

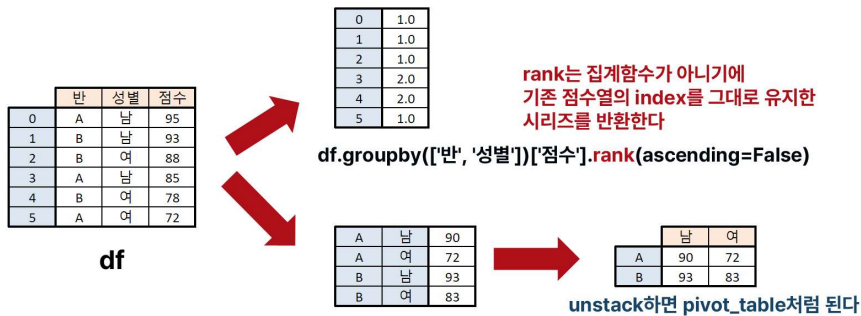
▼ Ch10. 판다스 스킬업

▼ 1. groupby 스킬업

groupby 로 집계함수를 적용하면 다음과 같은 결과를 얻는다

적용하는 함수가 집계함수일 때와 아닐 때의 groupby의 차이

1. 집계함수가 아닐 때



2. 집계함수일 때

`df.groupby(['반', '성별'])['점수'].mean()`


mean은 집계함수라서 index가 없어지고
그룹을 나누는 key가 새로운 index가 된다

그런데 왜 pivot_table 을 두고 groupby 로 집계함수를 적용하는 것일까?

- 실제로는 groupby 로 집계를 하는 경우가 더 많다
- groupby 는 agg 와 transform 등과 결합해 보다 유용한 집계를 할 수 있다

실습 준비 코드

```
import pandas as pd
pd.options.display.float_format = '{:.2f}'.format # 소수점 출력옵션
data = [['김판다', 'A', '남', 95, 77], ['송중기', 'B', '남', 93, 92],
        ['김나현', 'B', '여', 88, 60], ['박효신', 'A', '남', 85, 83],
        ['강승주', 'B', '여', 78, 92], ['권보아', 'A', '여', 72, 75]]
df = pd.DataFrame(data, columns=['이름', '반', '성별', '국어', '수학'])
df
```

이름 반 성별 국어 수학 

```
# groupby로 집계함수 적용하기 복습
df.groupby(['반', '성별'])['국어'].mean()
```

```
반  성별
A  남    90.00
   여    72.00
B  남    93.00
   여    83.00
Name: 국어, dtype: float64
```


groupby 스킬업1

복수의 열에 groupby를 적용하자

```
# data도 복수로 지정이 가능하다 (사실 data는 대괄호 인덱싱)
df.groupby(['반', '성별'])[['국어', '수학']].mean()
```

```
      국어  수학 
반  성별
A  남    90.00  80.00
   여    72.00  75.00
B  남    93.00  92.00
   여    83.00  76.00
```

```
# 인덱싱을 하지 않으면 전체에 적용한다 (단 by에 지정된 열은 제외)
df.groupby(['반', '성별']).mean()
```

```
      국어  수학 
반  성별
A  남    90.00  80.00
   여    72.00  75.00
B  남    93.00  92.00
   여    83.00  76.00
```

더블클릭 또는 Enter 키를 눌러 수정

groupby 스킬업2

groupby로 집계함수를 사용하면 agg와 함께 집계함수를 사용가능하다

agg의 장점

1. agg와 리스트를 이용하면 튜플로 집계결과의 열이름을 바꿀 수 있다

2. 복수의 집계함수 사용가능
3. 열마다 다른 집계함수 사용가능
4. lambda 함수 사용가능

1은 set_axis 등으로 손쉽게 가능하고 2, 4는 pivot_table 로도 가능하므로 실질적 장점은 3이다

```
# 함수를 튜플로 입력해 열 이름을 바꿀 수 있다
df.groupby(['반'])['국어'].agg([('국어평균', 'mean')])
```

	국어평균
반	
A	84.00
B	86.33

```
# 복수의 집계함수 적용
df.groupby(['반'])['국어'].agg(['mean', 'std'])
```

	mean	std
반		
A	84.00	11.53
B	86.33	7.64

```
# 위 결과는 피벗 테이블로도 가능하다
df.pivot_table('국어', index='반', aggfunc=['mean', 'max']).droplevel(1, axis=1)
```

	mean	max
반		
A	84.00	95
B	86.33	93

```
# 복수의 집계함수 일 때도 열 이름을 바꿀 수 있다
df.groupby(['반'])['국어'].agg([('국어평균', 'mean'), ('표준편차', 'max')])
```

