1. **CAST OBJECTS TO A DATA TYPE**

SELECT customerNumber,

    COUNT(\*) AS number\_payments,

    MIN(CAST(amount AS INT)) AS min\_purchase,

    MAX(CAST(amount AS INT))  AS max\_purchase,

    AVG(CAST(amount AS INT)) AS avg\_purchase,

    SUM(CAST(amount AS INT)) AS total\_spent

FROM payments

pd.read\_sql('''

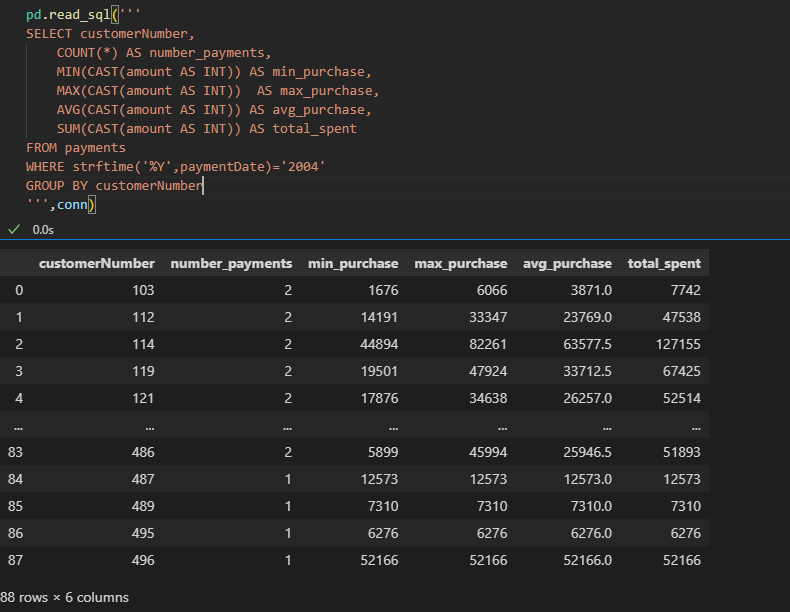
select cast(round(priceEach) as INTEGER) as rounded\_price\_int

        from orderDetails

            ''',conn)

**2.Strip year or month from date as a string object**

WHERE strftime('%Y',paymentDate)='2004'



pd.read\_sql('''

select orderDate,

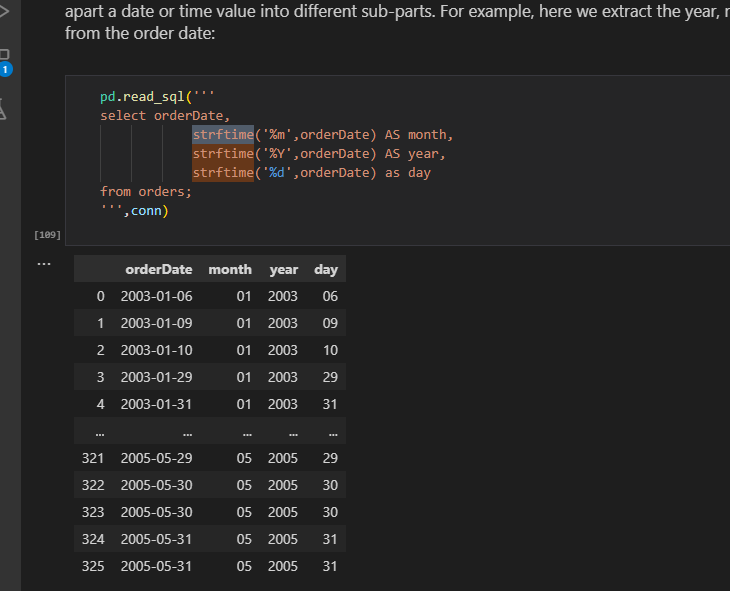
            strftime('%m',orderDate) AS month,

            strftime('%Y',orderDate) AS year,

            strftime('%d',orderDate) as day

from orders;

''',conn)



**Or use substr method**

pd.read\_sql('''

