	개수	비모수		모수
		서열	명목	등간/비율
단일	1	부호검정, 부호순위검정	Run	one-sample t-test
대응	2		맥니머	paired t-test
	k	프리드먼	코크란 Q	ANOVA
독립	2	윌콕슨 순위합검정 (만위트니 U)	카이제곱	independent t-test
	k	크루스칼 월리스		ANOVA

*부호검정: n ≤ 100 – 이항분포, **n > 100 – 정규분포**

*부호순위검정: n ≤ 20 – 이항분포, **n > 20 – 정규분포**

*윌콕슨 순위합검정: n ≤ 25 – 이항분포, **n > 25 – 정규분포**

*프리드먼: 3개 범주 – 9개 이하, 4개 범주 – 5개 이하 이외는 카이제곱

*크루스칼 월리스: 3개 범주 – 15개 이하, 4개 범주 – 14개 이하 이외는 카이제곱

***카이제곱: 각 셀 최소 5개 이상**

일표본 t-test 가정

1. 정규성을 가진다 -> 정규성검정 scipy.stats.shapiro H0: 정규성가짐
2. 독립성 -> 같은 측정 두번이면 대응표본 t-test
3. 연속형 변수

고양이들의 평균 몸무게가 2.6 인지 아닌지

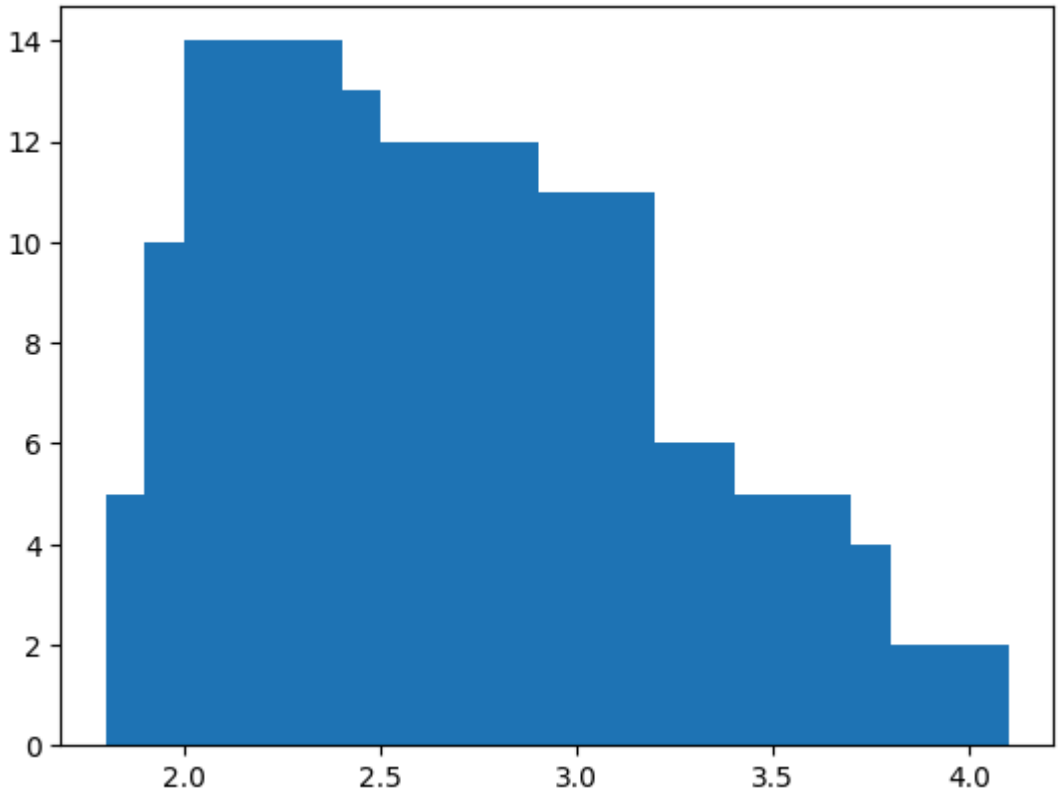
In [3]:
```
import pandas as pd
cats=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/cats.csv")
cats.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Sex     144 non-null    object 
 1   Bwt     144 non-null    float64
 2   Hwt     144 non-null    float64
dtypes: float64(2), object(1)
memory usage: 3.5+ KB
```

In [ ]:

```python
import scipy.stats as stats
from scipy.stats import shapiro

mu=2.6
#정규성 검정
print(shapiro(cats['Bwt']))
#정규분포 가 아니므로 윌콕슨 진행  정규 분포면 t-test stats.ttest_1samp()
print(stats.wilcoxon(cats.Bwt-mu,alternative='two-sided'))
import matplotlib.pyplot as plt
cats_Bwt_cnt = pd.value_counts(cats['Bwt'].values,sort=False)
width=0.4
plt.bar(cats_Bwt_cnt.index,cats_Bwt_cnt.values,width)
plt.show()
```

```
ShapiroResult(statistic=0.9518786668777466, pvalue=6.730254972353578e-05)
WilcoxonResult(statistic=3573.0, pvalue=0.02524520294814093)
/var/folders/hv/lqp1gn9n1ll0lbh2pfzn9pww0000gn/T/ipykernel_7310/899596944.py:10: FutureWarning: pandas.value_counts is deprecate
d and will be removed in a future version. Use pd.Series(obj).value_counts() instead.
  cats_Bwt_cnt = pd.value_counts(cats['Bwt'].values,sort=False)
```