	국어평균	표준편차
반		
A	84.00	95
B	86.33	93

얼마다 다른 집계함수 적용

```
df.groupby(['반']).agg({'국어': 'mean', '수학': 'count'})
```

	국어	수학
반		
A	84.00	3
B	86.33	3

lambda 함수도 사용 가능하다. 반별 수학 점수가 90 이상인 인원수

```
df.groupby(['반'])['수학'].agg(lambda x: (x > 80).sum())
```

반	
A	1
B	2

Name: 수학, dtype: int64

groupby 스킬업3

- transform과 집계함수를 이용해 집계 결과를 열로 만들수 있다
- 열로 만들수 있으므로 집계 결과로 불리언 인덱싱도 할 수 있다

transform은 집계함수의 적용 결과를 원본 df와 같은 길이의 시리즈로 반환한다

```
df.groupby('반')['수학'].transform('mean')
```

0	78.33
1	81.33
2	81.33
3	78.33
4	81.33
5	78.33

Name: 수학, dtype: float64

반평균보다 점수가 높은 학생들의 데이터만 필터링

```
cond1 = df['수학'] > df.groupby('반')['수학'].transform('mean')
df[cond1]
```

	이름	반	성별	국어	수학
1	송중기	B	남	93	92
3	박효신	A	남	85	83
4	강승주	B	여	78	92

▼ 2. apply 스킬업

시리즈에 적용할 때와 데이터프레임에 적용할 때의 apply의 차이

시리즈에 **apply** 적용할 때

각 셀마다 함수 적용

0	x1
1	x2
2	x3
3	x4



f(x1)
f(x2)
f(x3)
f(x4)

각 셀을 x1, x2, x3, x4라고 생각하면
f(x1), f(x2), f(x3), f(x4)가 반환

데이터 프레임에 **apply** 적용할 때

각 시리즈마다 함수 적용

	col1	col2
0	y1	z1
1	y2	z2
2	y3	z3
3	y4	z4

x1
x2
x3
x4



f(x1)
f(x2)
f(x3)
f(x4)

axis=1일때 기준으로
각 행을 x1, x2, x3, x4라고 생각하면
f(x1), f(x2), f(x3), f(x4)가 반환

apply가 잘 기억 안나는 분들은 링크 참조 : <https://kimpanda.tistory.com/86>

실습 준비 코드

```
import pandas as pd
data = [['82점', '81점', '77점'], ['91점', '95점', '83점'],
        ['78점', '72점', '88점'], ['82점', '87점', '72점']]
s1 = pd.Series(['80점', '75점', '77점', '60점'])
s2 = pd.Series(['4등', '3등', '1등', '2등'], index=list('ABCD'))
s3 = pd.Series(['없음', '사오기', '만원', '2만원'],
               index=['1등', '2등', '3등', '4등'])
df = pd.DataFrame(data, index=list('ABCD'), columns=['국어', '영어', '수학'])
df
```

	국어	영어	수학
A	82점	81점	77점
B	91점	95점	83점
C	78점	72점	88점
D	82점	87점	72점

apply와 map의 차이

시리즈에 apply 적용하기

```
s1.apply(lambda x: int(x[:-1]))
```

```
0    80
1    75
2    77
3    60
dtype: int64
```

map 함수는 시리즈에 적용할 때 apply와 유사하다

```
s1.map(lambda x: int(x[:-1]))
```

```
0    80
1    75
2    77
3    60
dtype: int64
```

apply와 map의 차이점을 알기 위해 s2와 s3를 이용하자

```
print(s2)
```

```
print(s3)
```

```
A    4등
B    3등
C    1등
D    2등
dtype: object
1등    없음
2등    사오기
3등    만원
4등    2만원
dtype: object
```

map 함수는 함수대신 매퍼를 입력받을 수 있다

```
s2.map(s3)
```

```
A    2만원
B     만원
C     없음
D    사오기
dtype: object
```

apply는 매퍼를 입력받지 못하지만 lambda 함수로 유사한 기능 가능

```
s2.apply(lambda x: s3[x])
```

```
A    2만원
B     만원
C     없음
D    사오기
dtype: object
```

apply와 applymap의 차이

실습에 쓰일 df 출력

```
df
```

국어 영어 수학

applymap은 apply와 달리 데이터 프레임도 셀마다 함수를 적용한다

```
df.applymap(lambda x: int(x[:-1]))
```