SELECT customerNumber,

    COUNT(\*) AS number\_payments,

    MIN(CAST(amount AS INT)) AS min\_purchase,

    MAX(CAST(amount AS INT))  AS max\_purchase,

    AVG(CAST(amount AS INT)) AS avg\_purchase,

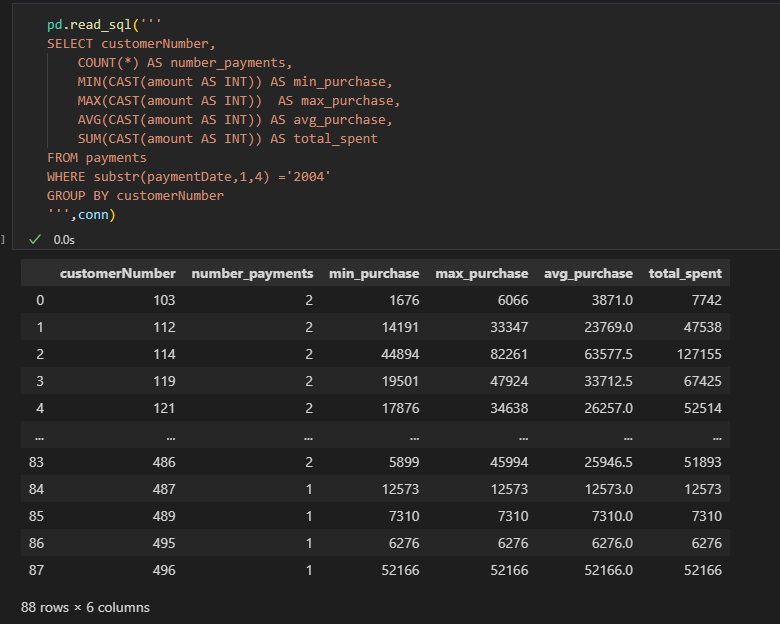
    SUM(CAST(amount AS INT)) AS total\_spent

FROM payments

WHERE substr(paymentDate,1,4) ='2004'

GROUP BY customerNumber

''',conn)



**3.Convert select statement to dataframe**

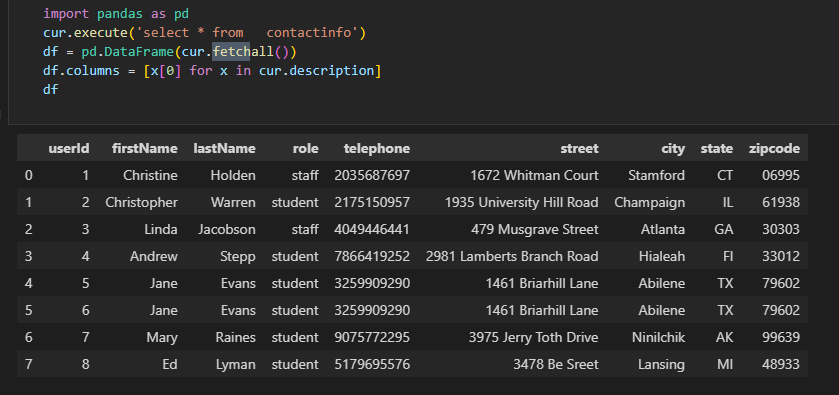
import pandas as pd

cur.execute('select \* from   contactinfo')

df = pd.DataFrame(cur.fetchall())

df.columns = [x[0] for x in cur.description]

df



* 4. Highest -**altitude**
* Southern/northern – **latitude**

**5.Pandasql Error**

**----> 6** passenger\_names **=** pysqldf**(**q**)**

**ImportError**: Unable to find a usable engine; tried using: 'sqlalchemy'.

A suitable version of sqlalchemy is required for sql I/O support.

