(r'User:\s*([가-힣]+)')
grouped = df.groupby('User')['Amount'].mean().reset_index()
print("\n그룹별 평균 Amount 계산:")
```

```
##
## 그룹별 평균 Amount 계산:
```

```
print(grouped.head())
```

```
##    User        Amount
## 0  김철수  3666.666667
## 1  박지성  5750.000000
## 2  이영희  4250.000000
## 3  장보고  5750.000000
## 4  최강타  3750.000000
```