Link for web page:
https://en.wikipedia.org/wiki/List_of_FIFA_World_Cup_winning_players#cite_note-3

Web Page:

Purpose of this project:

- Web scraping the data
- Cleaning the data
- Data Exploration
- Data Analysis
- Data Visualization

WEB SCRAPING

01. Importing the libraries

```
In [1]: import numpy as np      # For mathematical operations
        import pandas as pd     # For data manipulation
        from bs4 import BeautifulSoup  # For webscraping
        import requests  # For making request from webserver
        import re        # For data cleaning
```

02. Getting the URL of the web page and making request

```
In [2]: # Getting 1 page of the website
        url="https://en.wikipedia.org/wiki/List_of_FIFA_World_Cup_winning_players#cite_note-3"

        # Making requests using get method
        req=requests.get(url)
        req

        # The status code is 200, thereby we can proceed
```

```
Out[2]: <Response [200]>
```


Hanan Fouzer

03. Web scraping the data using beautiful soup

```
In [3]: data=BeautifulSoup(req.content)
        data
```

```
Out[3]: <!DOCTYPE html>
        <html class="client-nojs vector-feature-language-in-header-enablec
        r-feature-sticky-header-disabled vector-feature-page-tools-pinned-
        ure-main-menu-pinned-disabled vector-feature-limited-width-clientp
        eature-custom-font-size-clientpref-1 vector-feature-appearance-pir
        me-clientpref-day vector-toc-available" dir="ltr" lang="en">
        <head>
        <meta charset="utf-8"/>
        <title>List of FIFA World Cup winning players - Wikipedia</title>
        <script>(function(){var className="client-js vector-feature-langua
        eader-disabled vector-feature-sticky-header-disabled vector-featur
```

04. Identifying the useful data out of all data

```
]: # Since relevant data is in between <th> tags
   # Using findAll function
   for i in data.findAll('th'):
       print(i)
```

```
<th rowspan="2" scope="col">Player
</th>
<th rowspan="2" scope="col">Team
</th>
<th colspan="2" scope="col">Titles won
</th>
<th rowspan="2" scope="col">Other appearances
</th>
```

05. Converting the above data into string and putting in a list

```
In [6]: arr=[]
        for i in data.findAll('th'):
            arr.append(str(i))

        arr
```

```
Out[6]: ['<th rowspan="2" scope="col">Player\n</th>',
         '<th rowspan="2" scope="col">Team\n</th>',
         '<th colspan="2" scope="col">Titles won\n</th>',
         '<th rowspan="2" scope="col">Other appearances\n</th>',
         '<th class="unsortable" rowspan="2" scope="col">Profile\n</th>',
         '<th rowspan="2">Birth\n</th>',
         '<th rowspan="2">Death\n</th>',
         '<th scope="col">Number\n</th>',
```

06. Identifying where data starts from the list

```
In [7]: # Using indexing:

        arr[9]
```

```
Out[7]: '<th data-sort-value="Pele" scope="row"><a href="/wiki/Pel%C3%A9" title="Pelé">Pelé</a>\n</th>'
```

07. Using regex module sub function

```
In [14]: re.sub('<th.*">|<th.*title="|">.*\n</th>|</a>.*\n</th>',"",arr[12])
```

```
Out[14]: '1962'
```

From arr, which searched data for <th> we were able to extract the below data

- Name of Player
- Year won

08. Using findAll and identify where the remaining data is:

```
In [16]: # Since the data which has td gives us the other useful data
         # We will create another list
         crr=[]
         for i in data.findAll("td"):
             crr.append(str(i))

         crr
```

```
Out[16]: ['<td align="left"><span style="white-space:nowrap"><span class="flag
         an><img alt="" class="mw-file-element" data-file-height="504" data-fi
         d.wikimedia.org/wikipedia/commons/thumb/2/2e/Flag_of_Brazil_%281968%E
         992%29.svg.png" srcset="//upload.wikimedia.org/wikipedia/commons/thum
         lag_of_Brazil_%281968%E2%80%931992%29.svg.png 1.5x, //upload.wikimedi
         8%E2%80%931992%29.svg/43px-Flag_of_Brazil_%281968%E2%80%931992%29.svg
```