국어 영어 수학

A	82	81	77
B	91	95	83
C	78	72	88
D	82	87	72

위 결과를 apply로 얻으려면 map 함수를 한번 더 사용해야 한다

```
df.apply(lambda x: x.map(lambda y: int(y[:-1])))
```

국어 영어 수학

A	82	81	77
B	91	95	83
C	78	72	88
D	82	87	72

map과 apply와 applymap의 차이

	적용 대상	개별 요소	인수
map	시리즈, 인덱스	셀	함수, 매퍼
apply	시리즈, 데이터 프레임	셀(시리즈) 시리즈(데이터 프레임)	함수
applymap	데이터 프레임	셀	함수


apply, map, applymap의 차이 참고 강의 링크 : <https://youtu.be/upY7WRenFjo>

3. 데이터 구조 바꾸기 스킬업

pandas explode

리스트의 데이터가 행으로 확장된다

	점수	이름
0	95	[김판다, 강승주]
1	92	송중기



	점수	이름
0	95	김판다
0	95	강승주
1	92	송중기

df

df.explode('이름')

실습 준비 코드

import pandas as pd

pd.options.display.max_rows = 6 # 판다스 버전업에 따라 6행만 출력의 바뀐 코드

df1 = pd.DataFrame([[95, ['김판다', '강승주']], [92, '송중기']],
columns=['점수', '이름'])df2 = pd.DataFrame([[95, '김판다/강승주'], [92, '송중기']],
columns=['점수', '이름'])

실습에 쓰일 df1. df1은 셀에 리스트를 갖고 있다

df1

	점수	이름
0	95	[김판다, 강승주]
1	92	송중기

explode 함수로 리스트의 데이터를 분리할 수 있다

df1.explode('이름')

	점수	이름
0	95	김판다
0	95	강승주
1	92	송중기

왜 데이터에 리스트가 있는걸까? 보통의 경우 우리가 만드는 것이다

df2

	점수	이름
0	95	김판다/강승주
1	92	송중기

이름열의 /를 분리해서 리스트로 만들고 explode를 사용하면 전처리된다

df2['이름'] = df2['이름'].str.split('/')

df2.explode('이름')

	점수	이름	
0	95	김판다	
0	95	강승주	
1	92	송중기	

explode 함수 실습하기

작품으로 나열된 네이버 웹툰의 데이터를 이용해 작가 중심으로 해당 작가의 작품을 묶어보자

네이버 웹툰 데이터 : https://raw.githubusercontent.com/panda-kim/csv_files/main/naver.csv

```
url = 'https://raw.githubusercontent.com/panda-kim/csv_files/main/naver.csv'
df_ex1 = pd.read_csv(url, usecols=['title', 'author'])
df_ex1
```

	title	author	
0	가난을 등에 업은 소녀	B급달궁 / 오은지	
1	가담항설	랑또	
2	가령의 정체불명 이야기	가령	
...	
1853	[드라마원작] 한번 더 해요	미티 / 구구	
1854	[드라마원작] 내 ID는 ...	기맹기	
1855	[드라마원작] 지금 우리 ...	주동근	

1856 rows × 2 columns

```
df_ex2 = df_ex1.copy()
df_ex2['author'] = df_ex2['author'].str.split(' / ')
df_ex2.explode('author').groupby('author', as_index=False)['title'].agg(' / '.join)
```

	author	title	
0	-2°C	헬 인 파라다이스	
1	0환이	가짜인간	
2	12B	뱀피르	
...	
1737	히디	시월드 판타지	
1738	히어리 꽃미남 저승사자 / 재혼 황후 / 하렘의 남자들		
1739	힐링달	인싸라이프	

1740 rows × 2 columns

join 함수는 시리즈의 문자열을 합치는 함수이다

```
s = pd.Series(['a', 'bc'])
'/'.join(s)

'a/bc'
```

▼ 4. 시각화 스킬업

파이썬 시각화를 학습할 때의 문제점

- 대부분 시각화 라이브러리를 제대로 쓰려면 matplotlib부터 배워야 한다
- 그래서 정작 그래프는 그리지 않는 matplotlib만 배우고 끝나는 경우가 많다