Trying to import the above resulted in these errors:

**- Pandas requires version '1.4.0' or newer of 'sqlalchemy' (version '1.3.19' currently installed).**

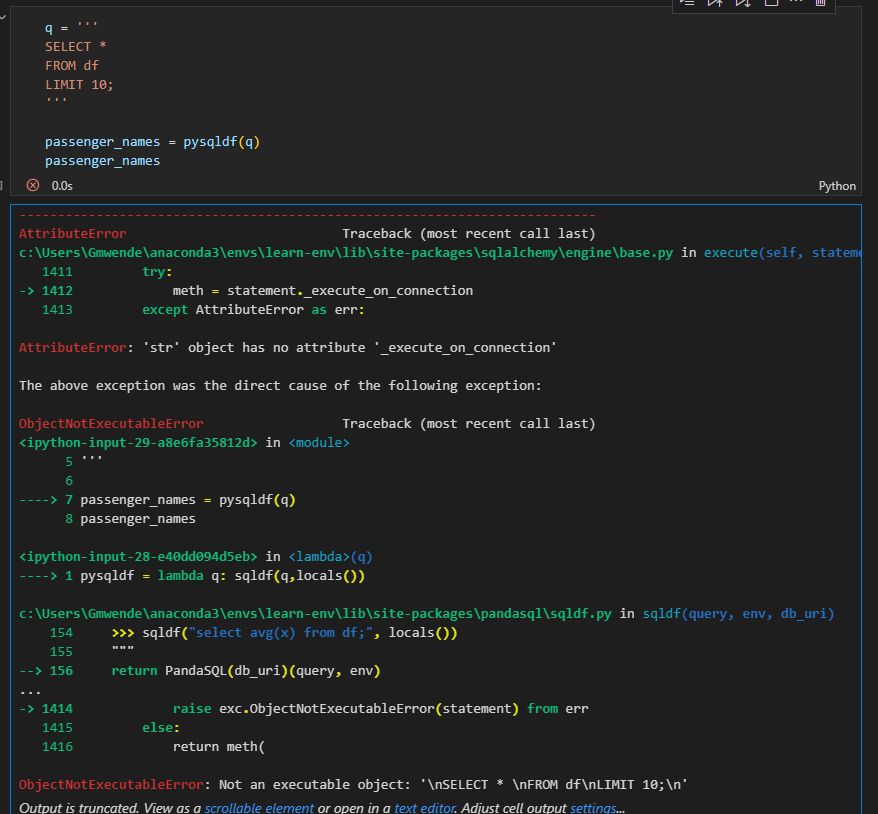
**TO update use**

conda update sqlalchemy

Check version if updated

pip show sqlalchemy

6.



Works well in colab

**7.Put dataframe in memory as to use conn**

#put df to memory

import sqlite3

conn = sqlite3.connect(':memory:')

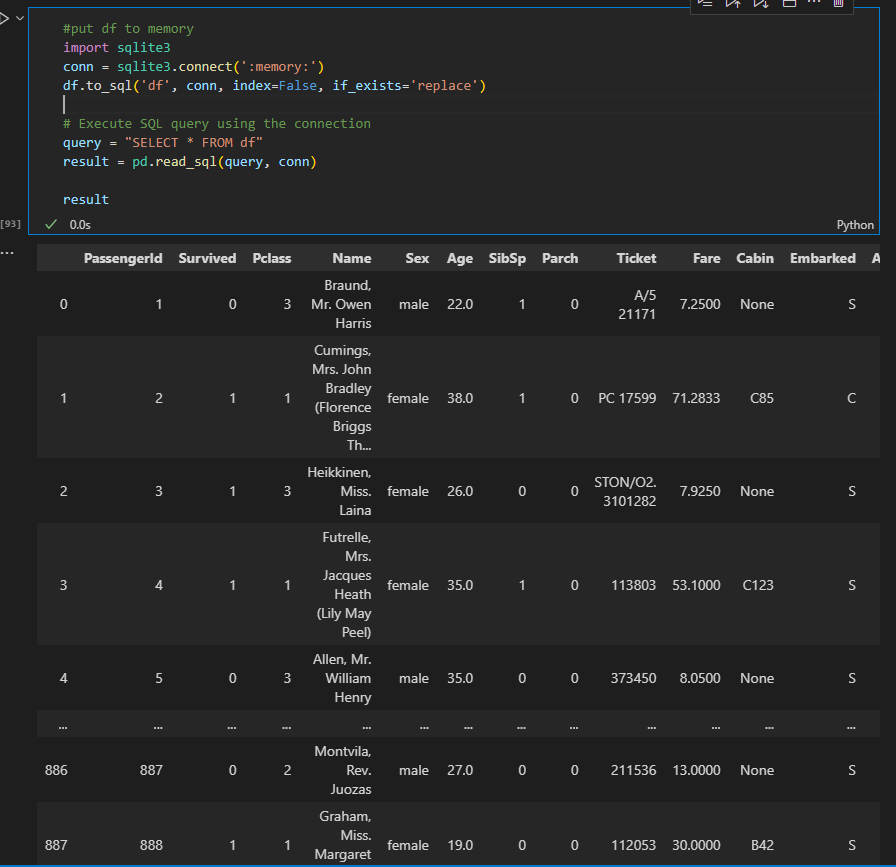
df.to\_sql('df', conn, index=False, if\_exists='replace')