대응표본 T test 정규성 검정: 차이값(d = X_1 - X_2)에 대해 수행

In [ ]:
```
import pandas as pd
data={'before':[7,3,4,5,2,1,6,6,5,4],
    'after':[8,4,5,6,2,3,6,8,6,5]}
data=pd.DataFrame(data)
from scipy.stats import shapiro
diff = data.before.values - data.after.values
print(shapiro(diff))  # p > 0.05 → 정규성 만족
print(stats.ttest_rel(data.after,data.before,alternative='greater'))#정규성
stat, p = stats.wilcoxon(data.after,data.before,alternative='greater')#정규성 아닐때
print(f"stat = {stat:.3f}, p-value = {p:.3f}")
```

```
ShapiroResult(statistic=0.8148399591445923, pvalue=0.021947935223579407)
TtestResult(statistic=4.743416490252569, pvalue=0.0005269356285082764, df=9)
stat = 36.000, p-value = 0.004
```

```
/opt/homebrew/Caskroom/miniforge/base/envs/general/lib/python3.11/site-packages/scipy/stats/_morestats.py:4088: UserWarning: Exa
ct p-value calculation does not work if there are zeros. Switching to normal approximation.
  warnings.warn("Exact p-value calculation does not work if there are "
/opt/homebrew/Caskroom/miniforge/base/envs/general/lib/python3.11/site-packages/scipy/stats/_morestats.py:4102: UserWarning: Sam
ple size too small for normal approximation.
  warnings.warn("Sample size too small for normal approximation.")
```

독립표본 t-test

1. 두모집단은 정규성만족
2. 두모집단은 분산이 같아야한다.

이름	설명	비교
Wilcoxon Signed-Rank Test	✅ 대응표본용 비모수 검정	**대응표본 t-test 대체**
Mann-Whitney U Test	✅ **독립표본용** 비모수 검정	독립표본 t-test 대체

검정 이름	조건	사용 상황
Levene 검정	정규성 없어도 사용 가능	**가장 일반적으로 많이 사용**
Bartlett 검정	정규성 만족해야 함	정규성 확신할 때만 사용
F 검정 (F-test)	두 집단 비교, 정규성 가정	가장 간단한 형태지만 실무에선 잘 안 씀

🧠 1️⃣ 핵심 비교표

구분	윌콕슨 부호순위 검정 (Wilcoxon Signed-Rank Test)	윌콕슨 순위합 검정 / 맨-휘트니 U (Wilcoxon Rank-Sum / Mann–Whitney U Test)
데이터 형태	**대응표본(paired samples)**	**독립표본(independent samples)**
목적	**두 관련된 집단(같은 대상 전후 비교)** 의 차이 검정	**두 독립된 집단의 중앙값 차이** 검정
모수 대응검정	대응표본 t-test (paired t-test)	독립표본 t-test (independent t-test)
가정	1. 자료는 쌍(pair)로 존재 2. 차이는 대칭적이어야 함	1. 두 집단 독립 2. 분포 모양은 동일(중앙만 다름)
검정 통계량	차이값의 **부호(+/-)** 와 **순위**를 함께 이용	두 집단의 순위를 합산하여 비교
귀무가설 (H_0)	"두 시점의 중앙값 차이가 0이다"	"두 집단의 분포가 동일하다 (중앙값 같음)"
데이터 예시	같은 사람의 **치료 전·후 혈압**	서로 다른 그룹의 **남/여 혈압 비교**
대표 함수	`scipy.stats.wilcoxon(x1, x2)`	`scipy.stats.mannwhitneyu(x1, x2)`

In [ ]:
```python
# *일표본 윌콕슨 부호순위 검정 (p.203)
import numpy as np
from scipy.stats import wilcoxon

work_hours = [203, 204, 197, 195, 201, 205, 198, 199, 194, 207]
baseline = 200

diff = np.array(work_hours) - baseline

stat, p_value = wilcoxon(diff)

print(f"윌콕슨 검정 통계량: {stat}")
```

```
print(f"p-값: {p_value:.4f}")
#*이표본 윌콕슨 부호순위 검정
import numpy as np
from scipy.stats import wilcoxon

before = [120, 130, 115, 140, 135, 128, 132, 110, 125, 137]
after = [115, 128, 118, 135, 130, 124, 131, 109, 120, 133]

stat, p_value = wilcoxon(before, after)

print(f"윌콕슨 검정 통계량: {stat}")
print(f"p-값: {p_value:.4f}")
```

- **`ranksums`**는 정규 근사를 사용하여 통계량을 계산하므로 큰 표본에서는 매우 유사한 결과를 나타냅니다. 그러나 **작은 표본**에서는 정규 근사가 정확하지 않을 수 있어 p-값에서 차이가 발생할 수 있습니다.
- **`mannwhitneyu`**는 각 데이터 쌍을 비교하여 U 통계량을 계산하므로, 작은 샘플에서도 잘 작동합니다. 작은 데이터 세트에서는 **U 통계량**을 사용한 맨-휘트니 U 검정이 더 정확할 수 있습니다.

