
▣ CH05 데이터 분석 기초

: 데이터 파악하기, 그래프를 그리기에 적합한 데이터 만들기

```
#.head() #.tail() #.shape() #.info() #.describe() #.copy() #.rename() #파생변수  
#.value_counts() #.to_frame()  
#.where() #빈도수막대그래프 #pandas.Serise.plot.bar() #.barh() #seabon.countplot()  
.sort_index() #.text() #.set_title() #.set_xlabel() #.set_ylabel() #matplotlib.pyplot.show()
```

[05-1] 데이터 파악하기

.head() : 앞부분 출력

.tail() : 뒷부분 출력

.shape() : 행, 열 갯수 출력

.info() : 변수 속성 출력

.describe() : 요약 통계량 출력

※사전에 워킹 디렉토리에 **exam_csv.csv** 파일을 위치시킴.

```
In [5]: ## 데이터 파악하기  
import pandas as pd  
exam = pd.read_csv('exam_csv.csv')  
  
exam.head() # 앞에서부터 5행까지 출력
```

```
Out[5]:    id  nclass  math  english  science
```

0	1	1	50	98	50
1	2	1	60	97	60
2	3	1	45	86	78
3	4	1	30	98	58
4	5	2	25	80	65

```
In [2]: exam.head(3) # 앞에서부터 3행까지 출력
```

```
Out[2]:    id  nclass  math  english  science
```

0	1	1	50	98	50
1	2	1	60	97	60
2	3	1	45	86	78

```
In [3]: exam.tail() # 뒤에서부터 5행까지 출력
```

```
Out[3]:    id  nclass  math  english  science
```

15	16	4	58	98	65
16	17	5	65	68	98
17	18	5	80	78	90
18	19	5	89	68	87
19	20	5	78	83	58

```
In [4]: exam.tail(3) # 뒤에서부터 3행까지 출력
```

```
Out[4]:   id  nclass  math  english  science
          17    18      5     80       78      90
          18    19      5     89       68      87
          19    20      5     78       83      58
```

```
In [9]: exam.info() #속성 출력
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   id        20 non-null    int64  
 1   nclass    20 non-null    int64  
 2   math      20 non-null    int64  
 3   english   20 non-null    int64  
 4   science   20 non-null    int64  
dtypes: int64(5)
memory usage: 928.0 bytes
```

```
In [8]: exam.shape #데이터가 몇 행, 몇 열로 구성되는지 출력
```

```
Out[8]: (20, 5)
```

```
In [4]: exam.describe(include='all') #요약 통계량 (전체: 문자열도 포함) 출력
exam.describe() #요약 통계량 (일부: 숫자 열만) 출력
```

Out[4]:

	id	nclass	math	english	science
count	20.00000	20.000000	20.000000	20.000000	20.000000
mean	10.50000	3.000000	57.450000	84.900000	59.450000
std	5.91608	1.450953	20.299015	12.875517	25.292968
min	1.00000	1.000000	20.000000	56.000000	12.000000
25%	5.75000	2.000000	45.750000	78.000000	45.000000
50%	10.50000	3.000000	54.000000	86.500000	62.500000
75%	15.25000	4.000000	75.750000	98.000000	78.000000
max	20.00000	5.000000	90.000000	98.000000	98.000000

In [5]: exam['math'].describe() #Service에 대한 요약 통계 출력

Out[5]:

```
count    20.000000
mean     57.450000
std      20.299015
min      20.000000
25%     45.750000
50%     54.000000
75%     75.750000
max     90.000000
Name: math, dtype: float64
```

In []:

mpg 데이터 파악하기

mpg(mile per gallon)는 미국 환경보호국에서 공개한 데이터

1999~2008년 미국에 출시된 자동차 24종의 제원 정보

주요 mpg 데이터 컬럼

displ : 배기량, cyl : 실린더 수, trans : 변속기 종류, drv : 구동 방식, fl : 연료 종류

cty : 시내 연비, hwy : 고속도로 연비

```
In [6]: ## mpg 데이터 불러오기  
import pandas as pd  
mpg = pd.read_csv('mpg.csv')
```

```
In [10]: mpg.describe() #요약 통계량 (일부: 숫자 열만) 출력
```

Out[10]:

	displ	year	cyl	cty	hwy
count	234.000000	234.000000	234.000000	234.000000	234.000000
mean	3.471795	2003.500000	5.888889	16.858974	23.440171
std	1.291959	4.509646	1.611534	4.255946	5.954643
min	1.600000	1999.000000	4.000000	9.000000	12.000000
25%	2.400000	1999.000000	4.000000	14.000000	18.000000
50%	3.300000	2003.500000	6.000000	17.000000	24.000000
75%	4.600000	2008.000000	8.000000	19.000000	27.000000
max	7.000000	2008.000000	8.000000	35.000000	44.000000

```
In [11]: mpg.describe(include='all') #요약 통계량 (전체: 문자열도 포함) 출력
```

Out[11]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
count	234	234	234.000000	234.000000	234.000000	234	234	234.000000	234.000000	234	234
unique	15	38	NaN	NaN	NaN	10	3	NaN	NaN	5	7
top	dodge	caravan	2wd	NaN	NaN	NaN	auto(l4)	f	NaN	NaN	r
freq	37	11	NaN	NaN	NaN	83	106	NaN	NaN	168	62
mean	NaN	NaN	3.471795	2003.500000	5.888889	NaN	NaN	16.858974	23.440171	NaN	NaN
std	NaN	NaN	1.291959	4.509646	1.611534	NaN	NaN	4.255946	5.954643	NaN	NaN
min	NaN	NaN	1.600000	1999.000000	4.000000	NaN	NaN	9.000000	12.000000	NaN	NaN
25%	NaN	NaN	2.400000	1999.000000	4.000000	NaN	NaN	14.000000	18.000000	NaN	NaN
50%	NaN	NaN	3.300000	2003.500000	6.000000	NaN	NaN	17.000000	24.000000	NaN	NaN
75%	NaN	NaN	4.600000	2008.000000	8.000000	NaN	NaN	19.000000	27.000000	NaN	NaN
max	NaN	NaN	7.000000	2008.000000	8.000000	NaN	NaN	35.000000	44.000000	NaN	NaN

In [12]: ## mpg 데이터 확인
mpg.tail()

Out[12]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

In [13]: ## mpg 데이터 확인
mpg.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   manufacturer 234 non-null    object 
 1   model        234 non-null    object 
 2   displ         234 non-null    float64
 3   year          234 non-null    int64  
 4   cyl           234 non-null    int64  
 5   trans         234 non-null    object 
 6   drv           234 non-null    object 
 7   cty           234 non-null    int64  
 8   hwy           234 non-null    int64  
 9   fl            234 non-null    object 
 10  category      234 non-null    object 
dtypes: float64(1), int64(4), object(6)
memory usage: 20.2+ KB
```

In [14]: `mpg.head()`

Out[14]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact

[05-2] 변수명 바꾸기

데이터 프레임 복사

데이터 프레임 열 변수명 바꾸기

In [21]:

```
## 데이터 프레임 복사
df = mpg.copy() #deep copy
df.head(1)
```

```
Out[21]:   manufacturer model  displ  year cyl  trans  drv  cty  hwy fl  category
          0      audi     a4    1.8 1999    4 auto(l5)    f  18    29 p  compact
```

```
In [22]: ## 데이터 프레임 열 변수명 바꾸기
df = df.rename(columns = {'cty' : 'city', 'hwy' : 'highway'})
df.head(1)
```

```
Out[22]:   manufacturer model  displ  year cyl  trans  city  highway fl  category
          0      audi     a4    1.8 1999    4 auto(l5)    f    18      29 p  compact
```

df = mpg.copy()와 df = mpg의 차이

- > df = mpg.copy()는 mpg 데이터를 df에 복사하는 방식으로 처리(deep copy)
- > df = mpg는 mpg 데이터를 df가 참조하는 방식으로 처리(shallow copy)

```
In [23]: # copy()로 복사된 df의 행 값 변경
df.at[0, 'city'] = df.at[0, 'city'] + 100 #.at[index, column_name]
df.head(1)
```

```
Out[23]:   manufacturer model  displ  year cyl  trans  city  highway fl  category
          0      audi     a4    1.8 1999    4 auto(l5)    f   118      29 p  compact
```

```
In [24]: mpg.head(1) #변화 없는 원본 데이터 확인
```

```
Out[24]:   manufacturer model  displ  year cyl  trans  drv  cty  hwy fl  category
          0      audi     a4    1.8 1999    4 auto(l5)    f  18    29 p  compact
```

```
In [9]: #mpg 데이터를 df가 참조하는 방식으로 처리
df = mpg #mpg 데이터를 df가 참조하는 방식으로 처리
df.head(1) #copy()된 데이터 확인
```

```
Out[9]:    manufacturer model  displ  year cyl  trans  drv  cty  hwy fl  category
          0      audi     a4   1.8 1999     4 auto(l5)   f  18   29 p  compact
```

```
In [10]: # mpg를 참조하는 df의 행 값 변경
df.at[0, 'cty'] = df.at[0, 'cty'] + 100
```

```
In [11]: df.head(1)
```

```
Out[11]:    manufacturer model  displ  year cyl  trans  drv  cty  hwy fl  category
          0      audi     a4   1.8 1999     4 auto(l5)   f 118   29 p  compact
```

```
In [12]: mpg.head(1) # 함께 바뀐 원본 데이터 확인
```

```
Out[12]:    manufacturer model  displ  year cyl  trans  drv  cty  hwy fl  category
          0      audi     a4   1.8 1999     4 auto(l5)   f 118   29 p  compact
```

```
In [ ]:
```

[05-3] 파생변수 만들기

- > 파생 변수(derived variable)는 기존 속성들로부터 새롭게 만들어낸 변수
- > 변수의 추가는 데이터 프레임에 새로운 열(column) 변수명을 사용하여 연산식으로 대입하면 됨.