# Execute SQL query using the connection

query = "SELECT \* FROM df"

result = pd.read\_sql(query, conn)

result



**8.Get female and children that is female and male less than or equal to 15**

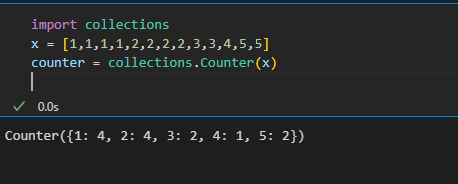
df[(df['Sex'] == 'female') | (df['Age'] <= 15)]

**9.Select everyonelse other than 1**

df[df['Pclass'] != '1']

**10.calculate totals using counter(get frequency for each value)**

counter = collections.Counter(x)



11. **Create two vertical subplots sharing 15% and 85% of plot space**

**Create density instead of count on seaborn histogram**

*#Create two vertical subplots sharing 15% and 85% of plot space*

*#sharex allows sharing of axes i.e building multiple plots on the same axes*

*fig, (ax,ax2) = plt.subplots(2,sharex=True,gridspec\_kw={'height\_ratios':(.15,.85)},figsize=(10,8))*

*sns.histplot(data['Height'],*

*lw=2,*

*edgecolor='r',*

*alpha=0.4,*

*color='w',*

*label='Histogram',*