In [ ]:
```
# *윌콕슨 순위합 검정

# 두 독립된 그룹 간의 차이를 비교하는 비모수 검정
# 독립표본 t-검정의 비모수 버전
from scipy.stats import ranksums, mannwhitneyu

group_a = [4.1, 3.5, 5.0, 4.8, 3.8]
group_b = [3.9, 4.2, 3.1, 4.0, 3.6]

stat, p_value = ranksums(group_a, group_b)
print(f"윌콕슨 순위합 검정 통계량: {stat}")
print(f"p-값: {p_value:.4f}")

stat, p_value = mannwhitneyu(group_a, group_b)
print(f"맨-휘트니 U 통계량: {stat}")
print(f"p-값: {p_value:.4f}")
```

In [11]:
```
import pandas as pd
cats=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/cats.csv")
```

```
cats.info()
female=cats.loc[cats.Sex=='F','Bwt']
male=cats.loc[cats.Sex=='M','Bwt']
print(stats.levene(female,male))
```

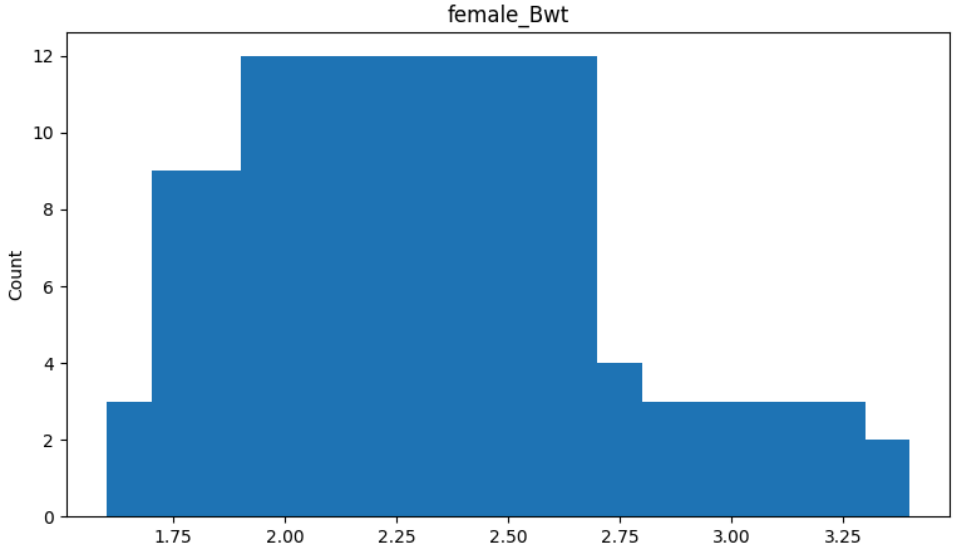
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Sex     144 non-null    object 
 1   Bwt     144 non-null    float64
 2   Hwt     144 non-null    float64
dtypes: float64(2), object(1)
memory usage: 3.5+ KB
LeveneResult(statistic=19.43101190877999, pvalue=2.0435285255189404e-05)
```

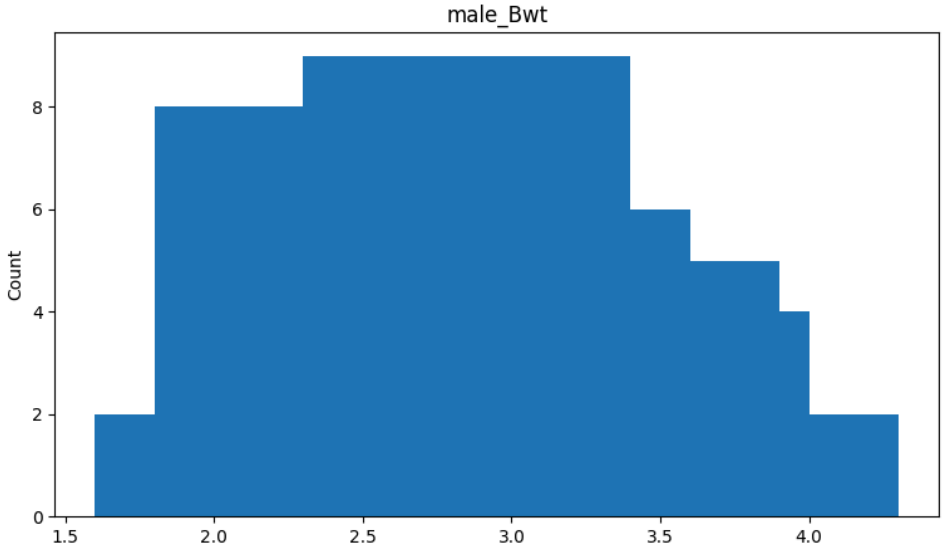
In [14]:
```
print(stats.ttest_ind(female,male,equal_var=False))# pvalue 작으니까 등분산이아님
print(female.mean())
print(male.mean())
print('pvalue가 작고 두개 평균ㅜ ㄴ 차이 존재 ')
stat, p = stats.mannwhitneyu(female, male, alternative='two-sided')

print(f"U-statistic = {stat:.3f}")
print(f"p-value     = {p:.4f}")
```

```
TtestResult(statistic=-8.70948849909559, pvalue=8.831034455859356e-15, df=136.83788299625363)
2.359574468085107
2.8999999999999995
pvalue가 작고 두개 평균ㅜ ㄴ 차이 존재
U-statistic = 757.500
p-value     = 0.0000
```

In [16]:
```
female_Bwt_cnt=pd.value_counts(female.values,sort=False)
male_Bwt_cnt=pd.value_counts(male.values,sort=False)
fig,ax=plt.subplots(1,2,figsize=(20,5))
width=0.4
ax[0].bar(female_Bwt_cnt.index,female_Bwt_cnt.values)
ax[0].set_title('female_Bwt')
ax[0].set_ylabel('Count')
ax[1].bar(male_Bwt_cnt.index,male_Bwt_cnt.values)
ax[1].set_title('male_Bwt')
ax[1].set_ylabel('Count')
plt.show()
```