From crr we will be able to get the data

- titles won
- team name
- year lost
- birth data
- death year

09. Identify where data starts from the list

```
In [18]: # Using indexing
         # We can note that crr[1] gives us the titles won data
         crr[1]
```

```
Out[18]: '<td><b>3</b>\n</td>'
```

10. Using regex module sub function

```
In [24]: # To get awards won, year lost, birth year,death year,team

         re.sub('<td.*team">|<td.*<b>|</b.*\n</td>|<td.*Cup">|</.*\n</td>|<t.*>|\n</td>',"",crr[6])
```

```
Out[24]: 'Brazil'
```

11. Consider final regex patterns

```
In [25]: # Final pattern for arr
         re.sub('<th.*">|<th.*title="|">.*\n</th>|</a>.*\n</th>',"",arr[9])
```

```
In [26]: # Final pattern for crr
         re.sub('<td.*team">|<td.*<b>|</b.*\n</td>|<td.*Cup">|</.*\n</td>|<t.*>|\n</td>',"",crr[6])
```

Index of data starts for arr with:

- 9 is for name
- 10 for year won

Index of data starts for crr with:

- 1 is for titles won
- 2 is for year lost
- 3 is for empty
- 4 is for Birth year
- 5 is for death year
- 6 is for team

12. Creating lists for each data and inserting each cleaned into respective lists

In [27]:
```
# Using arr logic
# Forming the loop to add data seperately in columns for Name and year won
Name=[]
Year_Won=[]
count=0
for i in data.findAll('th'):
    i=re.sub('<th.*">|<th.*title="|">.*\n</th>|</a>.*\n</th>',"",str(i))
# Since data starts from the 9th index onwards
    if count==0:
        count+=1
    elif count==1:
        count+=1
    elif count==2:
        count+=1
    elif count==3:
        count+=1
    elif count==4:
        count+=1
    elif count==5:
        count+=1
    elif count==6:
        count+=1
    elif count==7:
        count+=1
    elif count==8:
        count+=1
    elif count==9:
        Name.append(i)
        count+=1
    else:
        count=9
        Year_Won.append(i)
```

In [28]:
```
# Using crr logic
# Forming the loop to add data seperately in columns for the data
Titles_Won=[]
Year_Lost=[]
Empty=[]
Birth_Year=[]
Death_year=[]
Team=[]
count=0
for i in data.findAll('td'):
    i=re.sub('<td.*team">|<td.*<b>|</b.*\n</td>|<td.*Cup">|</.*\n</td>|<t.*>|\n</td>',"",str(i))
# Since data starts from 1st index onwards
    if count==0:
        count+=1
    elif count==1:
        Titles_Won.append(i)
        count+=1
    elif count==2:
        Year_Lost.append(i)
        count+=1
    elif count==3:
        Empty.append(i)
        count+=1
    elif count==4:
        Birth_Year.append(i)
        count+=1
    elif count==5:
        Death_year.append(i)
        count+=1
    else:
        count=1
        Team.append(i)
```

Each list will have the respective cleaned data:

Example:

```
In [29]: Name
```

```
Out[29]: ['Pelé',
          'Bellini',
          'Cafu',
          'Castilho',
          'Didi',
          'Djalma Santos',
          'Giovanni Ferrari',
          'Garrincha',
          'Gilmar',
          'Guido Masetti',
          'Mauro',
          'Giuseppe Meazza',
          'Eraldo Monzeglio',
```

13. Checking the len of each list (mainly name)

```
In [36]: # Checking how many elements are present
         len(Name)
```

```
Out[36]: 480
```

14. Seeing where the data ends exactly

```
In [43]: # Name[-1] gives us the last data. But it isnt relevant
         # We can see the last name ends with Name[-10]
         Name[-10]

         # The actual data of name is 480-10=470
```

```
Out[43]: 'Dino Zoff'
```