*stat='density',*

*ax=ax2*

*)*

*sns.kdeplot(data.Height,*

*lw=3,*

*color='b',*

*label='Kernerl Density Estimation plot',*

*alpha=0.7,*

*ax=ax2*

*)*

*mean = data.Height.mean()*

*std = data.Height.std()*

*parametric\_dist = stats.norm(loc=mean, scale=std)*

*x=np.linspace(parametric\_dist.ppf(0.01),parametric\_dist.ppf(0.99),100)*

*ax2.plot(x,*

*parametric\_dist.pdf(x),*

*color='g',*

*alpha=0.7,*

*lw=3,*

*label = 'Parametric Fit'*

*)*

*ax2.set\_title('Density Estimations')*

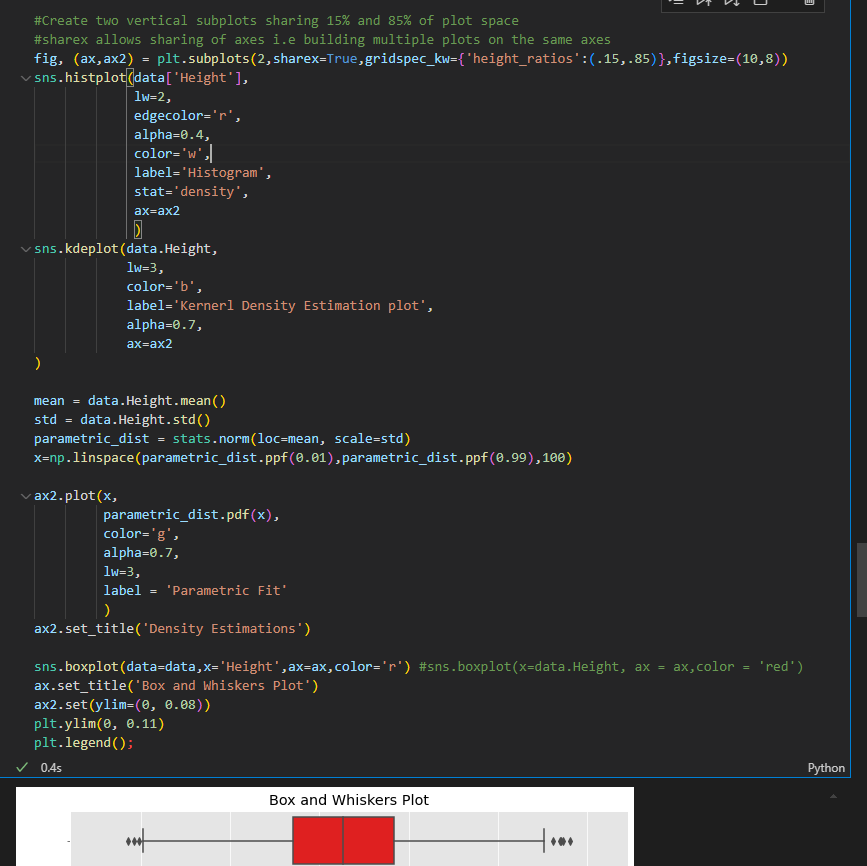
*sns.boxplot(data=data,x='Height',ax=ax,color='r') #sns.boxplot(x=data.Height, ax = ax,color = 'red')*

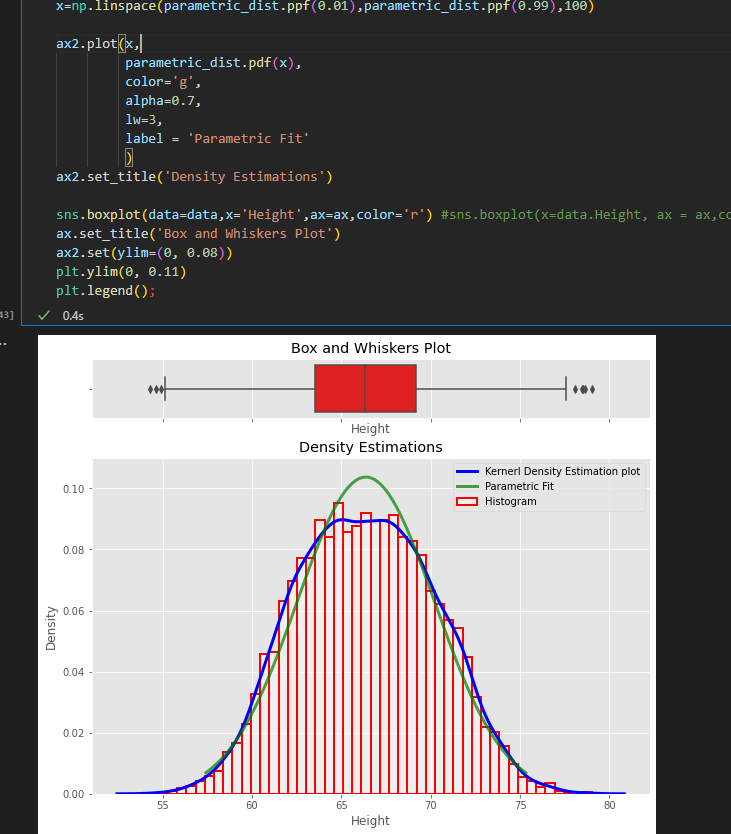
*ax.set\_title('Box and Whiskers Plot')*

*ax2.set(ylim=(0, 0.08))*

*plt.ylim(0, 0.11)*

*plt.legend();*





**12. Add density (probability) instead of counts in matplotlib histogram**

xtick\_locations = range(1,6)

bins = np.arange(6) +0.5 #[0.5, 1.5, 2.5, 3.5, 4.5, 5.5]