```
/var/folders/hv/lqp1gn9n1ll0lbh2pfzn9pww0000gn/T/ipykernel_7310/3147949573.py:1: FutureWarning: pandas.value_counts is deprecate
d and will be removed in a future version. Use pd.Series(obj).value_counts() instead.
  female_Bwt_cnt=pd.value_counts(female.values,sort=False)
/var/folders/hv/lqp1gn9n1ll0lbh2pfzn9pww0000gn/T/ipykernel_7310/3147949573.py:2: FutureWarning: pandas.value_counts is deprecate
d and will be removed in a future version. Use pd.Series(obj).value_counts() instead.
  male_Bwt_cnt=pd.value_counts(male.values,sort=False)
```





분산분석 🔍 사전 검토 시 확인해야 할 항목 데이터 분포 확인 각 집단별 히스토그램, Q-Q plot 또는 scipy.stats.shapiro()로 정규성 검정

등분산 검정 scipy.stats.levene(group1, group2, ...) p ≥ 0.05 → 등분산 만족

독립성 확보 실험 설계 단계에서 고려

설문/실험에서 서로 영향을 안 주도록 구성

가정 위반	대처법
정규성 위반	→ **Kruskal-Wallis test** (비모수 ANOVA)
등분산성 위반	→ **Welch's ANOVA**
독립성 위반	→ 재설계 또는 repeated measures ANOVA (대응표본)

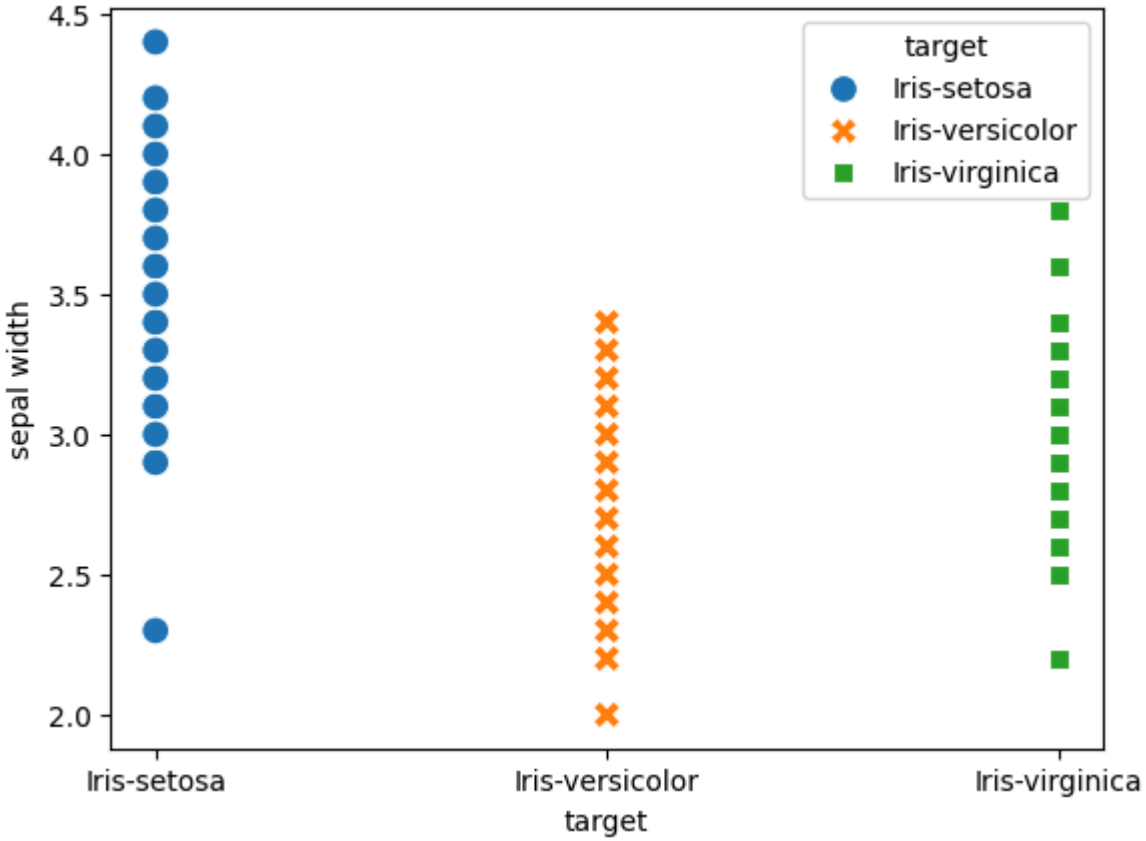
사후검정 던칸 MRT,LSD, 튜키 HSD

In [17]:
```
import scipy.stats as stats
import pandas as pd
```

```
Iris_data=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/iris.csv")
Iris_data["target"].unique()
```

Out[17]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [18]:
```
target_list=Iris_data["target"].unique()
setosa=Iris_data[Iris_data["target"]==target_list[0]]['sepal width']
versicolor=Iris_data[Iris_data["target"]==target_list[1]]['sepal width']
virginica=Iris_data[Iris_data["target"]==target_list[2]]['sepal width']

import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(x='target',y='sepal width',hue="target",style="target",s=100,data=Iris_data)
plt.show()
```



In [19]:
```
print(stats.shapiro(setosa))
print(stats.shapiro(versicolor))
print(stats.shapiro(virginica))
```