15. Creating a data frame:

```
In [45]: # We will create the dataframe with the above range
         # Using slicing for each list we end the data at 470 elements
         # Since slicing we put 471
         df=pd.DataFrame({"Name":Name[:471],"Year_Won":Year_Won[:471],"Titles_Won":Titles_Won[:471],
                          "Year_Lost":Year_Lost[:471],"Birth_Year":Birth_Year[:471],
                          "Death_year":Death_year[:471],"Team":Team[:471]})
         df
```

Out[45]:

	Name	Year_Won	Titles_Won	Year_Lost	Birth_Year	Death_year	Team
0	Pelé	1970	3	1966	1940	2022	Brazil
1	Bellini	1962	2	1966	1930	2014	Brazil
2	Cafu	2002	2	2006	1970		Brazil

DATA EXPLORATION AND FURTHER CLEANING

16. Exploring data

```
In [46]: # Exploring the info of the df
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 471 entries, 0 to 470
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Name        471 non-null    object
 1   Year_Won    471 non-null    object
 2   Titles_Won  471 non-null    object
 3   Year_Lost   471 non-null    object
 4   Birth_Year  471 non-null    object
 5   Death_year  471 non-null    object
 6   Team        471 non-null    object
dtypes: object(7)
memory usage: 25.9+ KB
```

17. Identifying incorrect entries

```
In [53]: df["Team"].unique()
         # Here we can see in team 1930 is there
         # Also there is FR Germany and Germany separately
```

```
Out[53]: array(['Brazil', 'Italy', 'Argentina', 'Spain', 'Uruguay', 'France',
                'England', 'FR Germany', 'Germany', '1930'], dtype=object)
```

18. Identifying place of error:

In [56]:
```
# Lets find the player who has the team assigned as 1930
# Using contains
error=df[df["Team"].str.contains('1930')]
error
# Dino Zoff is an Italian player
```

Out[56]:

	Name	Year_Won	Titles_Won	Year_Lost	Birth_Year	Death_year	Team
470	Dino Zoff	1982	1	1978	1942		1930

19. Further cleaning:

In [55]:
```
# Replacing
df["Team"]=df["Team"].str.replace('FR Germany','Germany')
```

In [57]:
```
# Lets replace it with Italy
df["Team"]=df["Team"].str.replace('1930','Italy')
```

20. Checking for null values:

In [59]:
```
# Checking for null values
df.isnull().sum()
```

Out[59]:
```
Name          0
Year_Won      0
Titles_Won    0
Year_Lost     0
Birth_Year    0
Death_year    0
Team          0
dtype: int64
```

21. Changing relevant data types:

```
In [60]: df["Year_Won"]=pd.to_datetime(df["Year_Won"],format='%Y')
```

```
In [61]: df["Year_Lost"]=pd.to_datetime(df["Year_Lost"],format='%Y')
```

```
In [62]: df["Titles_Won"]=pd.to_numeric(df["Titles_Won"])
```

```
In [63]: df["Birth_Year"]=pd.to_datetime(df["Birth_Year"],format='%Y')
```

```
In [65]: df["Death_year"]=pd.to_datetime(df["Death_year"],format='%Y')
```

22. Checking for duplicate values

```
In [82]: # Using duplicated function
duplicates = df[df.duplicated()]
duplicates
```

Out[82]:

Name	Year_Won	Titles_Won	Year_Lost	Birth_Year	Death_year	Team

ANALYSIS

01. Find the player with highest titles_won

```
In [87]: # We can just use max function
Highest_titles=df[df["Titles_Won"]==max(df["Titles_Won"])]["Name"]
Highest_titles
```

```
Out[87]: 0    Pelé
Name: Name, dtype: object
```

02. Find the youngest title player

```
In [91]: # Using max to find the highest year which is the youngest
Youngest=df[df["Birth_Year"]==max(df["Birth_Year"])]["Name"]
Youngest
```

```
Out[91]: 26      Thiago Almada
161    Enzo Fernández
Name: Name, dtype: object
```

03. Find the oldest title player

```
In [99]: # Saying min to find the oldest year which is the oldest
         Oldest=df[df["Birth_Year"]==min(df["Birth_Year"])]["Name"]
         Oldest
```

```
Out[99]: 404    Héctor Scarone
         Name: Name, dtype: object
```