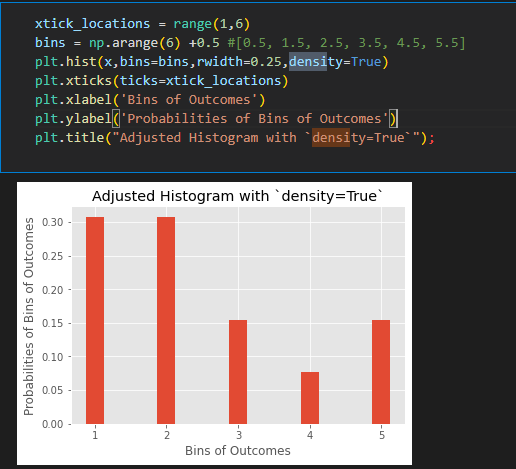
plt.hist(x,bins=bins,rwidth=0.25,density=True)

plt.xticks(ticks=xtick\_locations)

plt.xlabel('Bins of Outcomes')

plt.ylabel('Probabilities of Bins of Outcomes')

plt.title("Adjusted Histogram with `density=True`");



**13. ttest**

t= (x\_bar-mu)/(sample\_std/np.sqrt(25))

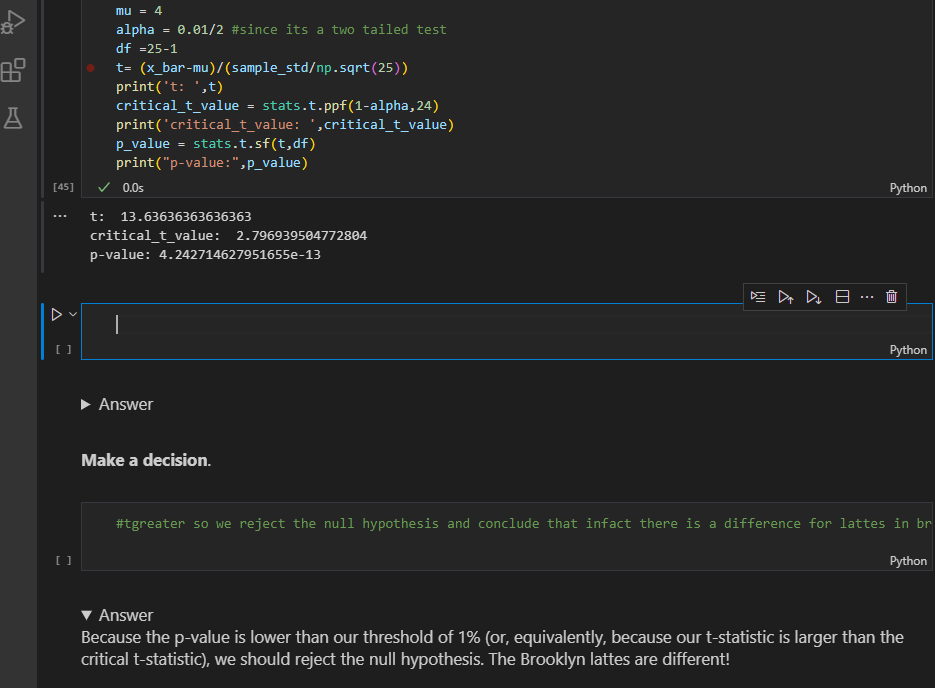
print('t: ',t)

critical\_t\_value = stats.t.ppf(1-alpha,24)

print('critical\_t\_value: ',critical\_t\_value)

p\_value = stats.t.sf(t,df)

print("p-value:",p\_value)



b)Example 2#one tailed left tail

sample =[20, 30, 30, 50, 75, 25, 30, 30, 40, 80]

x\_bar =  np.mean(sample)

sample\_std = np.std(sample,ddof=1)

n=len(sample)

df=n-1

mu =58

t\_stat1 = stats.ttest\_1samp(a=sample,popmean=58)

print('t\_stat1:', t\_stat1[0])

print('alpha:', t\_stat1[1]/2)

t\_stat2 = (x\_bar-mu)/(sample\_std/np.sqrt(n))