```
ShapiroResult(statistic=0.9686915278434753, pvalue=0.2046491503715515)
ShapiroResult(statistic=0.9741330742835999, pvalue=0.3379890024662018)
ShapiroResult(statistic=0.9673907160758972, pvalue=0.1808987259864807)
```

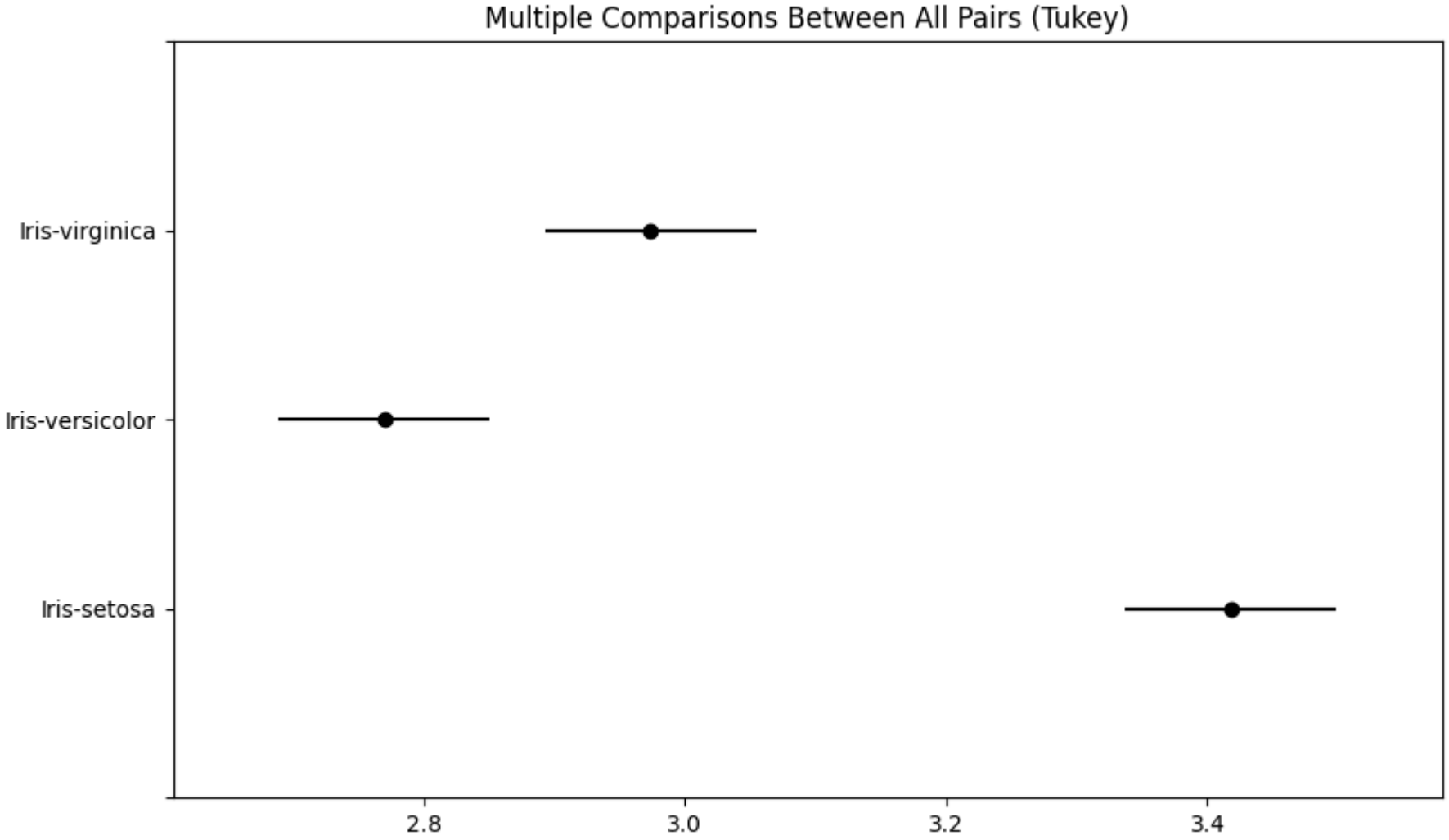
In [20]:
```
stats.levene(setosa,versicolor,virginica)
```

Out[20]:
```
LeveneResult(statistic=0.6475222363405327, pvalue=0.5248269975064537)
```

In [21]:
```
stats.f_oneway(setosa,versicolor,virginica)
```

Out[21]:
```
F_onewayResult(statistic=47.36446140299382, pvalue=1.3279165184572242e-16)
```

In [23]:
```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.multicomp import MultiComparison
mc=MultiComparison(data=Iris_data["sepal width"],groups=Iris_data['target'])
tuekyehsd=mc.tukeyhsd(alpha=0.05)
fig=tuekyehsd.plot_simultaneous()
plt.show()
```



In [ ]:
```
tuekyehsd.summary()
# meandiff = A - B
# → 항상 앞 그룹에서 뒤 그룹의 평균을 뺀 값이에요.
# 부호(+/-)는 어느 그룹이 더 큰지를 알려줍니다.
# 양수: group1이 group2보다 큼
# 음수: group1이 group2보다 작음
```

Out[ ]:

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
Iris-setosa	Iris-versicolor	-0.648	0.0	-0.8092	-0.4868	True
Iris-setosa	Iris-virginica	-0.444	0.0	-0.6052	-0.2828	True
Iris-versicolor	Iris-virginica	0.204	0.009	0.0428	0.3652	True

✅ 모델 수식 (수학적 표현)

이원 분산분석 모델 (교호작용 포함):

$$Y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}$$

- Y_{ijk}: i번째 A 수준, j번째 B 수준, k번째 반복에서의 관측값
- μ: 전체 평균
- α_i: 요인 A의 수준 i의 효과
- β_j: 요인 B의 수준 j의 효과
- $(\alpha\beta)_{ij}$: A와 B의 **상호작용 효과**
- ε_{ijk}: 오차 (정규성, 등분산성, 독립성 가정)

✅ 분산 분석표 구성 (ANOVA table)

Source	Sum of Squares (SS)	df	Mean Square (MS)	F-value
요인 A	SS_A	$a-1$	MS_A	$F_A = \frac{MS_A}{MS_E}$
요인 B	SS_B	$b-1$	MS_B	$F_B = \frac{MS_B}{MS_E}$
상호작용 $A \times B$	SS_{AB}	$(a-1)(b-1)$	MS_{AB}	$F_{AB} = \frac{MS_{AB}}{MS_E}$
오차 (Residual)	SS_E	$ab(n-1)$	MS_E	
전체	SS_T	$abn-1$		

- a: 요인 A의 수준 수
- b: 요인 B의 수준 수
- n: 각 조합에서 반복 측정 수

✅ 귀무가설 (각각의 검정 대상)

검정 대상	귀무가설 H_0	대립가설 H_1
요인 A의 주효과	$\alpha_i = 0 \forall i$	$\exists i, \alpha_i \neq 0$
요인 B의 주효과	$\beta_j = 0 \forall j$	$\exists j, \beta_j \neq 0$
상호작용 효과	$(\alpha\beta)_{ij} = 0 \forall i, j$	$\exists i, j, (\alpha\beta)_{ij} \neq 0$

In [ ]:
```python
# *크루스칼 월리스 (p.208)