판다스의 시각화

- 데이터 프레임에 바로 함수를 적용해 간단한 코드로 시각화가 가능한 라이브러리들이 많다
ex) seaborn, pandas-bokeh
- 다만 꼭 matplotlib가 아니더라도 제대로 쓰려면 여전히 기반 라이브러리를 배워야 한다
ex) seaborn은 matplotlib 기반, pandas-bokeh는 bokeh 기반

→ 기반 라이브러리를 많이 써야 할 수록 판다스 시각화의 간결함의 장점이 사라진다

해결책

- 데이터 프레임에 바로 함수를 적용해 간결한 코드로 판다스의 시각화가 가능하며
- 가급적 기반 라이브러리를 사용하고 않고 해당 판다스 시각화 라이브러리 자체로 해결이 가능
- 반응형 그래프를 그릴 수 있는 라이브러리

위와 같은 라이브러리라면 간결하고 예쁜 시각화가 손쉽게 가능하다

-> pyechart 기반의 pandasecharts를 배워보자

pyecharts

echarts를 python에서 사용할 수 있게 해주는 라이브러리

- 기본값이 예쁘다
- 반응형이다

pyecharts 사이트 링크 : <https://pyecharts.org/#/en-us/>

갤러리 링크 : https://gallery.pyecharts.org/#/README_EN

pyecharts 설치 (1.9.1버전 설치)

```
!pip install --upgrade pyecharts==1.9.1
```

pandasecharts

데이터 프레임에 pyecharts를 쉽게 사용하게 해주는 라이브러리

pandasecharts 설치

```
!pip install pandasecharts
```

pyecharts 1.9.1 설치

```
!pip install --upgrade pyecharts==1.9.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyecharts==1.9.1 in /usr/local/lib/python3.9/dist-packages (1.9.1)
Requirement already satisfied: simplejson in /usr/local/lib/python3.9/dist-packages (from pyecharts==1.9.1) (3.17.2)
Requirement already satisfied: prettytable in /usr/local/lib/python3.9/dist-packages (from pyecharts==1.9.1) (3.10.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from pyecharts==1.9.1) (3.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->pyecharts==1.9.1) (2.1.1)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.9/dist-packages (from prettytable->pyecharts==1.9.1) (0.2.5)
```

