Table of contents 1.The Dataset 2.Sum,Average,Max,Min,Count,Columns,Row 3.IF -Replace IF with np.where -Replace nested if with np.select 4.SumIF,CountIF,AverageIF -One condition(Select a columns with square brackets[]) -Two or more conditions(select columns and use & or) 5.Basic Data Cleaning -Change the case of text with str.lower,str.upper,str.title -Extract text in a columns with str.extract -Identify whether a cell is empty with the .isnull method 6.Vlookup -Find an elemnent with .loc[-] -Merge two tables with pd.merge or pd.concat 7.Pivot table -Use the pivot table method 8.Replace excel graphs with pythons matplotlib or seaborn

In [14]:
```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

In [15]:
```
df_excel=pd.read_csv("studentsperformance.csv")
df_excel
```

Out[15]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [16]:
```
df_excel["math score"].mean()
```

Out[16]:
```
66.089
```

In [17]:
```
df_excel["math score"].max()
```

Out[17]:
```
100
```

In [18]:
```
df_excel["math score"].min()
```

Out[18]:
```
0
```

In [19]:
```
df_excel["math score"].count()
```

Out[19]:
```
1000
```

Now let's say we want to calculate the average of the 3 marks(math,reading and writing).These values are within different columns, so we have 2 options we can either sum each columns

In [20]:
```
##df_excel["average"]=(df_excel["math score"]+df_excel["reading score"]+df_excel["writing score"])/3
```

Or,use this method we used before to sum values within a columns,but in this case,we add axis =1

In [21]:
```
df_excel["average"]=df_excel.mean(axis=1)
```

Keep in mind that we used here the second option.Because all three columns(math_score,reading_score writing score)values are numerical.If there are additional columns have without numerical values then we must be used first option.

In [22]:
```
df_excel["gender"].value_counts()
```

Out[22]:
```
female    518
male      482
Name: gender, dtype: int64
```

IF

We can easily replace Excel's if function by using Numpy.

Replace IF with np.where

Let's imagine we want to know whether a student has passed or failed an exam and create a new columns with that information.We can easily do that with the following code.

In [23]:
```
df_excel["Pass/Fail"]=np.where(df_excel["average"]>70,"Pass","Fail")
```

As you can see np.where needs 3 arguments -the condition,the value if the condition is True and The value if the condition is False.

Replace nested IF with np.select

Let's imagine we want to give grades from A to F based on the scores obtained.In this case we have more than 2 values,so we use np.select() needs to arguments -a list of conditions and a list of values.A list in python is represented by the square brackets[]

In [24]:
```
conditions = [
    (df_excel["average"]>=90),
    (df_excel["average"]>=80) & (df_excel["average"]<90),
    (df_excel["average"]>=70) & (df_excel["average"]<80),
    (df_excel["average"]>=60) & (df_excel["average"]<70),
    (df_excel["average"]>=50) & (df_excel["average"]<60),
    (df_excel["average"]>=40) & (df_excel["average"]<50)
]
values=["A","B","C","D","E","F"]
```

Keep in mind that each conditions should be within parenthesis.Now we use the .select() method and assign it to a new["grades"] columns.

In [25]:
```
df_excel["grades"]=np.select(conditions,values)
```

SumIF,CountIF,AverageIF

To sum,count or calculate the average based on a condition,in python, we first filter out values and then make the calcultaion.

One condition (select a column with square brackets[ ])

Let's imagine we want to sum the scores for only the female gender.To do so,first we write the condition df_excel["gender"]=="female" and then we select the condition inside the df_excel frame by using square brckets[].

In [26]:
```
df_excel=df_excel[df_excel["gender"]=="female"]
```

Two or more conditions(select columns and use & or I)

If we have two or more conditions,the code will look similar to the one above but with some changes.Let's imagine we want to calculate the score of features within group B.(race/ethnicity)

In [27]:
```
df_sumifs=df_excel[(df_excel["gender"]=="female") & (df_excel["race/ethnicity"]=="group B")]
```

Since there are 2 conditions we could use &I that represents and/or respectivity.Keep in mind that each condition should be within parenthesis. Now let's sum the scores.

In [29]:
```
df_sumifs=df_sumifs.assign(sumifs=df_sumifs["math score"]+df_sumifs["reading score"]+df_sumifs["writing score"])
```

In this case,I used the .assign() method to show you another way to create a new column while doing calculations.

Basic data cleaning

We are going to check a few methods used for data cleaning. Change the case of text with .str.lower,.str.upper,.str.title . To access the strings contained in a column,we use .str then we can change the case of text with the followings.

In [33]:
```
df_excel["gender"].str.title()
df_excel["gender"].str.upper()
df_excel["gender"].str.title()
```

Out[33]:
```
0      Female
1      Female
2      Female
5      Female
6      Female
        ...
993    Female
995    Female
997    Female
998    Female
999    Female
Name: gender, Length: 518, dtype: object
```

To save the values we can overwrite a column like in the code below.

In [34]:
```
df_excel["gender"]=df_excel["gender"].str.title()
```

Extract text in a column with .str.extract

We can easily extract text from a column with .str.extract . In addition to that,if we want to extract specific patters of a text we can use regular expressions. Let's imagine we want to extract only the words in upper case within the column race/ethnicity(e.g. "B" from "group B"). To do so,we write the following code.

In [36]:
```
df_excel["race/ethnicity"].str.extract(r"([A-Z])")
```

Out[36]:

	0
0	B
1	C
2	B
5	B
6	B
...	...
993	D
995	E
997	C
998	D
999	D

518 rows × 1 columns

In this case,we used the regular expression r"([A-Z])" where [A-Z] indicate words in upper case,while the parethesis() is necessary to pick the desired pattern.Regular expression might look intimidating,but they're simpler than you think.In the link below,you'll find a simple guide to easily learn regular expression.

Pivot table

Let's imagine we want to obtain the math and writting score of all the groups inside the race/ethnicity column.

In [38]:
```
df_excel=pd.read_csv("studentsperformance.csv")
df_pivot=df_excel.pivot_table(index="race/ethnicity",values=["math score","writing score"],aggfunc="mean")
df_pivot
```

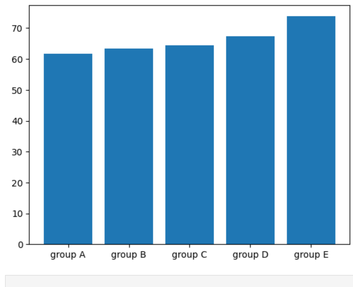
Out[38]:

race/ethnicity	math score	writing score
group A	61.629213	62.674157
group B	63.452632	65.600000
group C	64.463950	67.827586
group D	67.362595	70.145038
group E	73.821429	71.407143

Replace excel graphs with Python's Matplotlib or seaborn

Python contains different libraries to make visualizations as good as those excel offers. Let's make a simple barplot based on the results of df_pivot we created above.First we import matplotlib and the we use plt.bar().

In [39]:
```
import matplotlib.pyplot as plt
df_plot=df_pivot.reset_index()
plt.bar(df_plot["race/ethnicity"],df_plot["math score"])
plt.show()
```



In [ ]:
```
```