# 세 개 이상의 독립된 그룹 간의 중앙값 차이를 비교하는 비모수 검정
from scipy.stats import kruskal
import scikit_posthocs as sp

group_a = [5.1, 4.8, 6.2, 5.0, 5.3]
group_b = [4.5, 4.9, 5.1, 4.6, 4.8]
group_c = [6.0, 5.9, 6.3, 6.1, 6.4]

stat, p_value = kruskal(group_a, group_b, group_c)

print(f"크러스칼-월리스 검정 통계량: {stat}")
print(f"p-값: {p_value:.4f}")

data = np.array(group_a + group_b + group_c)
groups = ['A']*len(group_a) + ['B']*len(group_b) + ['C']*len(group_c)

dunn_result = sp.posthoc_dunn([group_a, group_b, group_c], p_adjust = 'bonferroni')
dunn_result

import numpy as np
from scipy.stats import friedmanchisquare
import scikit_posthocs as sp
#→ 사후검정 – Dunn's test
method_a = [85, 78, 92, 88]
```

```
method_b = [90, 85, 95, 89]
method_c = [88, 82, 91, 87]

stat, p_value = friedmanchisquare(method_a, method_b, method_c)

print(f"프리드먼 검정 통계량: {stat}")
print(f"p-값: {p_value:.4f}")

# 데이터를 하나로 결합하여 사후 검정을 수행
data = np.array([method_a, method_b, method_c]).T  # Transpose to match students by rows
dunn_result = sp.posthoc_dunn(data, p_adjust='bonferroni')
print(dunn_result)
```