pandasecharts 설치

```
!pip install pandasecharts
```

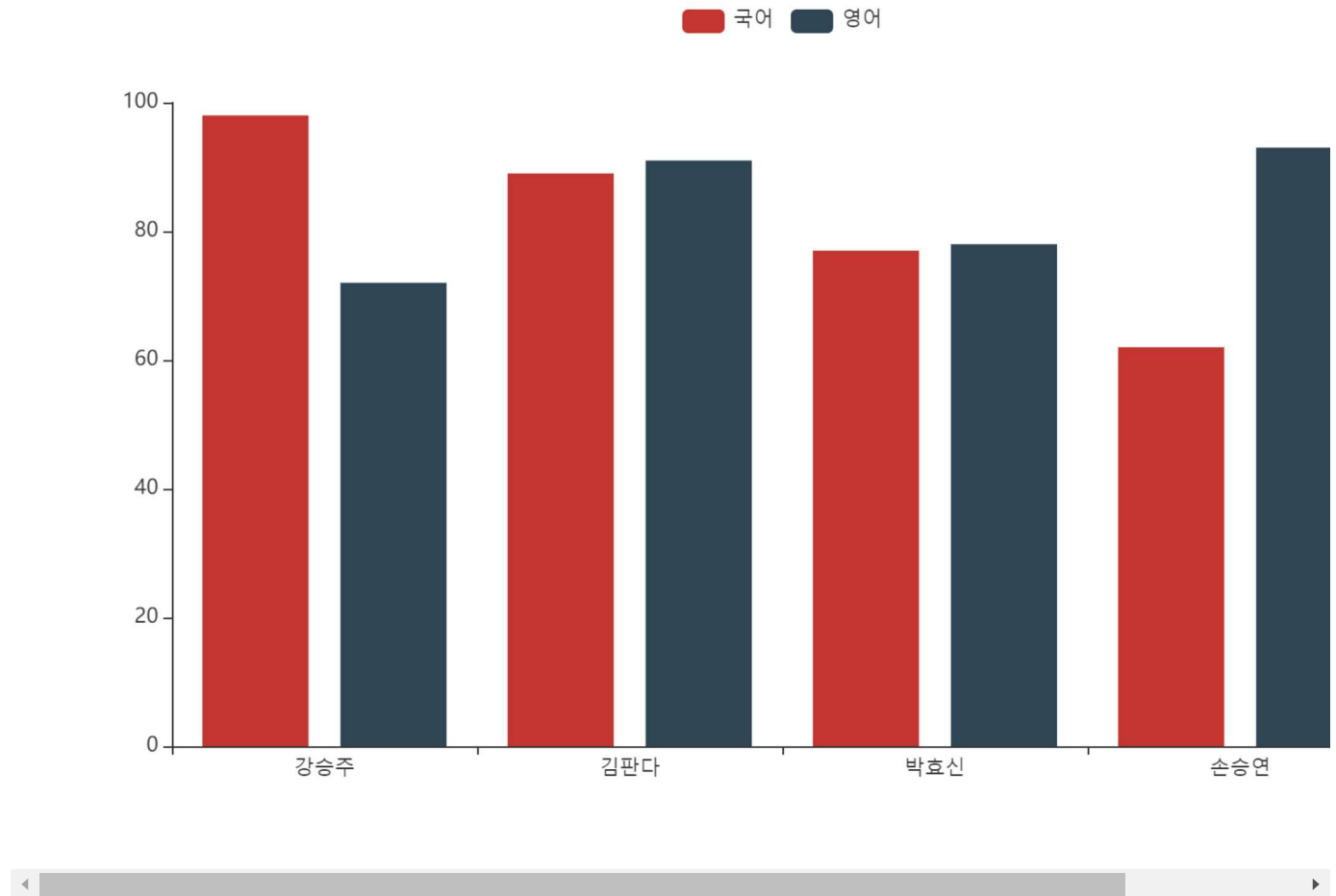
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandasecharts in /usr/local/lib/python3.9/dist-packages (0.5)
Requirement already satisfied: pyecharts>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from pandasecharts) (1.9.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from pandasecharts) (1.4.4)
Requirement already satisfied: prettytable in /usr/local/lib/python3.9/dist-packages (from pyecharts>=1.0.0->pandasecharts) (3.10.0)
Requirement already satisfied: simplejson in /usr/local/lib/python3.9/dist-packages (from pyecharts>=1.0.0->pandasecharts) (3.17.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from pyecharts>=1.0.0->pandasecharts) (3.1.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas->pandasecharts) (2.8.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->pandasecharts) (2020.5)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dist-packages (from pandas->pandasecharts) (1.21.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandasecharts) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->pyecharts>=1.0.0->pandasecharts) (2.1.1)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.9/dist-packages (from prettytable->pyecharts>=1.0.0->pandasecharts) (0.2.5)
```

실습 준비코드

```
import pandas as pd
from pyecharts.globals import ThemeType
from pyecharts.charts import Timeline
from pandasecharts import echart
import IPython

data1 = [['강승주', 98, 72], ['김판다', 89, 91],
         ['박효신', 77, 78], ['손승연', 62, 93]]
data2 = [['강승주', 72, 79], ['김판다', 83, 81],
         ['박효신', 74, 87], ['손승연', 92, 89]]
df1 = pd.DataFrame(data1, columns=['이름', '국어', '영어'])
df2 = pd.DataFrame(data2, columns=['이름', '국어', '영어'])
```

```
# bar 그래프 그리기(render.html 파일로 저장)  
df1.echart.bar(x='이름', ys=['국어', '영어']).render()  
IPython.display.HTML(filename='/content/render.html') # 코랩에서 출력
```



```
# 저장 파일명 바꾸기  
df1.echart.bar(x='이름', ys=['국어', '영어']).render('파일명.html')  
IPython.display.HTML(filename='/content/파일명.html') # 코랩에서 출력
```

국어 영어

저장 않고 노트북으로 출력(코랩에서는 되지 않는다)

```
df1.echart.bar(x='이름', ys=['국어', '영어']).render_notebook()
```

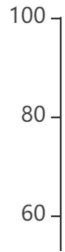


크기 변경