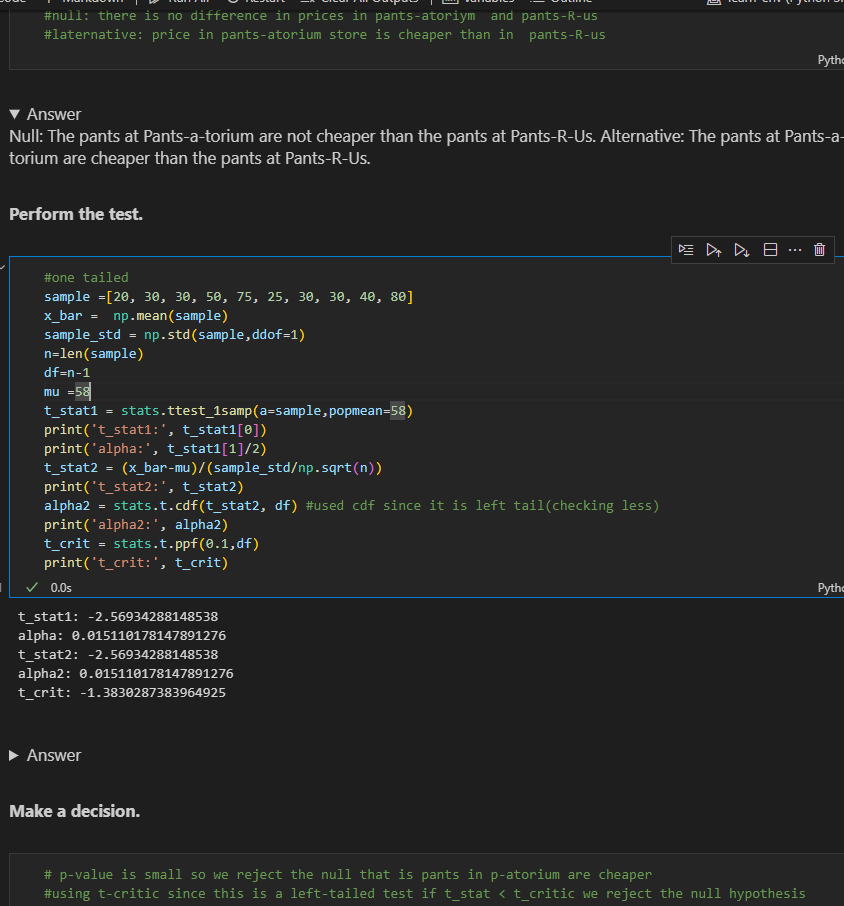
print('t\_stat2:', t\_stat2)

alpha2 = stats.t.cdf(t\_stat2, df) #used cdf since it is left tail(checking less)

print('alpha2:', alpha2)

t\_crit = stats.t.ppf(0.1,df)

print('t\_crit:', t\_crit)