```
In [25]: ## 파생변수 만들기: 데이터 프레임 생성
df = pd.DataFrame({'var1' : [4, 3, 8],
                   'var2' : [2, 6, 1]})
df
```

Out[25]:

	var1	var2
0	4	2
1	3	6
2	8	1

In [26]:

```
## 파생변수 만들기: 데이터 프레임에 파생변수 'var_sum' 추가
df['var_sum'] = df['var1'] + df['var2'] # var_sum 파생변수 만들기
df
```

Out[26]:

	var1	var2	var_sum
0	4	2	6
1	3	6	9
2	8	1	9

In [27]:

```
## 파생변수 만들기: 데이터 프레임에 파생변수 'var_mean' 추가
df['var_mean'] = (df['var1'] + df['var2']) / 2 # var_mean 파생변수 만들기
df
```

Out[27]:

	var1	var2	var_sum	var_mean
0	4	2	6	3.0
1	3	6	9	4.5
2	8	1	9	4.5

In []:

[실습-1] mpg 통합연비 파생변수 만들기

1. 'total' 열 변수를 추가하시오.
>> 'total'은 'cty'와 'hwy' 연비의 평균
2. 'total'로부터 전체 통합 평균 연비를 구하시오.

```
In [7]: ## mpg 데이터 불러오기
import pandas as pd
mpg = pd.read_csv('mpg.csv')
mpg.head()
```

```
Out[7]:
```

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact

```
In [8]: ## [실습] mpg 통합연비 파생변수 만들기
# 1. mpg 통합연비 파생변수 만들기
mpg['total'] = (mpg['cty'] + mpg['hwy']) / 2 # 통합 연비 변수 만들기
mpg.head()
```

```
Out[8]:
```

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category	total
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact	23.5
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact	25.0
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact	25.5
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact	25.5
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact	21.0

```
In [30]: ## [실습] mpg 통합연비 파생변수 만들기
# 2. 'total'로부터 전체 통합 평균 연비를 구하
sum(mpg['total']) / len(mpg)
```

```
Out[30]: 20.14957264957265
```

```
In [31]: mpg['total'].mean() # 통합 연비 변수 평균
```

```
Out[31]: 20.14957264957265
```

```
In [ ]:
```

numpy.nowhere() 조건문을 활용해 파생변수 만들기

1. 'total' <= 20인 행 확인

2. 'total' 값이 20 이상이면 'pass', 그렇지 않으면 'fail'로 치환

```
In [30]: ## numpy.nowhere() 조건문을 활용해 파생변수 만들기
# 'total' <= 20인 행 확인
mpg[(mpg['total'] <= 20)] #조건부 행 추출
```

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category	total	test
11	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact	20.0	pass
14	audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact	20.0	pass
15	audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize	19.5	fail
17	audi	a6 quattro	4.2	2008	8	auto(s6)	4	16	23	p	midsize	19.5	fail
18	chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv	17.0	fail
...
203	toyota	toyota tacoma 4wd	3.4	1999	6	manual(m5)	4	15	17	r	pickup	16.0	fail
204	toyota	toyota tacoma 4wd	3.4	1999	6	auto(l4)	4	15	19	r	pickup	17.0	fail
205	toyota	toyota tacoma 4wd	4.0	2008	6	manual(m6)	4	15	18	r	pickup	16.5	fail
206	toyota	toyota tacoma 4wd	4.0	2008	6	auto(l5)	4	16	20	r	pickup	18.0	fail
219	volkswagen	jetta	2.8	1999	6	auto(l4)	f	16	23	r	compact	19.5	fail

111 rows × 13 columns

```
In [31]: ## numpy.nowhere() 조건문을 활용해 파생변수 만들기
# 'total' 값이 20 이상이면 'pass', 그렇지 않으면 'fail'로 치환
import numpy as np #수치연산 지원 패키지 numpy
```

```
x = np.where(mpg['total'] >= 20, 'pass', 'fail') #np.where(조건식, True 시 실행문, False 시 실행문)
```

[실습-2] numpy.nowhere() 조건문을 활용해 파생변수 'test' 만들기

1. 통합연비('total')에 대한 'pass' / 'fail' 파생변수 'test' 추가
> ('total' > 20) 이면 'pass', 아니면 'fail'
 2. 'pass' 또는 'fail' 판정 빈도수 데이터 만들기
 3. 막대 그래프로 빈도 표현하기

In [32]:

```
## [실습] 조건문을 활용해 파생변수 만들기
#### 1. 합격 판정 변수 'test' 만들기: 20 이상이면 pass, 그렇지 않으면 fail 부여
import numpy as np #수치연산 지원 패키지 numpy
mpg['test'] = np.where(mpg['total'] >= 20, 'pass', 'fail')
mpg.head(20)
```

Out[32]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category	total	test
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact	23.5	pass
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact	25.0	pass
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact	25.5	pass
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact	25.5	pass
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact	21.0	pass
5	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact	22.0	pass
6	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact	22.5	pass
7	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact	22.0	pass
8	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact	20.5	pass
9	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact	24.0	pass
10	audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact	23.0	pass
11	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact	20.0	pass
12	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact	21.0	pass
13	audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact	21.0	pass
14	audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact	20.0	pass
15	audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize	19.5	fail
16	audi	a6 quattro	3.1	2008	6	auto(s6)	4	17	25	p	midsize	21.0	pass
17	audi	a6 quattro	4.2	2008	8	auto(s6)	4	16	23	p	midsize	19.5	fail
18	chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv	17.0	fail
19	chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	11	15	e	suv	13.0	fail

```
In [33]: ## [실습] 조건문을 활용해 파생변수 만들기
##### 2. 'pass' 또는 'fail' 판정 자동차 빈도 수 데이터 만들기
count_test = mpg['test'].value_counts()
count_test
```

```
Out[33]: test
pass    128
fail    106
Name: count, dtype: int64
```

빈도수 막대 그래프 그리기

A: pandas.Series.plot.bar()

> 집계된 빈도수 데이터(Serise, DataFrame)를 기반으로 작성

x, y : x, y 축에 사용될 데이터

xlabel, ylabel : x, y축 레이블

title : 제목

grid, legend : 그리드, 범례 사용 여부 True / False

color : 막대의 색상 [b(blue), g(green), r(red), w(white), y(yellow), C(청록), k(검정)]

rot : x축 레이블의 회전각도(default 90)

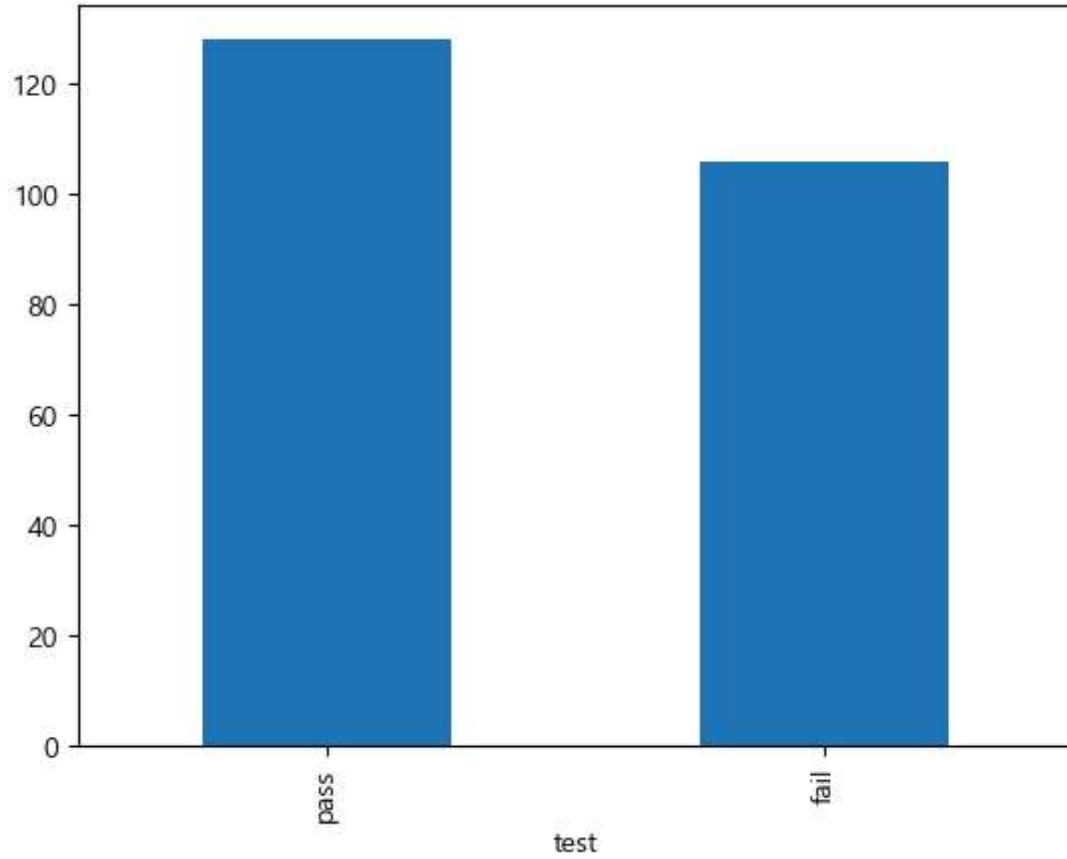
B: seaborn.countplot() 활용

> 메서드가 빈도수를 구해서 작성

[실습 A] pandas.Series.plot.bar() 활용

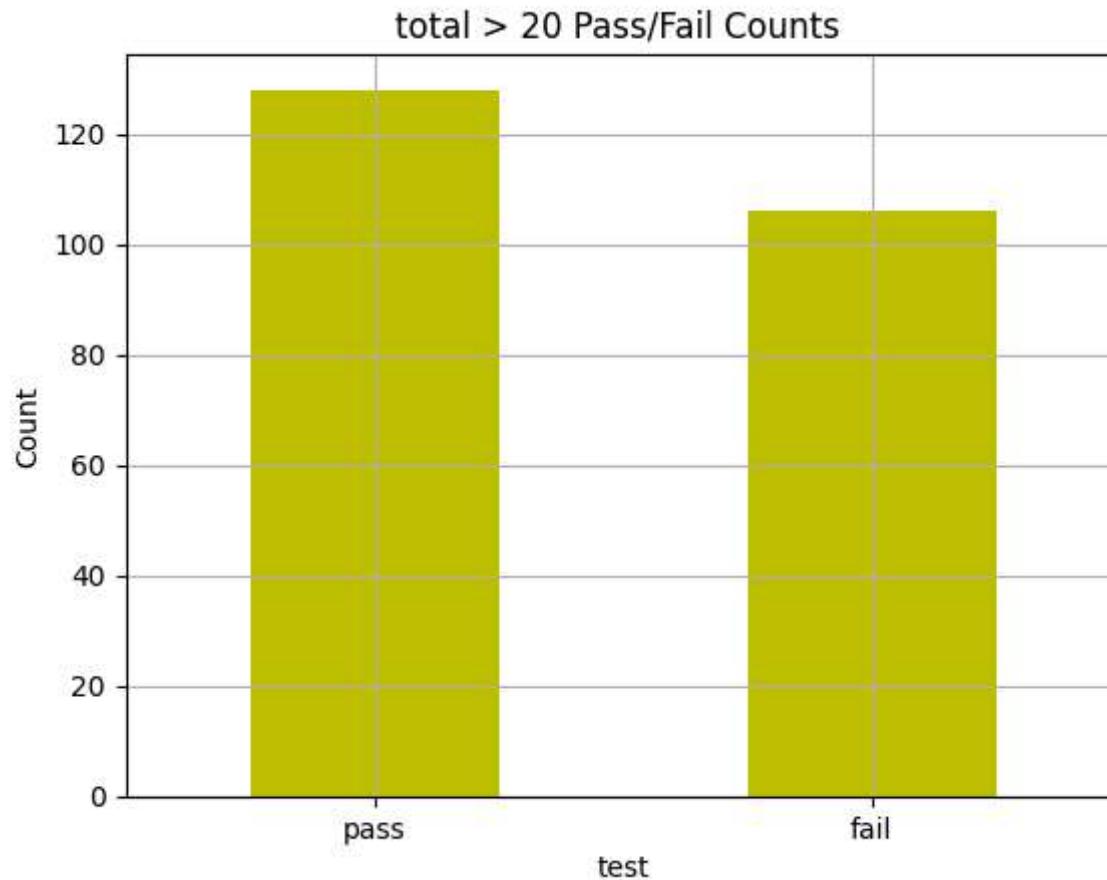
```
In [35]: ## [실습] 조건문을 활용해 파생변수 만들기
# 3. 막대 그래프로 빈도 표현하기: pandas 라이브러리 활용
count_test.plot.bar() # plot.bar()는 Serise 자료형에서 제공되는 메서드
```