```
df1.echart.bar(x='이름', ys=['국어', '영어'], figsize=(600,400)).render()
```

```
!Python.display.HTML(filename='/content/render.html') # 코랩에서 출력
```

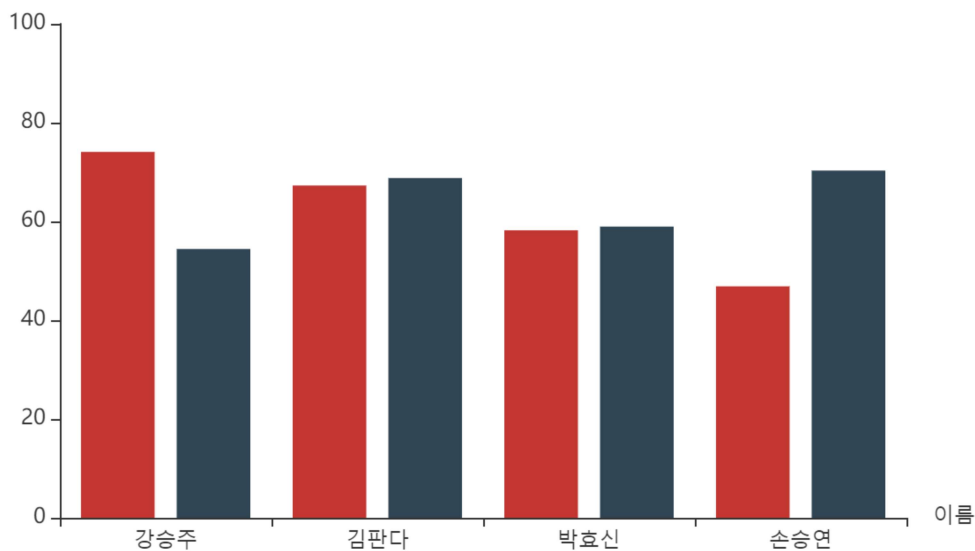
국어 영어



국어열의 값에 따라 정렬

```
df1.echart.bar(x='이름', ys=['국어', '영어'], sort='국어', figsize=(600,400)).render()
IPython.display.HTML(filename='/content/render.html') # 코랩에서 출력
```

국어 영어

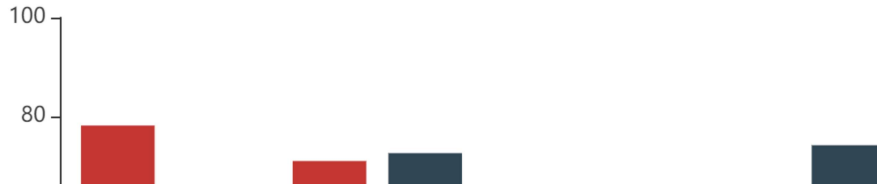


제목 생성

```
df1.echart.bar(x='이름', ys=['국어', '영어'], title='중간고사',
               figsize=(600,400)).render()
IPython.display.HTML(filename='/content/render.html') # 코랩에서 출력
```

중간고사

■ 국어 ■ 영어



소제목(subtitle) 생성

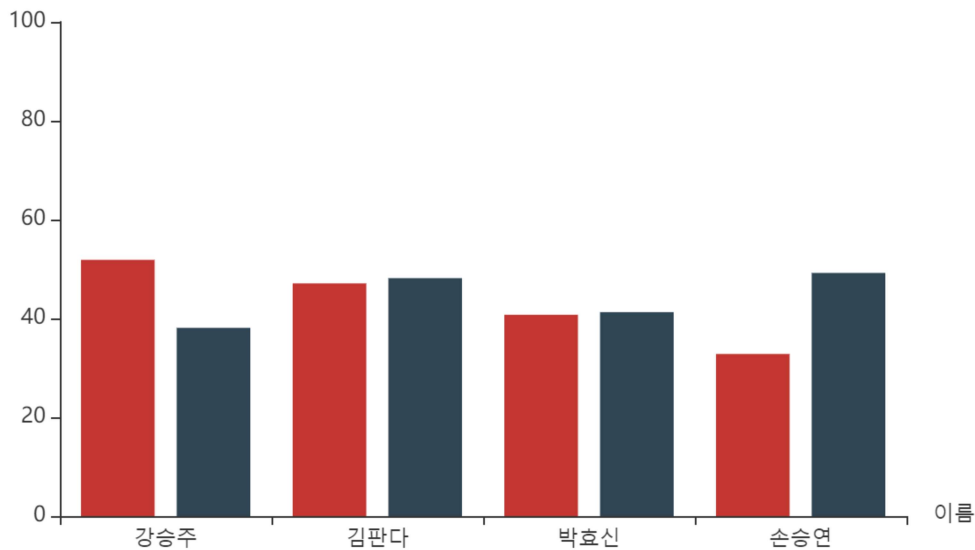
```
df1.echart.bar(x='이름', ys=['국어', '영어'], title='중간고사',
               subtitle='누가누가잘했나', figsize=(600,400)).render()
```

!Python.display.HTML(filename='/content/render.html') # 코랩에서 출력

중간고사

■ 국어 ■ 영어

누가누가잘했나



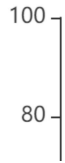
수치 표현하기