```
프리드먼 검정 통계량: 6.0
p-값: 0.0498
          1         2         3         4
1  1.000000  0.933085  0.669982  1.000000
2  0.933085  1.000000  0.015619  0.837838
3  0.669982  0.015619  1.000000  0.750256
4  1.000000  0.837838  0.750256  1.000000
```

- **p-값 < 0.05**: 세 학습 방법 간 점수에 통계적으로 유의미한 차이가 있음을 의미합니다. 즉, 세 방법 중 하나 이상이 다른 방법들과 차이가 있다고 할 수 있습니다.
- **p-값 > 0.05**: 세 학습 방법 간 차이가 통계적으로 유의하지 않음을 의미합니다. 즉, 세 방법이 비슷한 성과를 나타낸다고 해석할 수 있습니다.

In [26]:

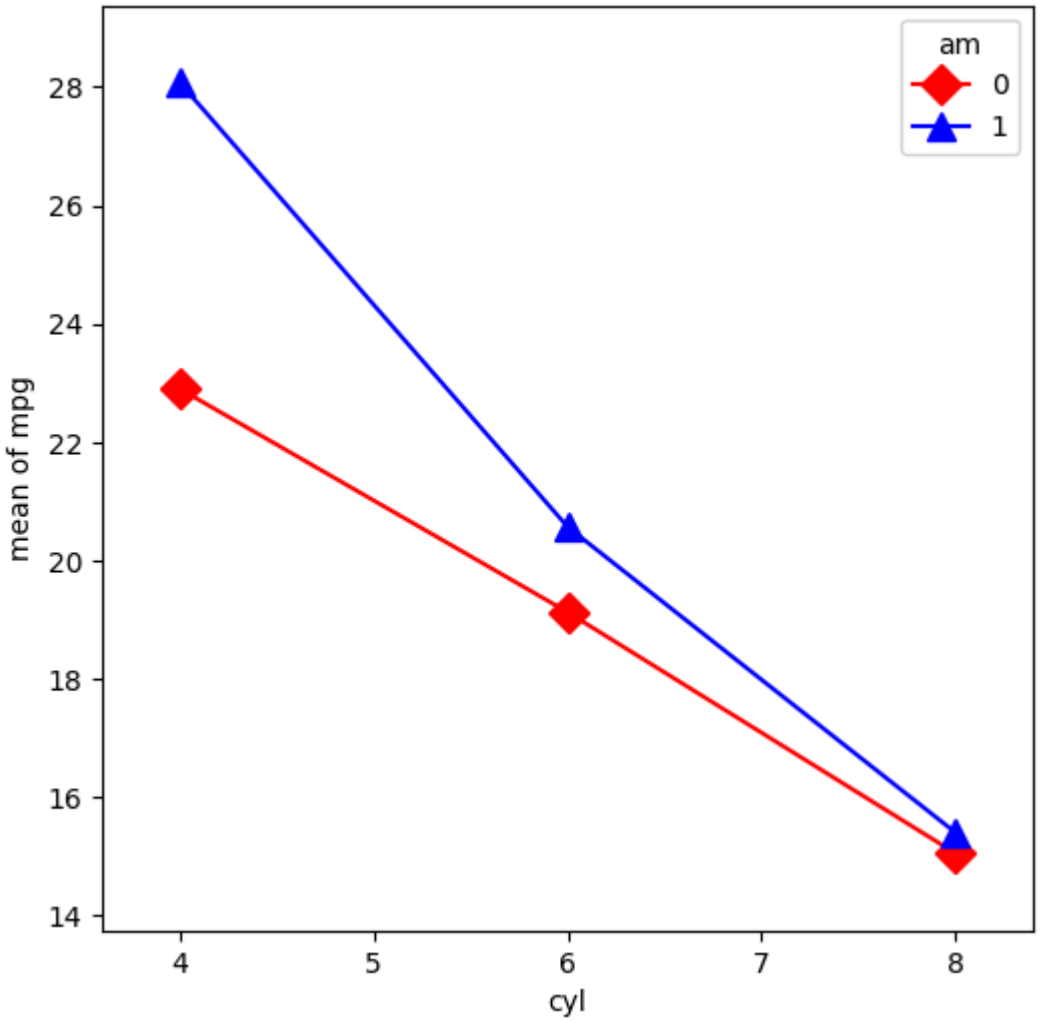
```
import pandas as pd
mtcars=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/mtcars.csv")
mtcars=mtcars[["mpg","am","cyl"]]
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
formula="mpg~C(cyl)+C(am)+C(cyl):C(am)"
model=ols(formula,mtcars).fit()
aov_table=anova_lm(model,typ=2)
aov_table
```

Out[26]:

	sum_sq	df	F	PR(>F)
C(cyl)	456.400921	2.0	24.819011	9.354735e-07
C(am)	36.766919	1.0	3.998759	5.608373e-02
C(cyl):C(am)	25.436511	2.0	1.383233	2.686140e-01
Residual	239.059167	26.0	NaN	NaN

In [30]:

```
from statsmodels.graphics.factorplots import interaction_plot
import matplotlib.pyplot as plt

cyl=mtcars["cyl"]
am= mtcars["am"]
mpg=mtcars["mpg"]
fig,ax=plt.subplots(figsize=(6,6))
fig=interaction_plot(cyl,am,mpg, colors=["red","blue"],markers=["D","^"],ms=10,ax=ax)
plt.show()
```



교차 분석

검정 이름	목적	주로 사용 예시
① **적합도 검정**	관측 분포가 기대 분포에 **적합**하는지 검정	동전이 공정한지, 주사위가 치우쳤는지
② **독립성 검정**	두 범주형 변수 사이에 **독립/관련성**이 있는지 검정	성별 vs 구매여부, 학력 vs 직업
③ **동질성 검정**	여러 집단의 범주형 분포가 **동일한지** 비교	지역별 정치 성향, 브랜드별 선호도

2. 표본 수가 충분히 커야 함 ✅

기대빈도(E)가 너무 작으면 카이제곱 근사가 부정확

2. 표본 수가 충분히 커야 함 ✅

- 기대빈도(E)가 너무 작으면 **카이제곱 근사가 부정확**

통상 기준:

- 기대빈도 $E_{ij} \geq 5$: 대부분의 셀에서 만족해야 함
- 20% 이상의 셀에서 기대빈도가 5 미만이면 → **Fisher의 정확검정** 추천

통상 기준:

🔶 **예제**

예를 들어 아래와 같은 교차표가 있다면:

	남자	여자	합계
좋아함	30	20	50
싫어함	10	40	50
합계	40	60	100

예를 들어, [좋아함 & 남자] 셀의 기대도수 E_{11} 은:

$$E_{11} = \frac{(\text{좋아함 행합}) \times (\text{남자 열합})}{\text{전체합}} = \frac{50 \times 40}{100} = 20$$

이런 식으로 **각 셀마다 기대도수**를 구합니다.

In [ ]:
```
#적합성 검정
import pandas as pd
df=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/titanic.csv")
df_t=df[df['survived']==1]
```

```
table=df_t[['sex']].value_counts()
total_num=table.sum()
from scipy.stats import chisquare
chi= chisquare(table,f_exp=[total_num//2,total_num//2])
print(chi)
```

```
Power_divergenceResult(statistic=44.95906432748538, pvalue=2.0119672574477235e-11)
```

In [42]:
```
#독립성검정
table = pd.crosstab(df['class'],df['survived'])
table
```

Out[42]:

survived	0	1
class		
First	80	136
Second	97	87
Third	372	119

In [43]:
```
from scipy.stats import chi2_contingency
chi,p,df,expect = chi2_contingency(table)
print('statistics',chi)
print('p-value',p)
print('dof',df)
print('expect',expect)
```