```
Out[35]: <Axes: xlabel='test'>
```



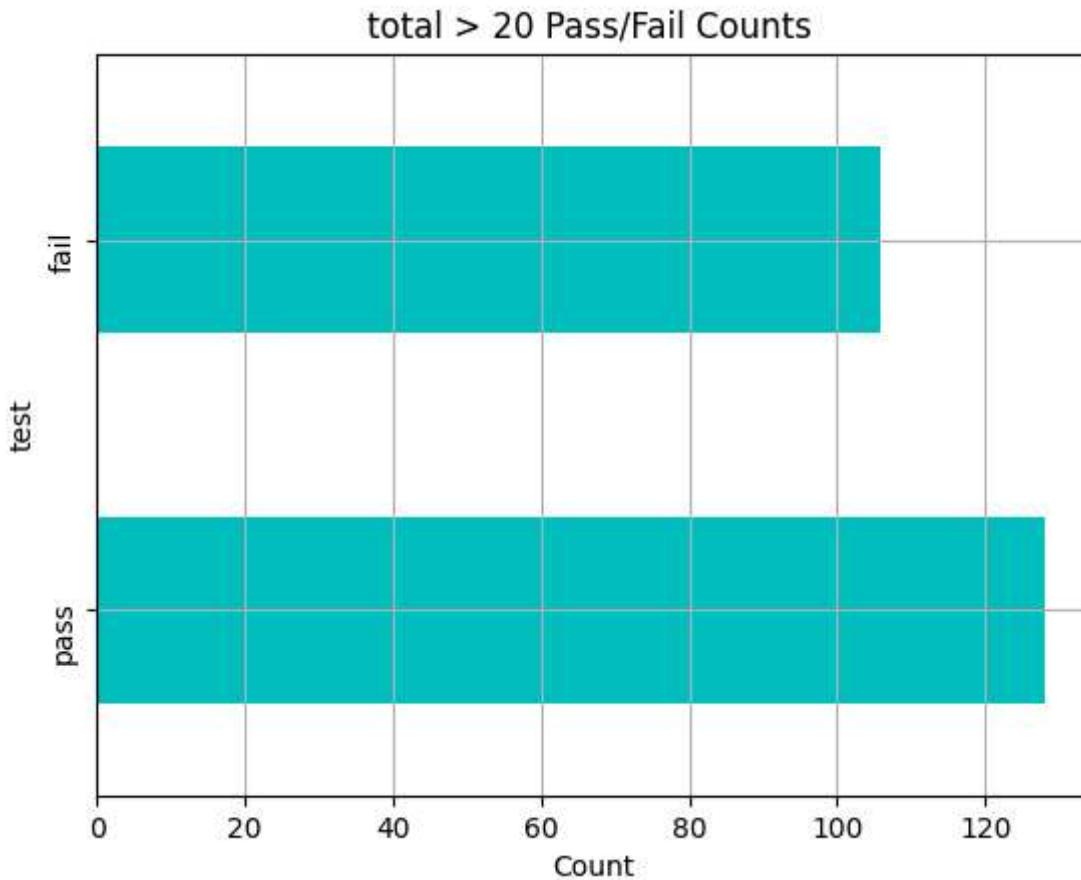
```
In [15]: ## Series나 DataFrame에 대한 막대 그래프 그리기: .plot.bar()의 매개변수  
count_test.plot.bar(y='count', title="total > 20 Pass/Fail Counts ", ylabel='Count', grid=True, color='y', rot=0)
```

```
Out[15]: <Axes: title={'center': 'total > 20 Pass/Fail Counts '}, xlabel='test', ylabel='Count'>
```



```
In [20]: ## 수평 막대로 그리기: .plot.barh()
count_test.plot.barh(x='count', title="total > 20 Pass/Fail Counts ", xlabel='Count', grid=True, color='c', rot=90)
```

```
Out[20]: <Axes: title={'center': 'total > 20 Pass/Fail Counts '}, xlabel='Count', ylabel='test'>
```



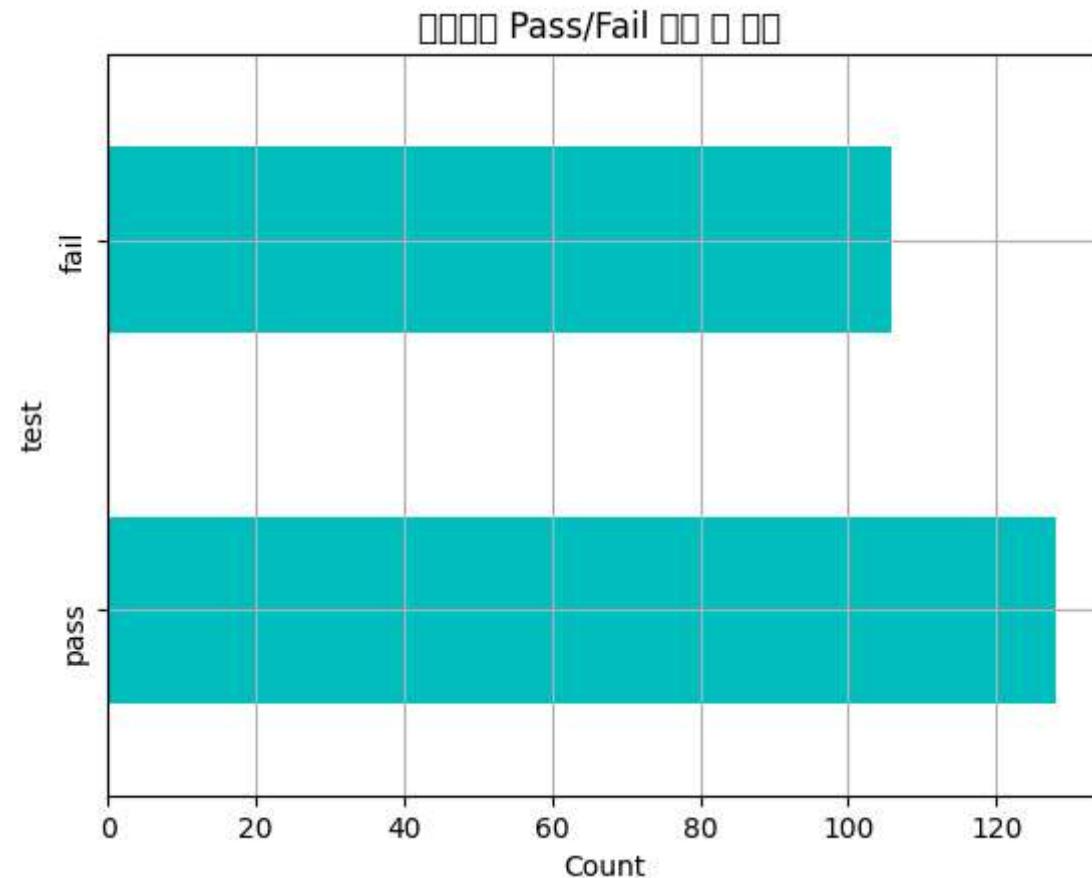
그래프의 제목(title)에 한글 사용하기