**c)two-sample t-test**

**delivery\_times\_A = [28.4, 23.3, 30.4, 28.1, 29.4, 30.6, 27.8, 30.9, 27.0, 32.8]**

**mean\_A = np.mean(delivery\_times\_A)**

**std\_A = np.std(delivery\_times\_A)**

**nobs\_A = len(delivery\_times\_A)**

**mean\_B = 26.8**

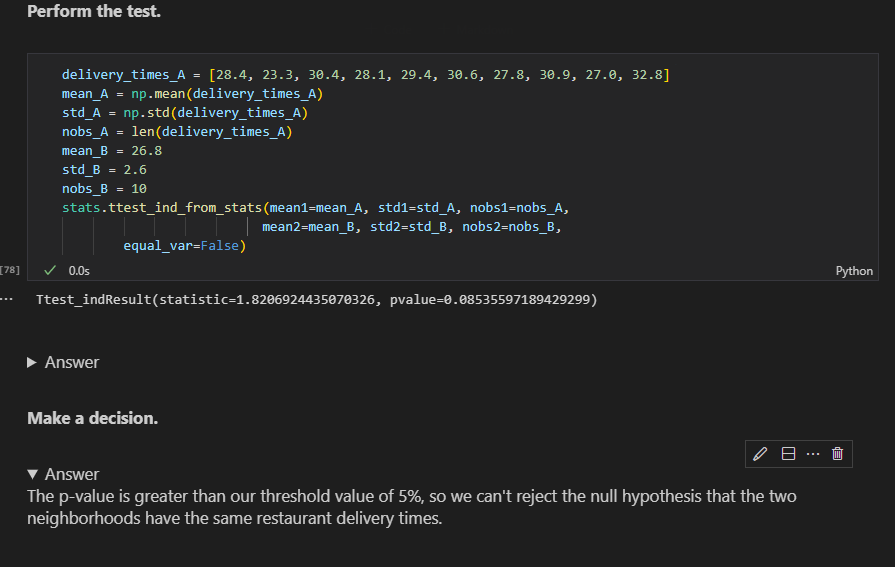
**std\_B = 2.6**

**nobs\_B = 10**

**stats.ttest\_ind\_from\_stats(mean1=mean\_A, std1=std\_A, nobs1=nobs\_A,**

**mean2=mean\_B, std2=std\_B, nobs2=nobs\_B,**

**equal\_var=False)**

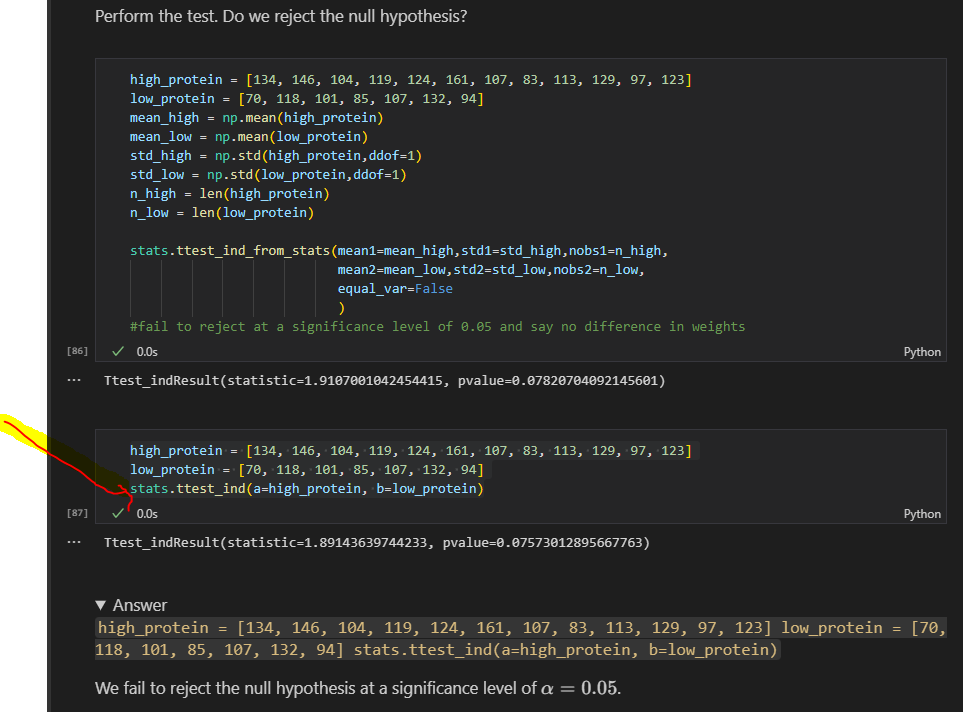


**c)2 sample again two tailed**

**high\_protein = [134, 146, 104, 119, 124, 161, 107, 83, 113, 129, 97, 123]**

**low\_protein = [70, 118, 101, 85, 107, 132, 94]**

**stats.ttest\_ind(a=high\_protein, b=low\_protein)**



**d)2 sample one tailed**

h\_bar = np.mean(high\_protein)

l\_bar = np.mean(low\_protein)

h\_df = len(high\_protein) - 1

l\_df = len(low\_protein) - 1

pooled\_var = (h\_df\*np.var(high\_protein) + l\_df\*np.var(low\_protein)) / (h\_df + l\_df)

t\_stat = (h\_bar - l\_bar) / np.sqrt(pooled\_var \* (1/len(high\_protein) + 1/len(low\_protein)))

t\_stat

stats.t(df=h\_df+l\_df).cdf(t\_stat)