04. List out the top 21 players who have got 3 or 2 as the titles won

```
In [122]: # Here we display the name and titles won column based on descending order of titles won
          Top21=df[["Name","Titles_Won"]].sort_values(by="Titles_Won",ascending=False).head(21)
          Top21
```

Out[122]:

	Name	Titles_Won
0	Pelé	3
11	Giuseppe Meazza	2
1	Bellini	2

05. Find the count of title players who are alive now

```
In [173]: # We can note that the null values represent players who are alive
          # Using isnull

          Alive_Players=df[df["Death_year"].isnull()]["Name"]
          Alive_Players.count()
```

```
Out[173]: 305
```

06. Find the count of players who are born from 1980 and onwards

```
In [178]: # When converting to datetime, the month and date was automatically added
          Players_80_onw=df[df["Birth_Year"]>='1980-01-01']["Name"]
          Players_80_onw.count()
```

```
Out[178]: 98
```

07. Find the count of title players who were born before 1980s

```
In [179]: Players_80_bef=df[df["Birth_Year"]<='1980-01-01']["Name"]
          Players_80_bef.count()
```

```
Out[179]: 376
```

08. Find the count of players who were born before 1980s and are still alive

In [194]:
```
# Using merge we can similarity
Born_bef1980_Alive=pd.merge(Players_80_bef,Alive_Players)
Born_bef1980_Alive.count()
# Out of 376, 211 players are still alive
```

Out[194]:
```
Name     211
dtype: int64
```

DATA VISUALIZATION

In [127]:
```
import matplotlib.pyplot as plt # For visualization
import seaborn as sns # For visualization
```

01. Create a new data frame with only the columns of name and titles won

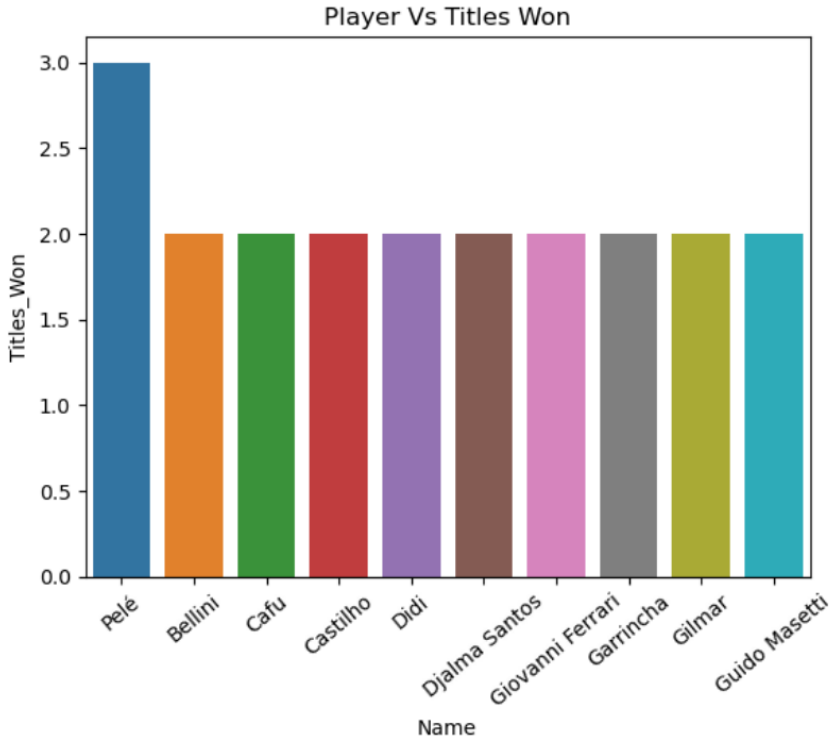
In [126]:
```
df2=df[["Name","Titles_Won"]]
df2
```

Out[126]:

	Name	Titles_Won
0	Pelé	3
1	Bellini	2
2	Cafu	2
3	Castilho	2

02. Create a plot with x being first 10 players and y being first 10 players titles_won

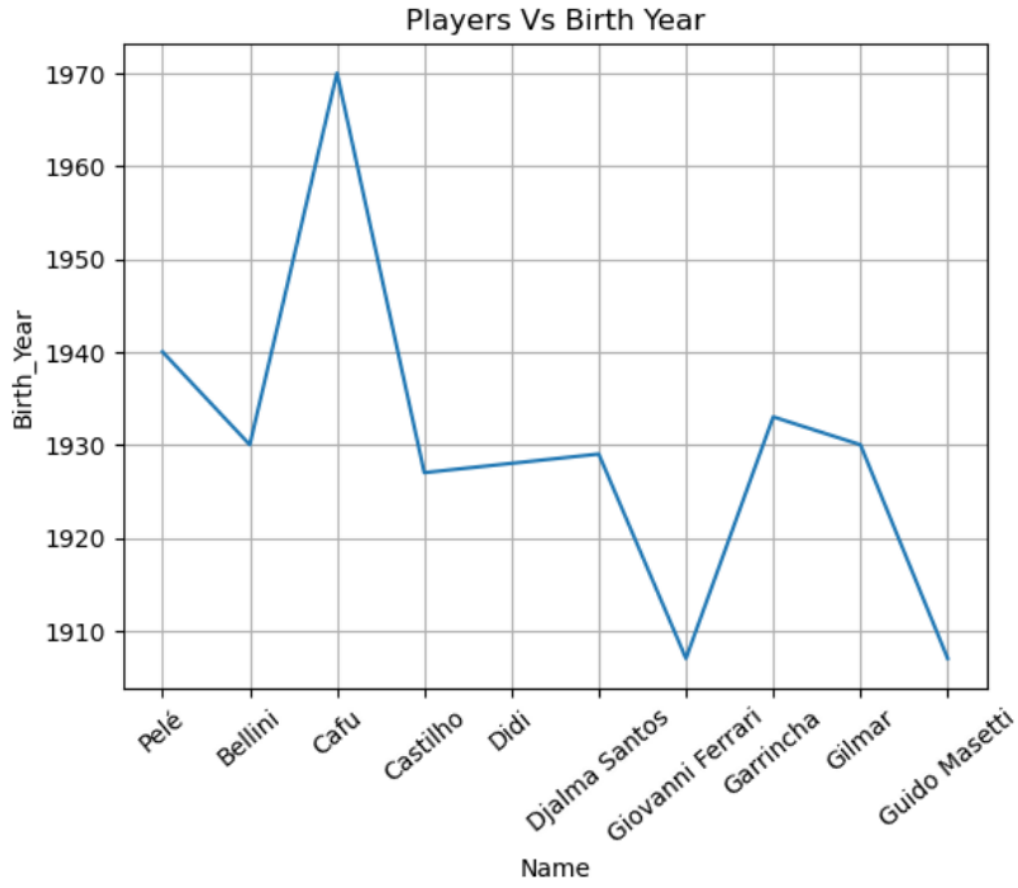
In [147]:
```
# Using sns, we create a bar plot
# By sayinh head(10), we get the first 10 data
plot1=sns.barplot(x=df2["Name"].head(10),y=df2["Titles_Won"].head(10),data=df2)
plt.title("Player Vs Titles Won")
# Since x ticks are put together without space, we rotating a bit for clear view
plot1.set_xticklabels(plot1.get_xticklabels(),rotation=40);
```



03. Create a plot with x being top 10 players and y being their birth year

In [148]:
```
# We are taking data from df
# Using a lineplot as this includes datetime data
plot2=sns.lineplot(x=df["Name"].head(10),y=df["Birth_Year"].head(10),data=df)

# For better view
plt.title("Players Vs Birth Year")
plt.grid()
plot2.set_xticklabels(plot2.get_xticklabels(),rotation=40);
```



Conclusions that can be made:

- We can see the titles winners are with birth years from 1890s until 2000s
- Out of the 471 title winners, 305 are still alive
- From birth year=1980s onwards, there are only 98 title winners
- Majority of the title winners are born before 1980s
- 211 players who were born before 1980s are still alive out of 376 players