```
In [21]: ## 수평 막대로 그리기: 한글 title  
count_test.plot.barh(x='count', title="통합연비 Pass/Fail 모델 수 비교", xlabel='Count', grid=True, color='c', rot=90)
```

```
Out[21]: <Axes: title={'center': '통합연비 Pass/Fail 모델 수 비교'}, xlabel='Count', ylabel='test'>
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 53685 (\uac00{HANGUL SYLLABLE TONG}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 54633 (\uac00{HANGUL SYLLABLE HAB}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 50672 (\uac00{HANGUL SYLLABLE YEON}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 48708 (\uac00{HANGUL SYLLABLE BI}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 47784 (\uac00{HANGUL SYLLABLE MO}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 45944 (\uac00{HANGUL SYLLABLE DEL}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 49688 (\uac00{HANGUL SYLLABLE SU}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 44368 (\uac00{HANGUL SYLLABLE GYO}) missing from current font.  
.. func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 53685 (\uac00{HANGUL SYLLABLE TONG}) missing from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 54633 (\uac00{HANGUL SYLLABLE HAB}) missing from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 50672 (\uac00{HANGUL SYLLABLE YEON}) missing from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 48708 (\uac00{HANGUL SYLLABLE BI}) missing from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 47784 (\uac00{HANGUL SYLLABLE MO}) missing from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 45944 (\uac00{HANGUL SYLLABLE DEL}) missing from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 49688 (\uac00{HANGUL SYLLABLE SU}) missing
```

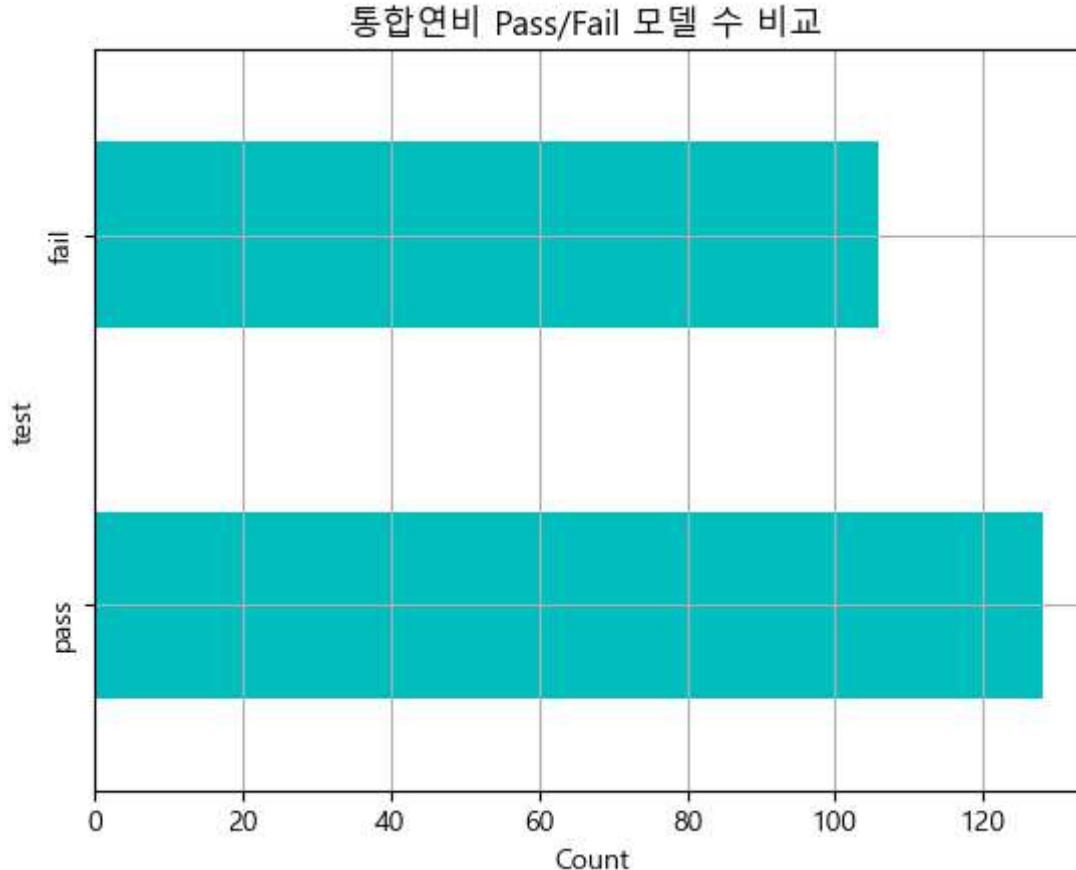
```
g from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 44368 (WN{HANGUL SYLLABLE GYO}) missing from current font.  
.. fig.canvas.print_figure(bytes_io, **kw)
```



맑은고딕 폰트 사용

```
In [22]: ## 한글 title 처리: 폰트 지정  
import matplotlib.pyplot as plt  
plt.rc('font', family='Malgun Gothic') #폰트 사용  
  
## 수평 막대로 그리기: .plot.barh()  
count_test.plot.barh(x='count', title="통합연비 Pass/Fail 모델 수 비교 ", xlabel='Count', grid=True, color='c', rot=90)
```

```
Out[22]: <Axes: title={'center': '통합연비 Pass/Fail 모델 수 비교 '}, xlabel='Count', ylabel='test'>
```



```
In [ ]:
```

▶ Pandas 라이브러리의 'Series' 객체에서 'plot.bar()' 메서드의 매개변수 확인

```
import pandas as pd  
  
bar_plotparams = pd.Series.plot.bar.\_doc_  
  
print(bar_plot_params)
```

```
In [23]: # plot.bar() 메서드의 매개변수 확인  
import pandas as pd
```

```
bar_plot_params = pd.Series.plot.bar.__doc__  
print(bar_plot_params)
```

Vertical bar plot.

A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the values that they represent. A bar plot shows comparisons among discrete categories. One axis of the plot shows the specific categories being compared, and the other axis represents a measured value.

Parameters

x : label or position, optional

.... Allows plotting of one column versus another. If not specified, the index of the DataFrame is used.

y : label or position, optional

.... Allows plotting of one column versus another. If not specified, all numerical columns are used.

color : str, array-like, or dict, optional

.... The color for each of the DataFrame's columns. Possible values are:

.... - A single color string referred to by name, RGB or RGBA code, for instance 'red' or '#a98d19'.

.... - A sequence of color strings referred to by name, RGB or RGBA code, which will be used for each column recursively. For instance ['green','yellow'] each column's bar will be filled in green or yellow, alternatively. If there is only a single column to be plotted, then only the first color from the color list will be used.

.... - A dict of the form {column name : color}, so that each column will be colored accordingly. For example, if your columns are called 'a' and 'b', then passing {'a': 'green', 'b': 'red'} will color bars for column 'a' in green and bars for column 'b' in red.

**kwargs

.... Additional keyword arguments are documented in

.... :meth:`DataFrame.plot`.

Returns

matplotlib.axes.Axes or np.ndarray of them

.... An ndarray is returned with one :class:`matplotlib.axes.Axes` per column when ``subplots=True``.

See Also

.. DataFrame.plot.bart : Horizontal bar plot.
.. DataFrame.plot : Make plots of a DataFrame.
.. matplotlib.pyplot.bar : Make a bar plot with matplotlib.

Examples

Basic plot.

.. plot::

:context: close-figs

```
>>> df = pd.DataFrame({'lab':['A', 'B', 'C'], 'val':[10, 30, 20]})  
>>> ax = df.plot.bar(x='lab', y='val', rot=0)
```

Plot a whole dataframe to a bar plot. Each column is assigned a distinct color, and each row is nested in a group along the horizontal axis.

.. plot::

:context: close-figs

```
>>> speed = [0.1, 17.5, 40, 48, 52, 69, 88]  
>>> lifespan = [2, 8, 70, 1.5, 25, 12, 28]  
>>> index = ['snail', 'pig', 'elephant',  
...           'rabbit', 'giraffe', 'coyote', 'horse']  
>>> df = pd.DataFrame({'speed': speed,  
...                      'lifespan': lifespan}, index=index)  
>>> ax = df.plot.bar(rot=0)
```

Plot stacked bar charts for the DataFrame

.. plot::

:context: close-figs

```
>>> ax = df.plot.bar(stacked=True)
```

Instead of nesting, the figure can be split by column with ``subplots=True``. In this case, a :class:`numpy.ndarray` of :class:`matplotlib.axes.Axes` are returned.

.. plot::

:context: close-figs