```
df1.echart.bar(x='이름', ys=['국어', '영어'], title='중간고사',
               subtitle='누가누가잘했나', label_show=True, figsize=(600,400)).render()
```

!Python.display.HTML(filename='/content/render.html') # 코랩에서 출력

중간고사

누가누가 잘했나

■ 국어
 ■ 영어


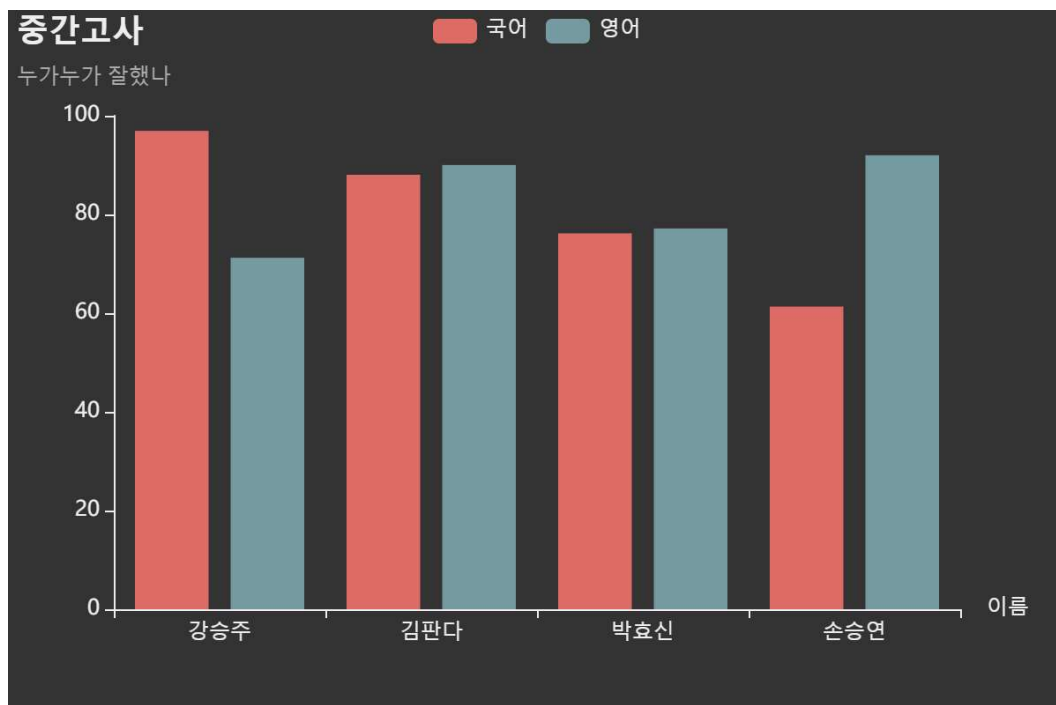
pyecharts 테마

- LIGHT
- DARK
- CHALK
- ESSOS
- INFOGRAPHIC
- MACARONS
- PURPLE_PASSION
- ROMA
- ROMANTIC
- SHINE
- VINTAGE

다음 링크로 테마를 확인해볼 수 있다: <https://pyecharts.org/#/en-us/themes?id=theme-style>

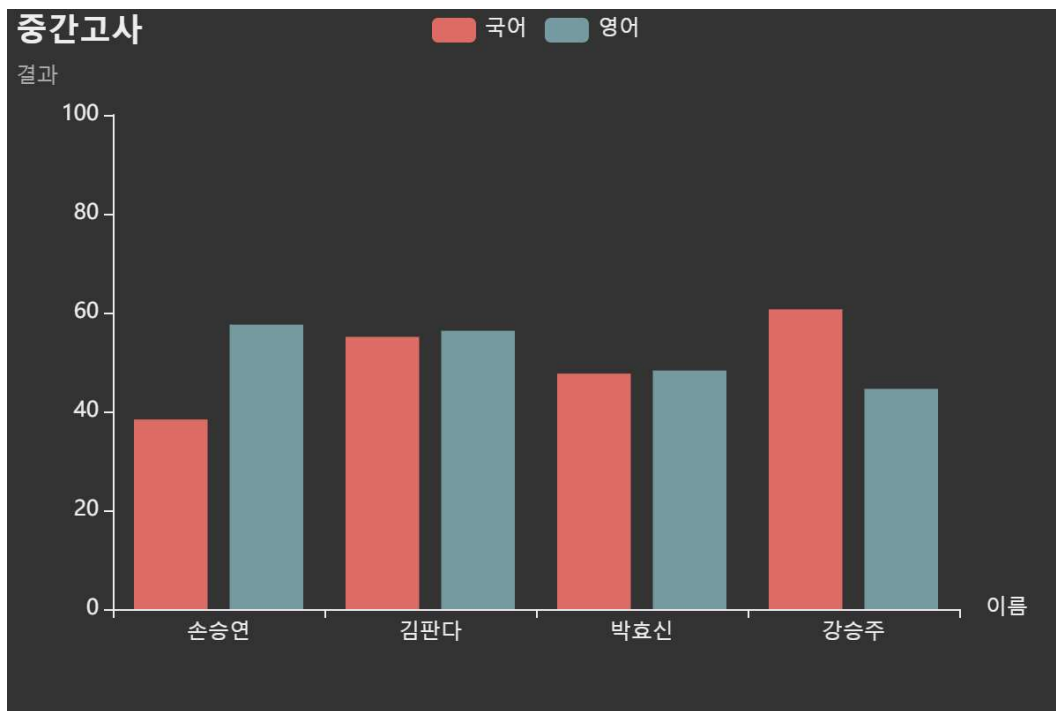
테마 변경

```
df1.echart.bar(x='이름', ys=['국어', '영어'], title='중간고사', figsize=(600,400),
               subtitle='누가누가 잘했나', theme=ThemeType.DARK).render()
!Python.display.HTML(filename='/_content/render.html')
```




```
# 함수 및 인자 정리
```

```
(df1
 .echart.bar(x='이름',
             ys=['국어', '영어'],
             sort='영어',
             title='중간고사',
             subtitle='결과',
             figsize=(600,400),
             theme=ThemeType.DARK
             )
 .render()
 )
 IPython.display.HTML(filename='/content/render.html')
```

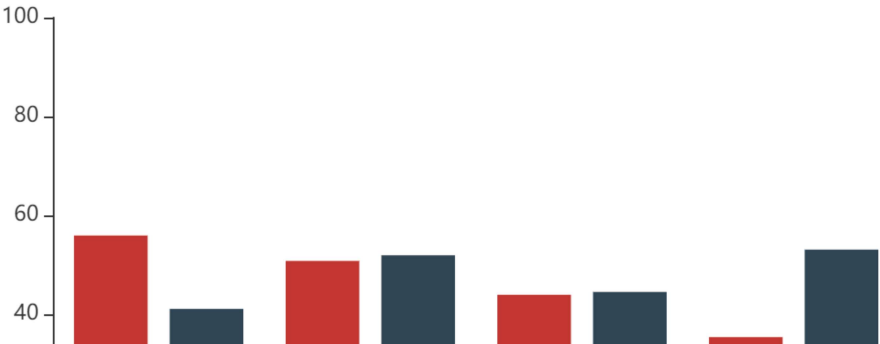


```
# 타임라인
```

```
tl = Timeline({'width': '600px', 'height': '400px'})
bar1 = df1.echart.bar(x='이름', ys=['국어', '영어'], title='1학기', subtitle='중간고사')
bar2 = df2.echart.bar(x='이름', ys=['국어', '영어'], title='1학기', subtitle='기말고사')
tl.add(bar1, '중간고사').add(bar2, '기말고사').render()
IPython.display.HTML(filename='/content/render.html')
```

1학기

중간고사



막대 그래프 외의 그래프를 pandasecharts로 그리고 싶다면 링크를 참조하라

링크 : <https://caoqinping.com/2021/12/17/pandasecharts%E4%BD%BF%E7%94%A8%E7%A4%BA%E4%BE%8B/>