```
statistics 102.88898875696056
p-value 4.549251711298793e-23
dof 2
expect [[133.09090909  82.90909091]
 [113.37373737  70.62626263]
 [302.53535354 188.46464646]]
```

In [ ]:
```
#카이제곱 대응되는 비모수 검정법
from scipy.stats import fisher_exact

table = [[1, 9],
         [11, 3]]
oddsratio, p = fisher_exact(table)
print(f"p={p:.4f}")
```

1. 맥니마 검정(McNemar's test)이란? 쌍으로 묶인 이진 분류 결과의 차이를 비교하는 검정.

주로 같은 데이터(피험자)에 대해 두 개의 처리/분류기/조건을 적용했을 때, 두 방법의 **정확도(또는 성공률)**가 유의하게 다른지 검정하는 데 사용.

전제: 표본이 같은 쌍(대응), 이진형 결과(예: 성공/실패, 정답/오답).

2. 2×2 분할표 구조

예: 두 분류기 A, B의 결과를 같은 표본에 적용했을 때

B = 성공 B = 실패

A=성공 n11 n10 A=실패 n01 n00

n11: 두 방법 다 성공 n00: 두 방법 다 실패 n10: A만 성공, B는 실패 n01: A는 실패, B만 성공

👉 핵심은 불일치 셀 n10,n01

3. 귀무가설 / 대립가설

귀무가설 : 두 방법의 성공 확률은 같다 → 불일치 빈도는 대칭이다. :P(A=1,B=0)=P(A=0,B=1)

즉, n10=n01

대립가설 : 두 방법의 성공 확률이 다르다 → 불일치 빈도 비대칭.

4. 검정통계량

큰 표본에서는

χ2=(|n10−n01|−1)^2/(n10+n01) (연속성 보정 포함; df=1)

표본이 적을 땐 이항검정(binomial test) 기반으로 p-value를 계산. 크면 카이 제곱

In [ ]:
```
# “같은 100명에게 두 진단검사를 적용했더니, 검사 A만 양성 30명, 검사 B만 양성 10명, 나머지는 동일.
# 맥니마 검정으로 두 검사 정확도 차이가 있는지 유의수준 5%에서 검정하라.”

# 풀이

# 불일치: n10=30, n01=10

# H0: 두 검사의 정확도는 같다

# 검정통계량 ≈ (|30−10|−1)² / (30+10) = 19.0
```

```python
# p-value < 0.001 → H0 기각

# 결론: 두 검사 정확도 차이 있음.
# [예상 문제 2]

# "머신러닝 분류기 A, B를 동일한 200개 샘플에 적용했더니: A만 맞춘 15개, B만 맞춘 25개.
# 맥니마 검정으로 성능 차이가 있는지 검정하라. 파이썬 코드로 제시하라."

import numpy as np
from scipy.stats import binomtest

# 예시: n10=30, n01=10
n10, n01 = 30, 10
n = n10 + n01
diff = n10 - n01

# 이항검정: 귀무가설 하에서 성공확률 = 0.5
res = binomtest(k=min(n10, n01), n=n, p=0.5, alternative="two-sided")
print("Binomial exact test p-value:", res.pvalue)
import statsmodels.api as sm

table = np.array([[50, 30],   # A 성공/실패
                  [10, 10]])  # A 실패/성공
# 주의: sm.stats.mcnemar은 [[n11, n10],[n01, n00]] 형태
result = sm.stats.mcnemar(table, exact=False, correction=True)
print("Chi2 =", result.statistic, "p =", result.pvalue)
# exact=True (이항검정) vs exact=False (χ² 근사)
```

카이제곱 근사(χ^2 test) 사용
불일치 셀 합 $n10+n01$ 이 충분히 클 때(대략 ≥25 이상) 적합
이때 자유도(df) = 1 자동 적용
correction=True → 연속성 보정(Yates correction) 적용

In [ ]:

```python
import numpy as np
import pandas as pd
from scipy.stats import poisson, chisquare

# 예: 관측 데이터 (카운트형)
data = np.array([0, 1, 2, 3, 2, 1, 0, 4, 3, 1, 2, 0, 1, 2, 3, 1, 0, 2, 1, 2])

# ① λ 추정
lambda_hat = np.mean(data)
n = len(data)

# ② 각 카운트별 빈도 (관측)
obs_counts = pd.Series(data).value_counts().sort_index()
```

```python
# ③ 이론적 포아송 기대빈도
max_k = obs_counts.index.max()
k_vals = np.arange(0, max_k + 1)
expected_probs = poisson.pmf(k_vals, mu=lambda_hat)
expected_counts = n * expected_probs

# ④ 기대빈도 너무 작은 건(보통 5 미만) 마지막 그룹에 합치기
while any(expected_counts < 5):
    expected_counts[-2] += expected_counts[-1]
    obs_counts.loc[max_k-1] += obs_counts.loc[max_k]
    expected_counts = expected_counts[:-1]
    obs_counts = obs_counts[:-1]
    max_k -= 1

# ⑤ χ² 검정
chi2_stat, p_value = chisquare(f_obs=obs_counts, f_exp=expected_counts)

print(f"λ 추정치: {lambda_hat:.3f}")
print(f"Chi-square: {chi2_stat:.3f}, p-value: {p_value:.4f}")

if p_value > 0.05:
    print("✅ 귀무가설 채택 → 포아송 분포로부터 유의한 차이 없음 (포아송 분포를 따른다고 볼 수 있음)")
else:
    print("❌ 귀무가설 기각 → 포아송 분포와 다름 (포아송이 아님)")
```