```
....     >>> axes = df.plot.bar(rot=0, subplots=True)
....     >>> axes[1].legend(loc=2) # doctest: +SKIP
.... 
....     If you don't like the default colours, you can specify how you'd
....     like each column to be colored.
.... 
....     .. plot::
....     :context: close-figs
.... 
....     >>> axes = df.plot.bar(
....         ... rot=0, subplots=True, color={"speed": "red", "lifespan": "green"}
....     )
....     >>> axes[1].legend(loc=2) # doctest: +SKIP
.... 
....     Plot a single column.
.... 
....     .. plot::
....     :context: close-figs
.... 
....     >>> ax = df.plot.bar(y='speed', rot=0)
.... 
....     Plot only selected categories for the DataFrame.
.... 
....     .. plot::
....     :context: close-figs
.... 
....     >>> ax = df.plot.bar(x='lifespan', rot=0)
```

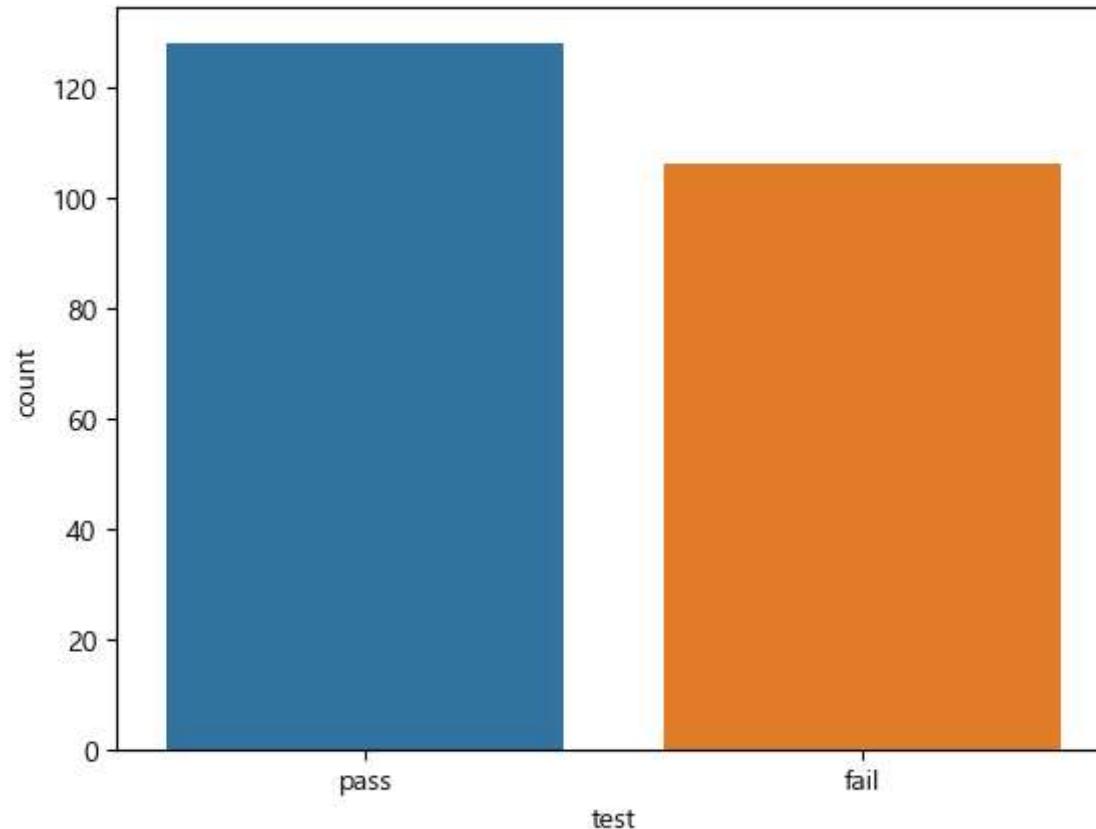
[실습 B] seaborn 라이브러리로 빈도수 막대 그래프 그리기

> 데이터프레임을 기반으로 메서드가 알아서 빈도수를 구해서 작성

```
In [27]: ## 빈도 막대 그래프 만들기: seaborn 라이브러리 활용
import seaborn as sns
sns.countplot(data = mpg, x = 'test') #x축에 빈도수 사용
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

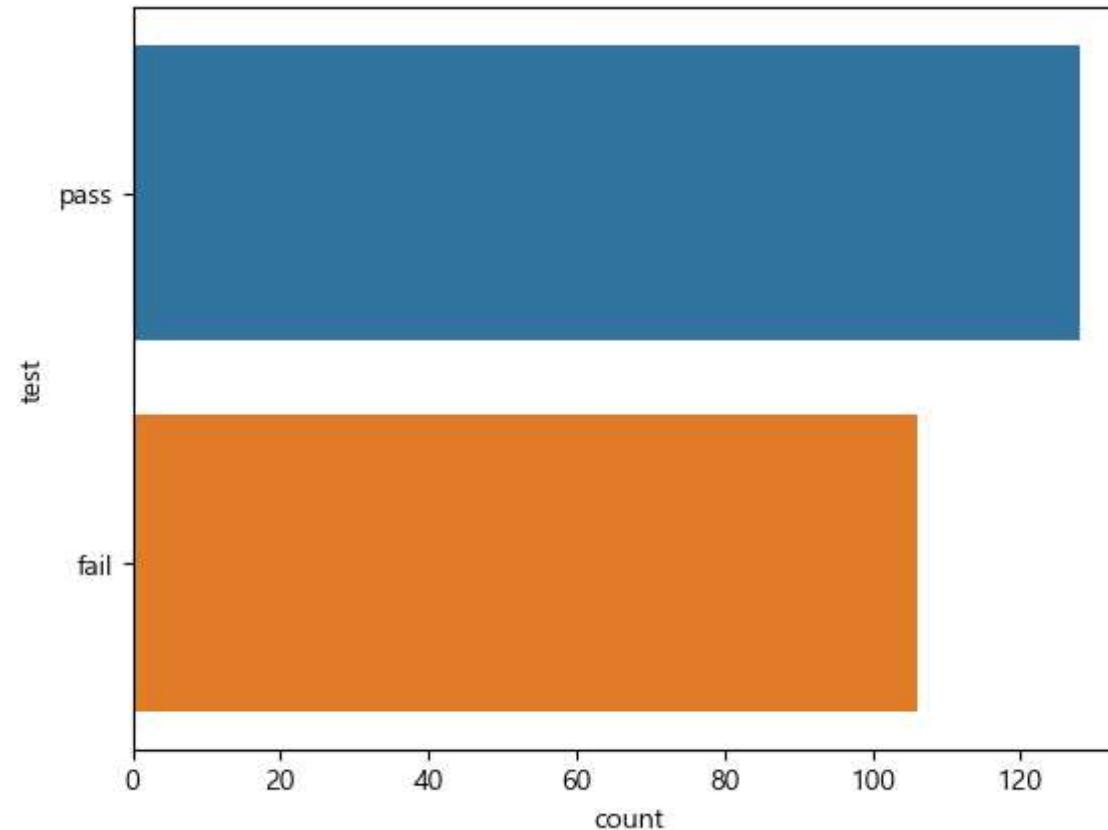
Out[27]: <Axes: xlabel='test', ylabel='count'>



In [26]: ## 빈도 막대 그래프 만들기: seaborn 라이브러리 활용
import seaborn as sns
sns.countplot(data = mpg, y = 'test') #y축에 빈도수 사용

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[26]: <Axes: xlabel='count', ylabel='test'>



In []:

[실습-3] numpy.newaxis() 중첩 사용 활용하기

1. 통합연비에 대한 등급 'grade' 파생변수 추가

>> 'grade'는 'total' ≥ 30 이면 'A' 등급, 아니고 'total' ≥ 20 이면 'B' 등급, 나머지는 'C' 등급

2. 등급 빈도 수 데이터 만들기

3. 빈도표 막대 그래프로 그리기

In [36]:

```
## [실습] 조건문을 활용해 파생변수 만들기
# 'total' >=30 이면 'A' 등급, 아니고 'total' >=20 이면 'B' 등급, 나머지는 'C' 등급
import numpy as np #수치연산 지원 패키지 numpy
x = np.where(mpg['total'] >= 30, 'A',
              np.where(mpg['total'] >=20, 'B', 'C')) #np.where(조건식, True 시 실행문, False 시 실행문)
x
```

Out[36]:

```
array(['B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'B', 'C', 'B', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'B',
       'B', 'C', 'C', 'C', 'C', 'B', 'B', 'B', 'B', 'B', 'B',
       'C', 'C', 'B', 'B', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C',
       'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C',
       'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C',
       'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C',
       'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'B',
       'B', 'B', 'B', 'C', 'C', 'C', 'C', 'C', 'A', 'B', 'B', 'B',
       'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C',
       'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C',
       'B', 'B', 'B', 'B', 'B', 'B', 'C', 'C', 'C', 'C', 'C', 'C',
       'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'A', 'A', 'A', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C', 'C',
       'B', 'B', 'B', 'B', 'A', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'A', 'A', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B'],
      dtype='|<U1')
```

In [37]:

```
## [실습] 조건문을 활용해 파생변수 만들기
#### 1. 'grade' 파생변수 추가
#### >> 'grade'는 'total' >=30 이면 'A' 등급, 아니고 'total' >=20 이면 'B' 등급, 나머지는 'C' 등급
import numpy as np #수치연산 지원 패키지 numpy
mpg['grade'] = np.where(mpg['total'] >= 30, 'A',
                       np.where(mpg['total'] >=20, 'B', 'C')) #np.where(조건식, True 시 실행문, False 시 실행문)
mpg.head()
```

Out[37]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category	total	test	grade
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact	23.5	pass	B
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact	25.0	pass	B
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact	25.5	pass	B
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact	25.5	pass	B
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact	21.0	pass	B

데이터프레임 개별 항목 값 조작

In [38]:

```
## [실습] 조건문을 활용해 파생변수 만들기
#### 2. 등급 빈도수 데이터 만들기
count_grade = mpg['grade'].value_counts() # 등급 빈도표 만들기 (default는 빈도수 높은 순으로 정렬 적용)
count_grade
```

Out[38]:

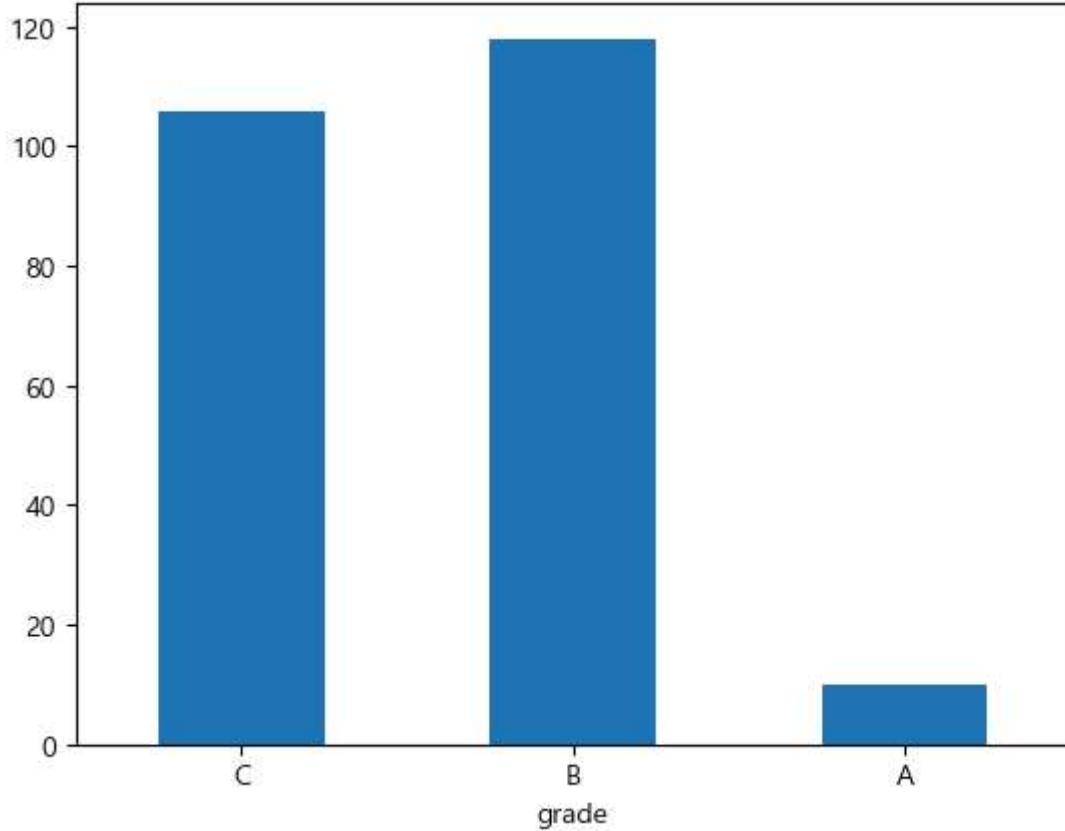
```
grade
B    118
C    106
A     10
Name: count, dtype: int64
```

In [42]:

```
## [실습] 조건문을 활용해 파생변수 만들기
## 3. 빈도 막대 그래프 그리기
count_grade.plot.bar(rot = 0) # 등급 빈도 막대 그래프 만들기
```

Out[42]:

```
<Axes: xlabel='grade'>
```



그래프에 막대 표현 순서 바꾸기

.value_counts()는 빈도수 순으로 데이터를 구성

> .sort_index()로 index 순으로 조정 가능

> 순서에 대한 리스트 값으로 강제 순서 지정 가능

```
In [40]: ## index 순 정렬 적용  
count_grade = mpg['grade'].value_counts().sort_index() # 등급 빈도표 만들기(index 순 정렬 적용)  
count_grade
```

```
Out[40]: grade  
A ... 10  
B ... 118  
C ... 106  
Name: count, dtype: int64
```

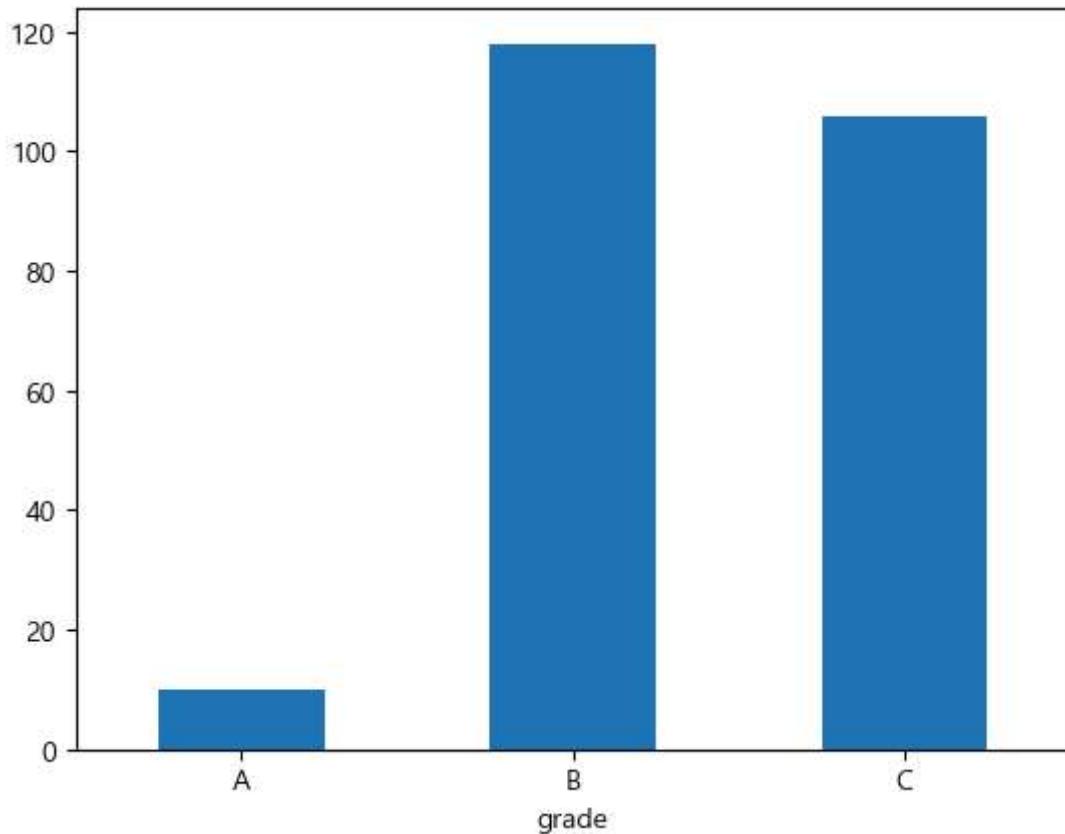
```
In [41]: ## 원하는 순서에 대한 리스트 값으로 강제 순서 지정  
count_grade = mpg['grade'].value_counts()  
new_order = ['C', 'B', 'A']      #List형으로 정렬 순서 지정  
count_grade = count_grade[new_order] # 등급 빈도표 만들기(지정한 순 정렬 적용)  
count_grade
```

```
Out[41]: grade  
C ... 106  
B ... 118  
A ... 10  
Name: count, dtype: int64
```

```
In [ ]:
```

```
In [43]: ## [실습] 조건문을 활용해 파생변수 만들기  
#### 3. 원하는 순으로 빈도표 막대 그래프로 그리기  
count_grade = mpg['grade'].value_counts() # 등급 빈도표 만들기 (default는 빈도수 높은 순으로 정렬 적용)  
new_order = ['A', 'B', 'C']      #List형으로 정렬 순서 지정  
count_grade = count_grade[new_order] # 등급 빈도표 만들기(지정한 순 정렬 적용)  
count_grade.plot.bar(rot = 0) # 등급 빈도 막대 그래프 만들기
```

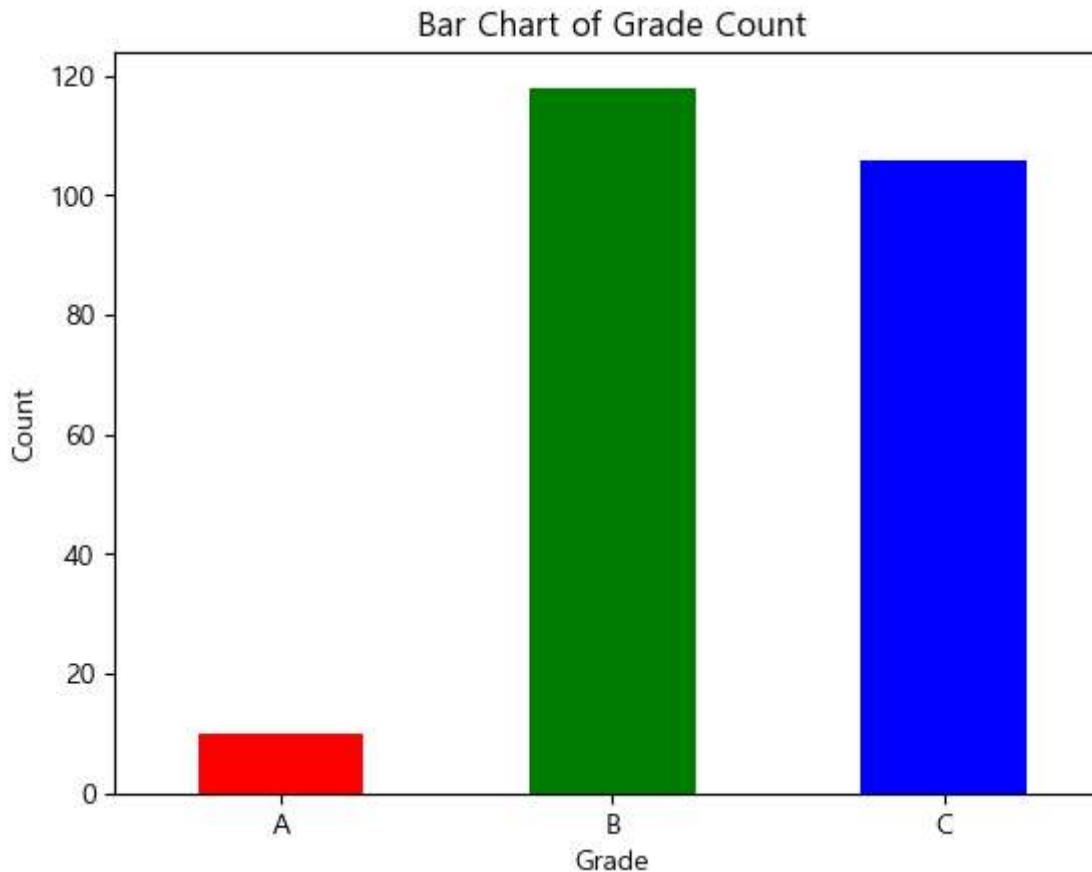
```
Out[43]: <Axes: xlabel='grade'>
```



그래프 막대 색상 지정하기

```
In [50]: clist = ['red', 'green', 'blue']
title = 'Bar Chart of Grade Count'
xlbl = 'Grade'
y	lbl = 'Count'
count_grade.plot.bar(rot=0, color=clist, title=title, xlabel=xlbl, ylabel=y_lbl)
```

```
Out[50]: <Axes: title={'center': 'Bar Chart of Grade Count'}, xlabel='Grade', ylabel='Count'>
```



A: 막대 위에 값 표기하기

```
In [51]: ## [실습] 조건문을 활용해 파생변수 만들기
#### 3. 빈도표 막대 그래프로 그리기
#### >> 'grade' 순으로 표현
#### >> color=['red', 'green', 'blue'] 순으로 사용
#### [방법-A] 그래프 객체 ax를 만들 때 특성 값 지정
count_grade = mpg['grade'].value_counts().sort_index() # 등급 빈도표 만들기(index 순 정렬 적용)

# 막대 그래프 그리기
clist = ['red', 'green', 'blue']
title = 'Bar Chart of Grade Count'
xlbl = 'Grade'
ylbl = 'Count'
```

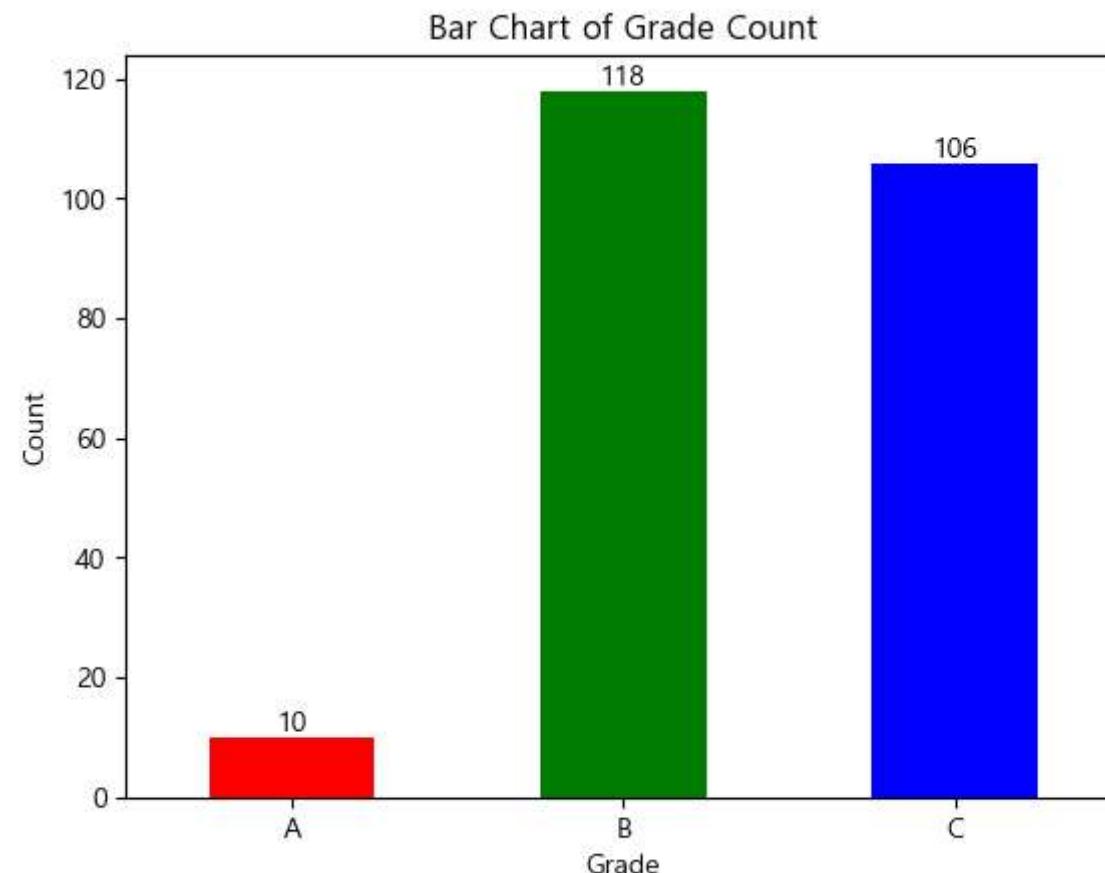
```

ax = count_grade.plot.bar(rot=0, color=clist, title=title, xlabel=xlbl, ylabel=ylbl)

# 막대 위에 값 표기
for idx, value in enumerate(count_grade):
    ax.text(idx, value + 1, str(value), ha='center')

# 그래프 표시
import matplotlib.pyplot as plt
plt.show()

```



B: 막대 위에 값 표기하기

In [52]: ## [실습] 조건문을 활용해 파생변수 만들기
3. 빈도표 막대 그래프로 그리기

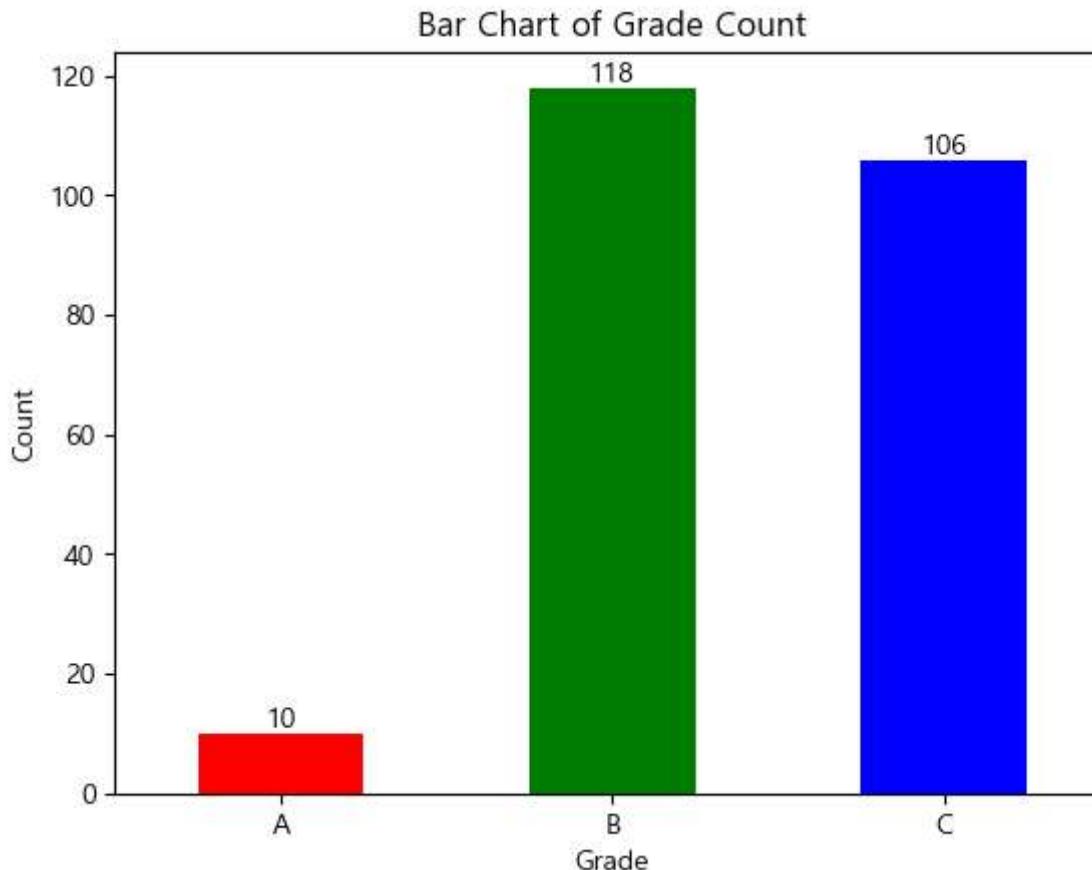
```
#### >> 'grade' 순으로 표현
#### >> color=['red', 'green', 'blue'] 순으로 사용
#### [방법-B] 그래프 객체 ax를 만들고, 나중에 특성 값 변경
count_grade = mpg['grade'].value_counts().sort_index() # 등급 빈도표 만들기(index 순 정렬 적용)

# 막대 그래프 그리기
ax = count_grade.plot.bar(rot = 0, color=['red', 'green', 'blue'])

# 막대 위에 값 표기
for idx, value in enumerate(count_grade):
    ax.text(idx, value + 1, str(value), ha='center')

# 그래프에 제목과 레이블 추가
ax.set_title('Bar Chart of Grade Count')
ax.set_xlabel('Grade')
ax.set_ylabel('Count')

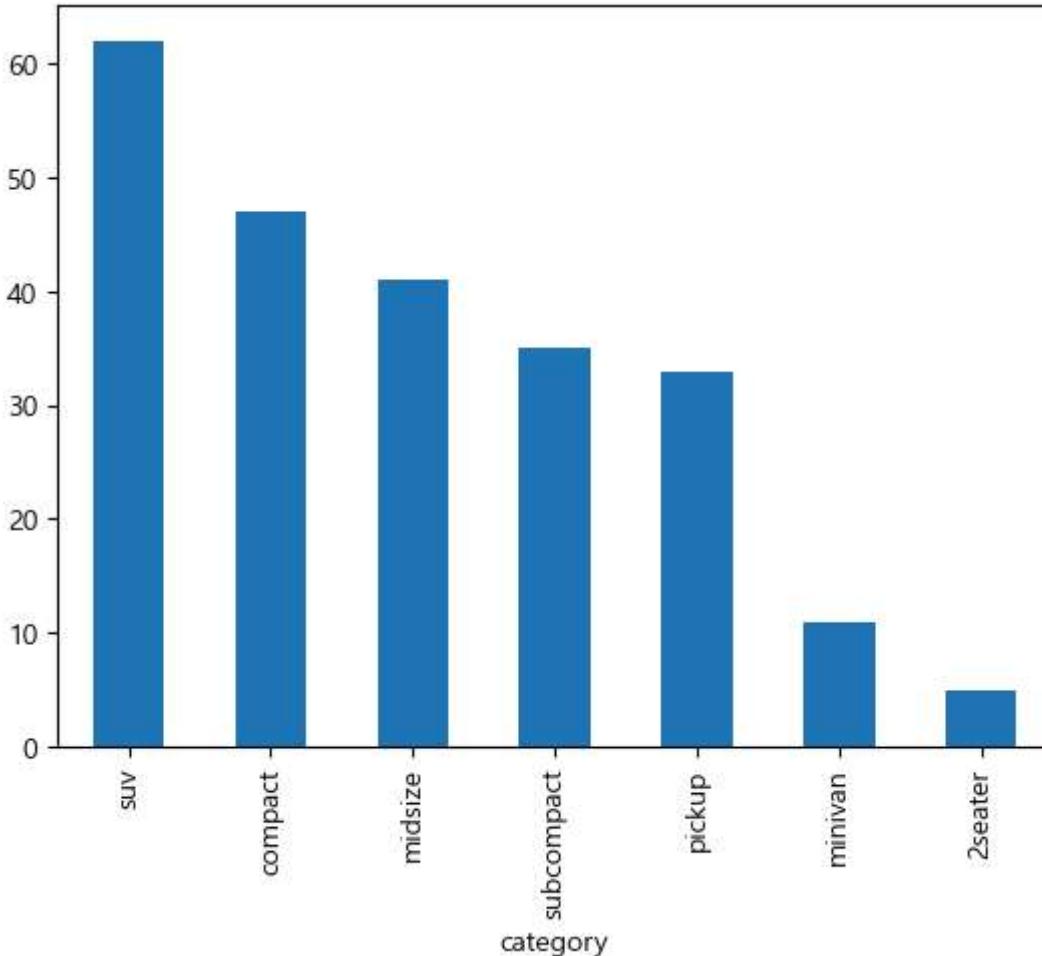
# 그래프 표시
import matplotlib.pyplot as plt
plt.show()
```



[실습] 'category'에 대한 빈도수 그래프 그리기

```
In [53]: ## [실습] 조건문을 활용해 파생변수 만들기  
count_cat = mpg['category'].value_counts() # 등급 빈도표 만들기  
count_cat  
count_cat.plot.bar() # rot는 반시계방향 각도
```

```
Out[53]: <Axes: xlabel='category'>
```



In []:

[과제] excel > csv > Data Frame > Data Frame 파생변수 추가 > 그래프 그리기

1. excel > csv 파일 형식 변환

> LMS에 제시된 엑셀 파일 Test_result.xlsx를 CSV 타입 파일로 변환

2. csv > Data Frame 데이터 변환

> 변환한 CSV 파일을 읽어 데이터 프레임으로 변환

> 변환된 데이터 프레임의 정보 확인

> Index 컬럼 'Unnamed: 0' 제거

3. Data Frame에 'Total' 파생 변수를 추가

> 'Total'은 총점으로 (Attendance + Homework + Midterm + Final)

4. Data Frame에 'grade' 파생 변수를 추가

> 'Grade'는 총점에 대한 등급 : "A+" : 100-95, "A" : 94-90, "B+" : 89-85, "B" : 84-80,

"C+" : 79-75, "C" : 74-70, "D+" : 69-65, "D" : 64-60, "F" : 59-0

5. 파생 변수로 그래프 그리기

> 'Grade'별 인원 수 분포 구하고, 확인하기

> 'Grade'별 인원 수 막대 그래프 그리기

> x-축을 높은 등급 순으로 그리기

>>[도전] 막대에 빈도 값, 표 타이틀, 축 레이블 표시하기

6. 데이터 프레임을 Test_result2.csv 파일로 변환

> index는 제외하고 변환

In []:

정리하기

In [1]:

```
# 1. 패키지 로드
import pandas as pd
import numpy as np

# 2. 데이터 불러오기
mpg = pd.read_csv('mpg.csv')

# 3. 데이터 파악하기
mpg.head()      # 데이터 앞부분
mpg.tail()      # 데이터 뒷부분
```

```

mpg.shape      # 행, 열 수
mpg.info()     # 속성
mpg.describe() # 요약 통계량

# 4. 변수명 바꾸기
mpg = mpg.rename(columns = {'manufacturer' : 'company'})

# 5. 파생변수 만들기
mpg['total'] = (mpg['cty'] + mpg['hwy'])/2           # 변수 조합
mpg['test'] = np.where(mpg['total'] >= 20, 'pass', 'fail') # 조건문 활용

# 6. 빈도 확인하기
count_test = mpg['test'].value_counts() # 빈도표 만들기
count_test.plot.bar(rot = 0)           # 빈도 막대 그래프 만들기

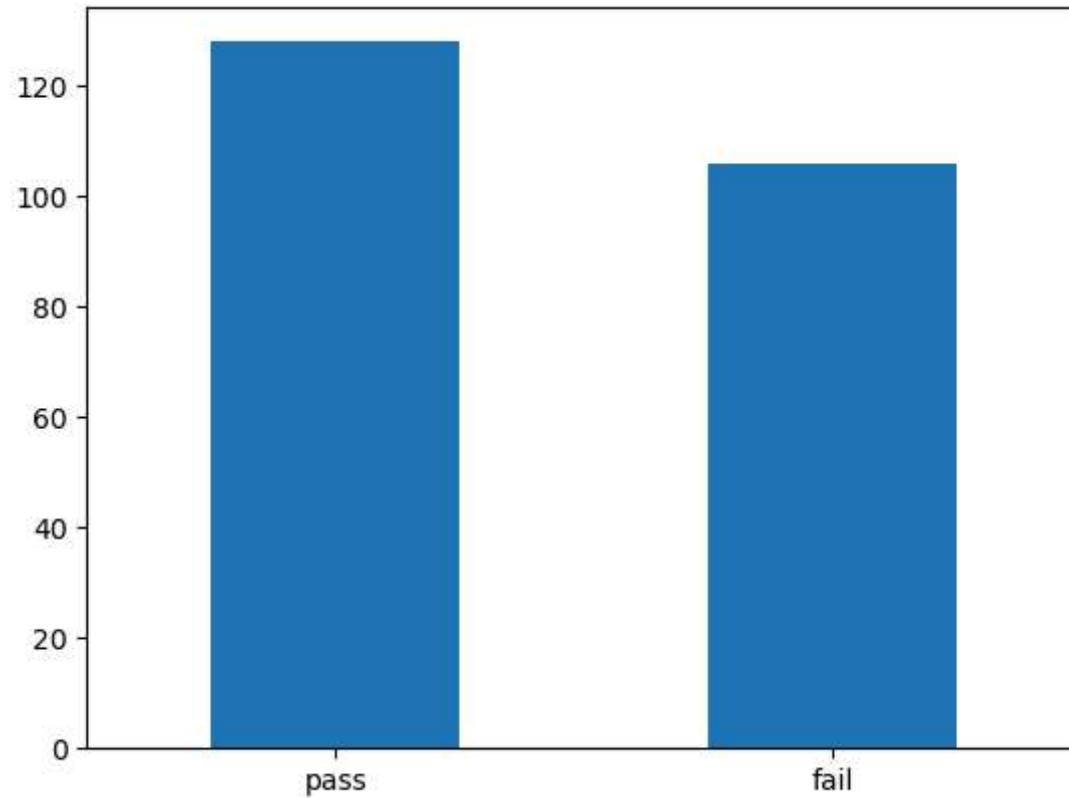
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   manufacturer    234 non-null   object 
 1   model          234 non-null   object 
 2   displ          234 non-null   float64
 3   year           234 non-null   int64  
 4   cyl            234 non-null   int64  
 5   trans          234 non-null   object 
 6   drv             234 non-null   object 
 7   cty            234 non-null   int64  
 8   hwy            234 non-null   int64  
 9   fl              234 non-null   object 
 10  category        234 non-null   object 
dtypes: float64(1), int64(4), object(6)
memory usage: 20.2+ KB

```

Out[1]:



In []: