# Pandas

March 18, 2020

## 1 Introduction

Data processing library for python.

```
pip install pandas
```

[2]:
```
import pandas as pd #importing pandas
```

## 2 Series and Dataframe

### 2.1 Dataframe

A dynamic 2D array (matrix / tables)

[3]:
```
ds = pd.read_csv('pandas_ds/data/olympics.csv') #rading the CSV file into 'ds'␣
↪object
```

[4]:
```
ds.head()
```

[4]:
```
  List of medallists at the Games of the Olympiad per edition, sport,
discipline, gender and event  \
0                                                  NaN
1  DISCLAIMER: The IOC Research and Reference Ser...
2                                                  NaN
3                                                 City
4                                               Athens

  Unnamed: 1 Unnamed: 2  Unnamed: 3     Unnamed: 4 Unnamed: 5 Unnamed: 6  \
0        NaN        NaN         NaN            NaN        NaN        NaN
1        NaN        NaN         NaN            NaN        NaN        NaN
2        NaN        NaN         NaN            NaN        NaN        NaN
3    Edition      Sport  Discipline        Athlete        NOC     Gender
4       1896   Aquatics    Swimming  HAJOS, Alfred        HUN        Men

       Unnamed: 7    Unnamed: 8 Unnamed: 9
0             NaN           NaN        NaN
1             NaN           NaN        NaN
2             NaN           NaN        NaN
3           Event  Event_gender      Medal
4  100m freestyle             M       Gold
```

[5]:
```
ds = pd.read_csv('pandas_ds/data/olympics.csv', skiprows=4) #skip first 4 rows
ds.head()
```

[5]:
```
      City  Edition     Sport Discipline             Athlete  NOC Gender  \
0  Athens     1896  Aquatics   Swimming       HAJOS, Alfred  HUN    Men
1  Athens     1896  Aquatics   Swimming    HERSCHMANN, Otto  AUT    Men
2  Athens     1896  Aquatics   Swimming   DRIVAS, Dimitrios  GRE    Men
3  Athens     1896  Aquatics   Swimming  MALOKINIS, Ioannis  GRE    Men
4  Athens     1896  Aquatics   Swimming  CHASAPIS, Spiridon  GRE    Men

                       Event Event_gender   Medal
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
3  100m freestyle for sailors            M    Gold
4  100m freestyle for sailors            M  Silver
```

## 2.2 Series

- 1D array of indexed data, dataframe is an ordered list of serieses.

- pandas supports alphanumeric indexing
- first column is called index
- Accessing a column(s)

    1. `df['col']`
    2. `df["col"]`
    3. `df.col` *doesn't work if col name has a space in it*
    4. `df[ ['col1','col2'] ]` *accessing multiple columns*

- to see object type use `type()` function

[6]:
```
# accessing a series (column)
print(ds['City'].head(5),'\n') # way1
print(ds["City"].head(5),'\n') # way2
print(ds.City.head(5),'\n')    # way3
print(ds[ ['City','Edition'] ].head(5))    # way3
```

```
0    Athens
1    Athens
2    Athens
3    Athens
4    Athens
Name: City, dtype: object

0    Athens
1    Athens
2    Athens
```

```
3    Athens
4    Athens
Name: City, dtype: object

0    Athens
1    Athens
2    Athens
3    Athens
4    Athens
Name: City, dtype: object

     City  Edition
0  Athens     1896
1  Athens     1896
2  Athens     1896
3  Athens     1896
4  Athens     1896
```

### 2.2.1 Confirming data type of pandas object

[7]:
```
type(ds.City)
```

[7]:
```
pandas.core.series.Series
```

[8]:
```
type(ds[ ['City', 'Edition'] ])
```

[8]:
```
pandas.core.frame.DataFrame
```

[9]:
```
type(ds)
```

[9]:
```
pandas.core.frame.DataFrame
```

## 2.3 Problem Excersise

1. list olympic dataframe
2. list only noc coloumn using both '.' and '[]' notation, what type is the object ?
3. list edition, city , athlete name and madel column, what type is this object ?

[10]:
```
# task 1
print('---------------task 1--------------------\n')
print(ds.head(5))

#task 2
print('---------------task 2--------------------\n')
print(ds.NOC.head(5),'\n')
print(ds['NOC'].head(5),'\n')
print(type(ds.NOC))

#task3
print('---------------task 3--------------------\n')
```

```
print(ds[ ['Edition', 'City', 'Athlete' , 'Medal']].head(5),'\n' )
print(type(ds[ ['Edition', 'City', 'Athlete' , 'Medal']].head(5)))
```

```
---------------task 1--------------------

     City  Edition     Sport Discipline             Athlete  NOC Gender  \
0  Athens     1896  Aquatics   Swimming       HAJOS, Alfred  HUN    Men
1  Athens     1896  Aquatics   Swimming    HERSCHMANN, Otto  AUT    Men
2  Athens     1896  Aquatics   Swimming   DRIVAS, Dimitrios  GRE    Men
3  Athens     1896  Aquatics   Swimming  MALOKINIS, Ioannis  GRE    Men
4  Athens     1896  Aquatics   Swimming  CHASAPIS, Spiridon  GRE    Men

                      Event Event_gender   Medal
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
3  100m freestyle for sailors            M    Gold
4  100m freestyle for sailors            M  Silver
---------------task 2--------------------

0    HUN
1    AUT
2    GRE
3    GRE
4    GRE
Name: NOC, dtype: object

0    HUN
1    AUT
2    GRE
3    GRE
4    GRE
Name: NOC, dtype: object

<class 'pandas.core.series.Series'>
---------------task 3--------------------

   Edition    City             Athlete   Medal
0     1896  Athens       HAJOS, Alfred    Gold
1     1896  Athens    HERSCHMANN, Otto  Silver
2     1896  Athens   DRIVAS, Dimitrios  Bronze
3     1896  Athens  MALOKINIS, Ioannis    Gold
4     1896  Athens  CHASAPIS, Spiridon  Silver

<class 'pandas.core.frame.DataFrame'>
```

# 3 Data Input and Validation

## 3.1 `read_csv()` function

reads a CSV file into a dataframe

[11]:
```
list(range(0,4))
```

[11]:
```
[0, 1, 2, 3]
```

[12]:
```
pd.read_csv('pandas_ds/data/olympics.csv', skiprows=list(range(0,4))).head()
```

[12]:
```
     City  Edition     Sport Discipline             Athlete  NOC Gender  \
0  Athens     1896  Aquatics   Swimming       HAJOS, Alfred  HUN    Men
1  Athens     1896  Aquatics   Swimming    HERSCHMANN, Otto  AUT    Men
2  Athens     1896  Aquatics   Swimming   DRIVAS, Dimitrios  GRE    Men
3  Athens     1896  Aquatics   Swimming  MALOKINIS, Ioannis  GRE    Men
4  Athens     1896  Aquatics   Swimming  CHASAPIS, Spiridon  GRE    Men

                       Event Event_gender   Medal
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
3  100m freestyle for sailors            M    Gold
4  100m freestyle for sailors            M  Silver
```

## 3.2 `shape` attribute

[13]:
```
ds.shape    #returns the dimention in a touple of tha dataframe (row,col)
```

[13]:
```
(29216, 10)
```

[14]:
```
print(f'rows = {ds.shape[0]} , cols = {ds.shape[1]} ')
```

```
rows = 29216 , cols = 10
```

## 3.3 `head()` & `Teil()`

- by default first five (head) and last five (tail) rows
- number can be manipulated by putting parameters
- use negative parameter e.g. `ds.head(-n)` or `ds.tail(-n)` to print all rows but **last n or first n** respectively

[15]:
```
ds.head(4)
```

[15]:
```
     City  Edition     Sport Discipline             Athlete  NOC Gender  \
0  Athens     1896  Aquatics   Swimming       HAJOS, Alfred  HUN    Men
1  Athens     1896  Aquatics   Swimming    HERSCHMANN, Otto  AUT    Men
2  Athens     1896  Aquatics   Swimming   DRIVAS, Dimitrios  GRE    Men
3  Athens     1896  Aquatics   Swimming  MALOKINIS, Ioannis  GRE    Men

                       Event Event_gender   Medal
```

```
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
3  100m freestyle for sailors            M    Gold
```

[16]:
```
ds.tail(4)
```

[16]:
```
          City  Edition      Sport       Discipline               Athlete  \
29212  Beijing     2008  Wrestling  Wrestling Gre-R  MIZGAITIS, Mindaugas
29213  Beijing     2008  Wrestling  Wrestling Gre-R       PATRIKEEV, Yuri
29214  Beijing     2008  Wrestling  Wrestling Gre-R         LOPEZ, Mijain
29215  Beijing     2008  Wrestling  Wrestling Gre-R        BAROEV, Khasan

       NOC Gender       Event Event_gender   Medal
29212  LTU    Men  96 - 120kg            M  Bronze
29213  ARM    Men  96 - 120kg            M  Bronze
29214  CUB    Men  96 - 120kg            M    Gold
29215  RUS    Men  96 - 120kg            M  Silver
```

### 3.4 info

- printing summary
- useful to see which column has any missing data,
- compare number of non-nulls with total number of entries

[17]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29216 entries, 0 to 29215
Data columns (total 10 columns):
City            29216 non-null object
Edition         29216 non-null int64
Sport           29216 non-null object
Discipline      29216 non-null object
Athlete         29216 non-null object
NOC             29216 non-null object
Gender          29216 non-null object
Event           29216 non-null object
Event_gender    29216 non-null object
Medal           29216 non-null object
dtypes: int64(1), object(9)
memory usage: 2.2+ MB
```

## 4 Basic Data analysis

### 4.1 `value_counts()`

- takes a series and returns a series object, counting all the unique values

- The returned sereis is ordered by frequency
- flags
  - `ascending` : alter freq order (false by def)
  - `dropna` : doesn't count N/A (true by def)
  - `sort` : sort by freq (true by def)

[18]:
```
ds['City'].value_counts()
```

[18]:
```
Athens                  2149
Los Angeles             2074
Beijing                 2042
Sydney                  2015
Atlanta                 1859
Barcelona               1705
London                  1618
Seoul                   1546
Paris                   1396
Moscow                  1387
Montreal                1305
Antwerp                 1298
Munich                  1185
Mexico                  1031
Tokyo                   1010
Helsinki                 889
Melbourne / Stockholm    885
Stockholm                885
Rome                     882
Berlin                   875
Amsterdam                710
St Louis                 470
Name: City, dtype: int64
```

[19]:
```
# top 10 years when most madels were given
ds.Edition.value_counts().head(10)
```

[19]:
```
2008    2042
2000    2015
2004    1998
1996    1859
1992    1705
1988    1546
1984    1459
1980    1387
1976    1305
1920    1298
Name: Edition, dtype: int64
```

[20]:
```
# how many madels were given to men and women throuout the history
ds.Gender.value_counts()
```

[20]:
```
Men      21721
Women     7495
Name: Gender, dtype: int64
```

### 4.2 sort_values()

- sort type : quick (def), merge, heap

[21]:
```
# sort the series by the atheletes' name
ds['Athlete'].sort_values().head(5)
```

[21]:
```
651                 AABYE, Edgar
2849       AALTONEN, Arvo Ossian
2852       AALTONEN, Arvo Ossian
7716    AALTONEN, Paavo Johannes
7730    AALTONEN, Paavo Johannes
Name: Athlete, dtype: object
```

[22]:
```
# sort the edition first, for each value of edition sort by athlete name
ds[ ['Edition','Athlete'] ].sort_values(by=['Edition', 'Athlete']).head(10)
```

[22]:
```
     Edition                   Athlete
7       1896          ANDREOU, Joannis
82      1896  ANDRIAKOPOULOS, Nicolaos
110     1896  ANDRIAKOPOULOS, Nicolaos
111     1896    ATHANASOPOULOS, Spyros
48      1896             BATTEL, Edward
19      1896              BLAKE, Arthur
134     1896              BOLAND, John
140     1896              BOLAND, John
13      1896             BURKE, Thomas
21      1896             BURKE, Thomas
```

### 4.3 Booean indexing

- boolean vectors can be used for filtering data (operators : `&`, `|`, `~`)
- multiple conditioned must be grouped in paranthesis

[23]:
```
# list atheletes who have won a gold madel
ds[ ds['Medal'] == 'Gold' ]['Athlete'].head(5)
```

[23]:
```
0          HAJOS, Alfred
3     MALOKINIS, Ioannis
6          HAJOS, Alfred
9          NEUMANN, Paul
13         BURKE, Thomas
Name: Athlete, dtype: object
```

[24]:
```
# all woman athlete who won a gold madel
ds[ (ds['Medal'] == 'Gold') & (ds['Gender'] == 'Women') ].head(5)
```

[24]:
```
          City  Edition    Sport Discipline                Athlete  NOC Gender  \
417      Paris     1900     Golf       Golf  ABBOTT, Margaret Ives  USA  Women
641      Paris     1900   Tennis     Tennis      COOPER, Charlotte  GBR  Women
649      Paris     1900   Tennis     Tennis      COOPER, Charlotte  GBR  Women
710  St Louis      1904  Archery    Archery  HOWELL, Matilda Scott  USA  Women
713  St Louis      1904  Archery    Archery  HOWELL, Matilda Scott  USA  Women

                                      Event Event_gender Medal
417                                individual            W  Gold
641                             mixed doubles            X  Gold
649                                   singles            W  Gold
710  double columbia round (50y - 40y - 30y)            W  Gold
713         double national round (60y - 50y)            W  Gold
```

### 4.4 String handling

[25]:
```
#search for entries with name = 'Florence'
ds[ds.Athlete.str.contains('Florence')]
```

[25]:
```
                       City  Edition      Sport      Discipline  \
1843                 London     1908    Skating  Figure skating
1848                 London     1908    Skating  Figure skating
4173                  Paris     1924   Aquatics        Swimming
8162                Helsinki     1952  Athletics       Athletics
9060   Melbourne / Stockholm     1956  Athletics       Athletics
10849                 Tokyo     1964  Athletics       Athletics
16817           Los Angeles     1984  Athletics       Athletics
18287                 Seoul     1988  Athletics       Athletics
18305                 Seoul     1988  Athletics       Athletics
18347                 Seoul     1988  Athletics       Athletics
18374                 Seoul     1988  Athletics       Athletics

                              Athlete  NOC Gender                  Event  \
1843                 SYERS, Florence  GBR  Women             individual
1848                 SYERS, Florence  GBR  Women                  pairs
4173                BARKER, Florence  GBR  Women  4x100m freestyle relay
8162       FOULDS-PAUL, June Florence  GBR  Women           4x100m relay
9060       FOULDS-PAUL, June Florence  GBR  Women           4x100m relay
10849  AMOORE-POLLOCK, Judith Florence  AUS  Women                  400m
16817       GRIFFITH-JOYNER, Florence  USA  Women                  200m
18287       GRIFFITH-JOYNER, Florence  USA  Women                  100m
18305       GRIFFITH-JOYNER, Florence  USA  Women                  200m
18347       GRIFFITH-JOYNER, Florence  USA  Women           4x100m relay
18374       GRIFFITH-JOYNER, Florence  USA  Women           4x400m relay

     Event_gender   Medal
1843            W    Gold
```

```
1848            X  Bronze
4173            W  Silver
8162            W  Bronze
9060            W  Silver
10849           W  Bronze
16817           W  Silver
18287           W    Gold
18305           W    Gold
18347           W    Gold
18374           W  Silver
```

### 4.5 Problem Excersise

1. in which event did **jasse Owens** win a madel ?
2. which country has won the most men's gold medalin singles badminton over the years ?
   - sort the results alphabetically bythe players name
3. which three countries won the most madels in **1984 to 2008** period ?
4. display the male gold madel winner forthe 100m track and field sprint event over the years
   - list the results from the most recent first
   - show the olympic city, edition,athlete and country they represented

[26]:
```
ds.columns
```

[26]:
```
Index(['City', 'Edition', 'Sport', 'Discipline', 'Athlete', 'NOC', 'Gender',
       'Event', 'Event_gender', 'Medal'],
      dtype='object')
```

[27]:
```
# tasks 1 in which event did jasse Owens win a madel ?
ds[ds.Athlete.str.contains('OWENS, Jesse') & (ds.Medal == 'Gold')].Event.
 ↪value_counts()
```

[27]:
```
long jump       1
100m            1
4x100m relay    1
200m            1
Name: Event, dtype: int64
```

[28]:
```
#which country has won the most men's gold medal in singles badminton over the␣
 ↪years ?
ds[(ds.Sport == 'Badminton') &
   (ds.Event == 'singles') &
   (ds.Gender == 'Men') &
   (ds.Medal == 'Gold')].NOC.value_counts()
```

[28]:
```
CHN    2
INA    2
DEN    1
Name: NOC, dtype: int64
```

[29]:
```
#sort the results alphabetically by the players name

ds[(ds.Sport == 'Badminton') &
   (ds.Event == 'singles') &
   (ds.Gender == 'Men') &
   (ds.Medal == 'Gold')].sort_values(by = ['NOC','Athlete'])
```

[29]:
```
            City  Edition      Sport Discipline                 Athlete  NOC  \
23717     Sydney     2000  Badminton  Badminton              JI, Xinpeng  CHN
27741    Beijing     2008  Badminton  Badminton                 LIN, Dan  CHN
21787    Atlanta     1996  Badminton  Badminton  HOYER-LARSEN, Poul Erik  DEN
20045  Barcelona     1992  Badminton  Badminton        BUDI KUSUMA, Alan  INA
25734     Athens     2004  Badminton  Badminton          HIDAYAT, Taufik  INA

      Gender    Event Event_gender Medal
23717    Men  singles            M  Gold
27741    Men  singles            M  Gold
21787    Men  singles            M  Gold
20045    Men  singles            M  Gold
25734    Men  singles            M  Gold
```

[30]:
```
# which three countries won the most madels in 1984 to 2008 period ?

ds[(ds.Edition >= 1984) & (ds.Edition <=2008)].NOC.value_counts().head(3)
```

[30]:
```
USA    1837
AUS     762
GER     691
Name: NOC, dtype: int64
```

[31]:
```
#display the male gold madel winner forthe 100m track in field sprint event␣
 ↪over the years
#list the results from the most recent first
#show the olympic city, edition,athlete and country they represented

ds[(ds.Event == '100m') &
   (ds.Medal == 'Gold') &
   (ds.Gender == 'Men')][['City' , 'Edition', 'Athlete' , 'NOC']].
 ↪sort_values(by=['Edition'],

                                                                    ␣
 ↪ascending = False)
```

[31]:
```
                      City  Edition                Athlete  NOC
27552               Beijing     2008            BOLT, Usain  JAM
25539                Athens     2004          GATLIN, Justin  USA
23521                Sydney     2000         GREENE, Maurice  USA
21598               Atlanta     1996         BAILEY, Donovan  CAN
19859             Barcelona     1992       CHRISTIE, Linford  GBR
18284                 Seoul     1988             LEWIS, Carl  USA
```

```
16794            Los Angeles     1984            LEWIS, Carl  USA
15374                 Moscow     1980          WELLS, Allan  GBR
14069                Montreal     1976      CRAWFORD, Hasely  TRI
12902                 Munich     1972        BORZOV, Valery  URS
11865                 Mexico     1968      HINES, James Ray  USA
10823                  Tokyo     1964         HAYES, Robert  USA
9924                    Rome     1960           HARY, Armin  EUA
9009   Melbourne / Stockholm     1956  MORROW, Robert Joseph  USA
8121                Helsinki     1952       REMIGINO, Lindy  USA
7302                  London     1948     DILLARD, Harrison  USA
6427                  Berlin     1936          OWENS, Jesse  USA
5806             Los Angeles     1932          TOLAN, Eddie  USA
5095               Amsterdam     1928       WILLIAMS, Percy  CAN
4236                   Paris     1924      ABRAHAMS, Harold  GBR
2996                 Antwerp     1920      PADDOCK, Charles  USA
2022               Stockholm     1912          CRAIG, Ralph  USA
1191                  London     1908      WALKER, Reginald  RSA
737                St Louis     1904          HAHN, Archie  USA
231                    Paris     1900       JARVIS, Francis  USA
13                    Athens     1896         BURKE, Thomas  USA
```

### 4.6 Basic Plotting

[32]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

### 4.7 plot type

use `plt.plot(kind = <type> )` command to specify the plot type * `kind = 'line'` : line plot, traking changes over time * `kind = 'bar'` : bar chart, comparing between different groups * `kind = 'barh'` : horizontal bar graph * `kind = 'pie'` : pie chart, comparing share or propoerion

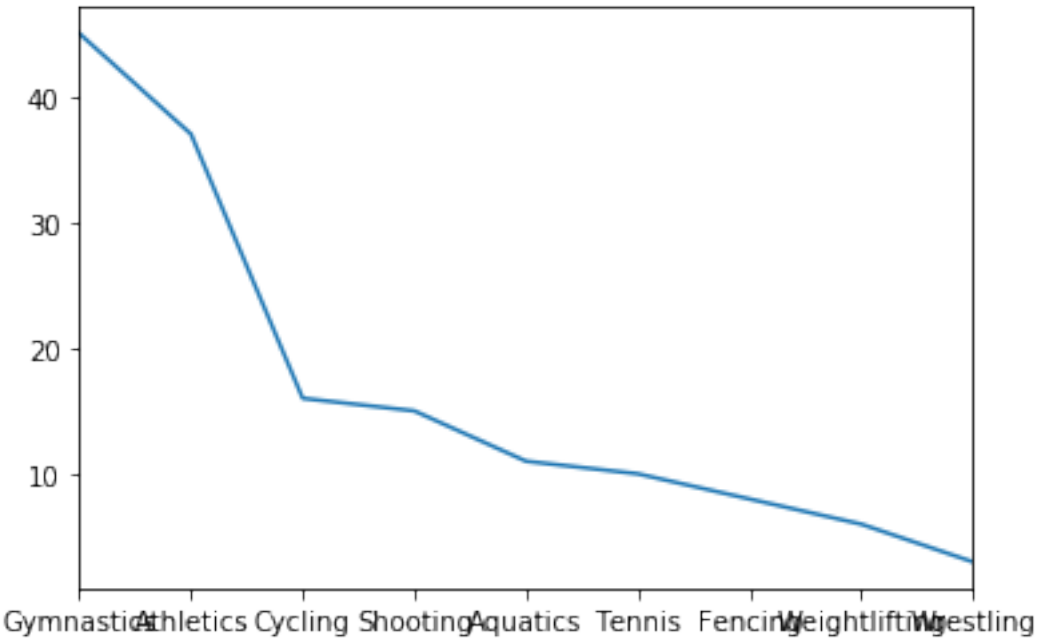
[33]:
```
# what were the sports in the forst olympics, plot them with different graphs

ds.Edition.min()    # the first olympics edition
ds[ ds.Edition == ds.Edition.min() ]  # data about first olympics
ds1 = ds[ ds.Edition == ds.Edition.min() ].Sport.value_counts() #stat of␣
 ↪various sports in fist olympics
ds1
```

[33]:
```
Gymnastics     45
Athletics      37
Cycling        16
Shooting       15
Aquatics       11
Tennis         10
Fencing         8
```

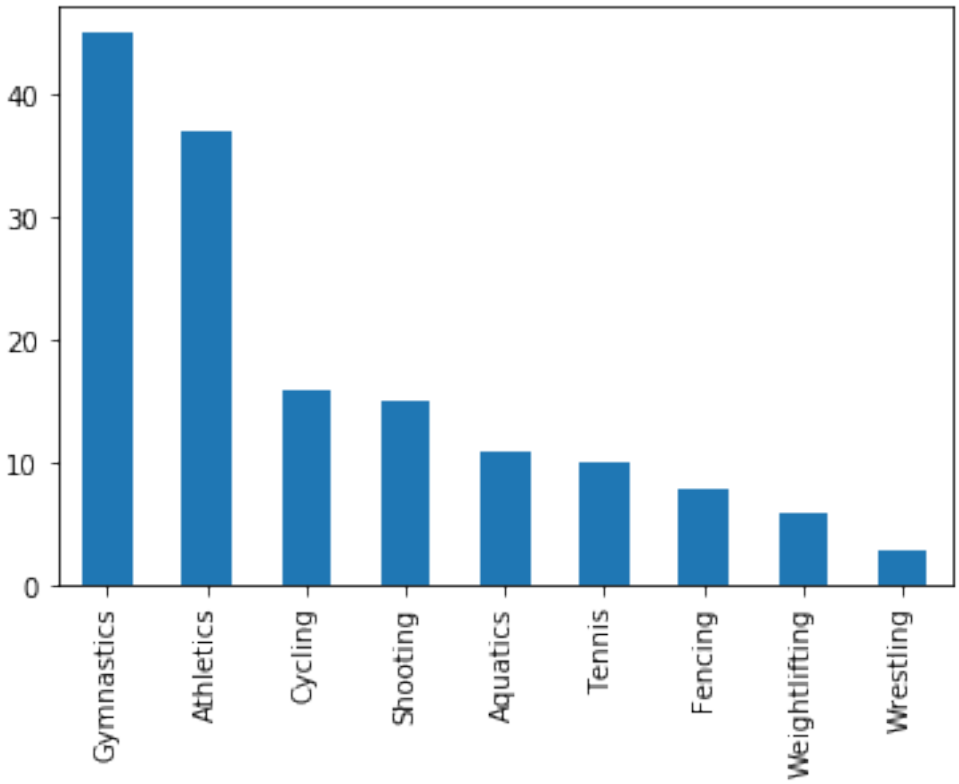
```
Weightlifting     6
Wrestling         3
Name: Sport, dtype: int64
```

[34]:
```
# line plot (default)
ds1.plot()
```

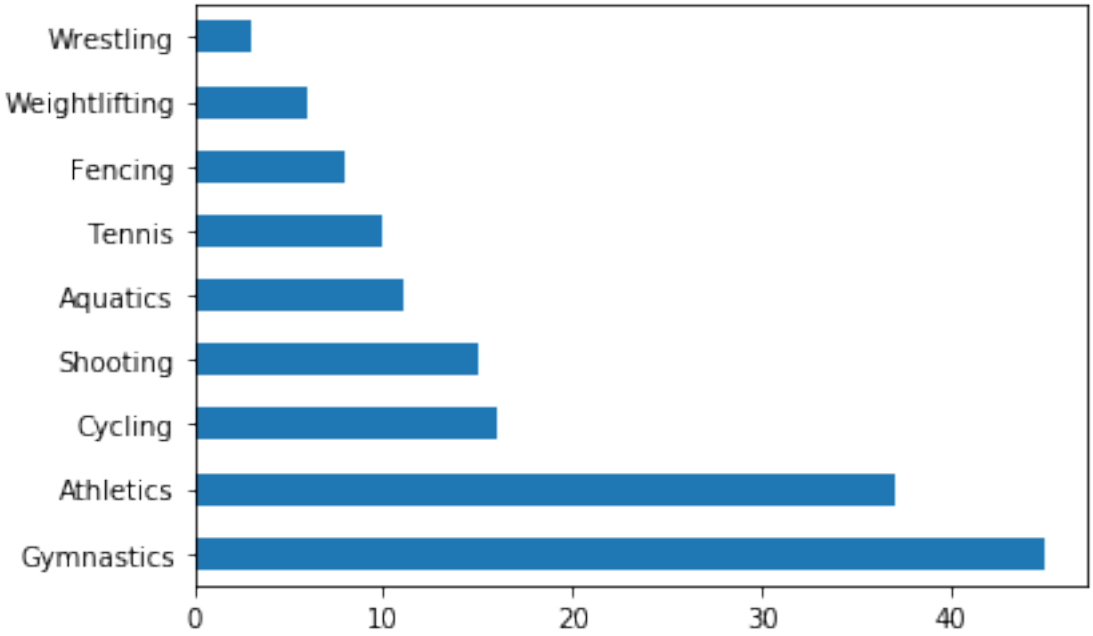
[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a91065c128>
```



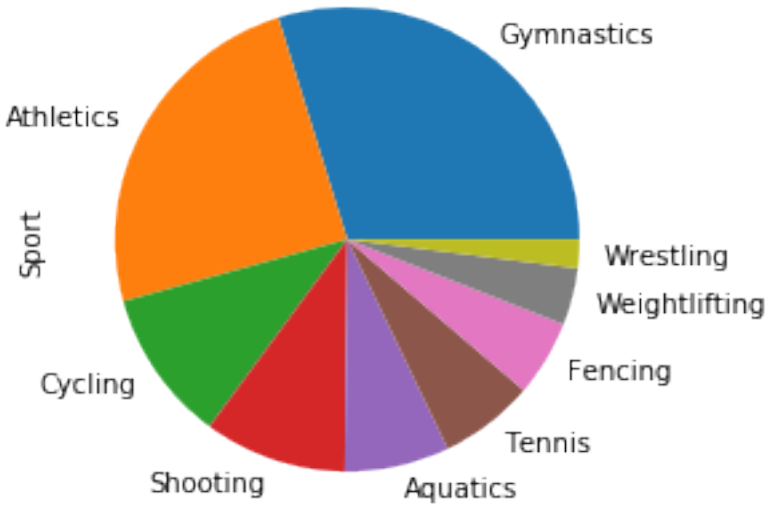

[35]:
```
# bar plot
ds1.plot(kind = 'bar'); # ';' to suppress log message
```




[36]:
```
# bar horizontal plot
ds1.plot(kind = 'barh');
```



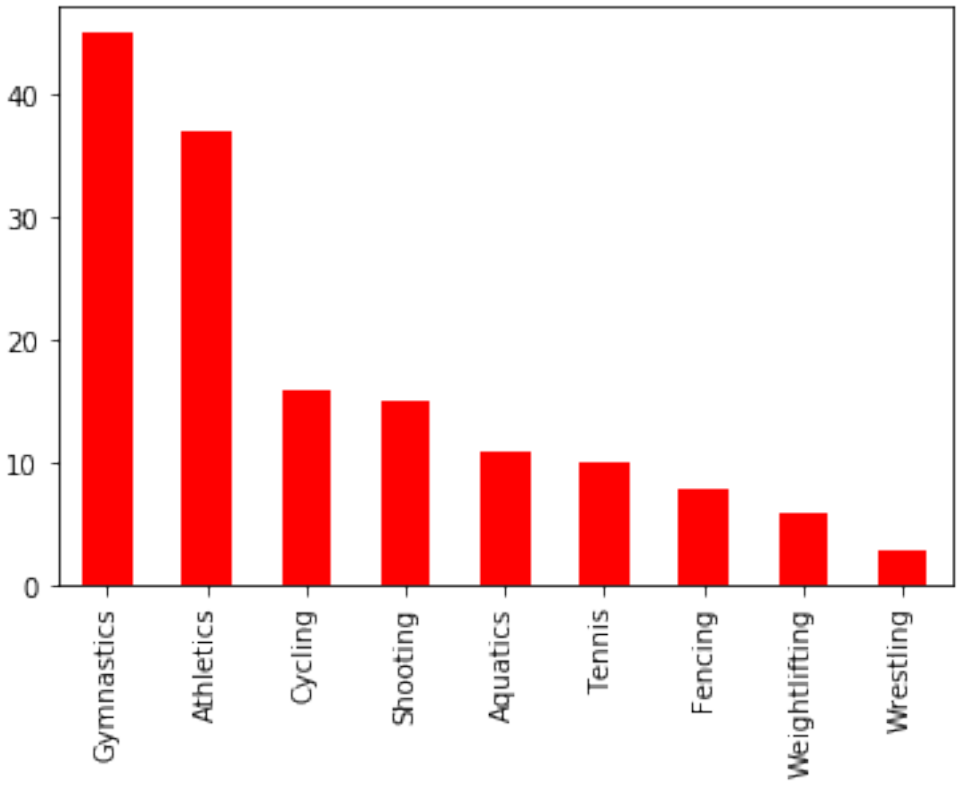
[37]:
```
# pie plot
ds1.plot(kind = 'pie');
```




### 4.8 Plot Colors

```
plt.plot(kind='<kind>',color='<col>')
```

[38]:
```
ds1.plot(kind='bar',color='red');
```

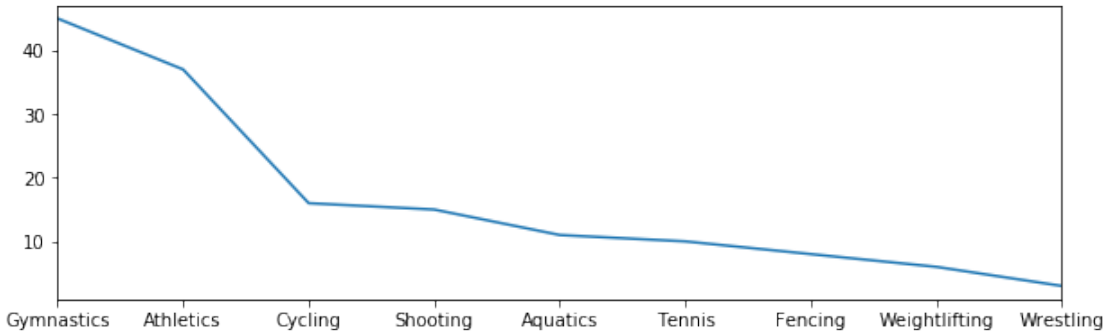



### 4.9 Figure size

A touple of (X,Y) size in inches

[39]:
```
ds1.plot(figsize=(10,3))
```

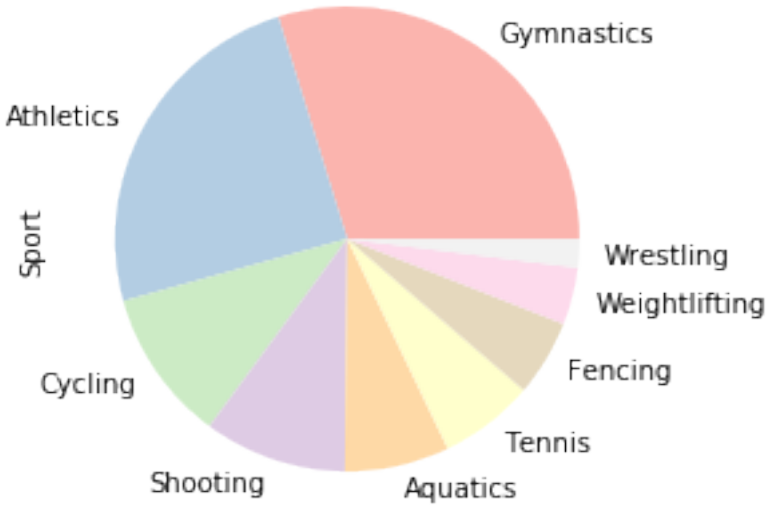
[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a910a7e6a0>
```



### 4.10 Colormaps

Three classs of maps * Sequential : used for order data * Diverging : when data deviates aroud a middle value * Qualitative : unordered color

[40]:
```
ds1.plot(kind='pie', colormap='Pastel1');
```




### 4.11 Seaborn basic plotting

- a plotting library based on matplotlib, produces reach stat plots
- not a substitute but a complement to matplotlib
- Integrate with pandas
- used over matplotlib when complex statistical data is plotted

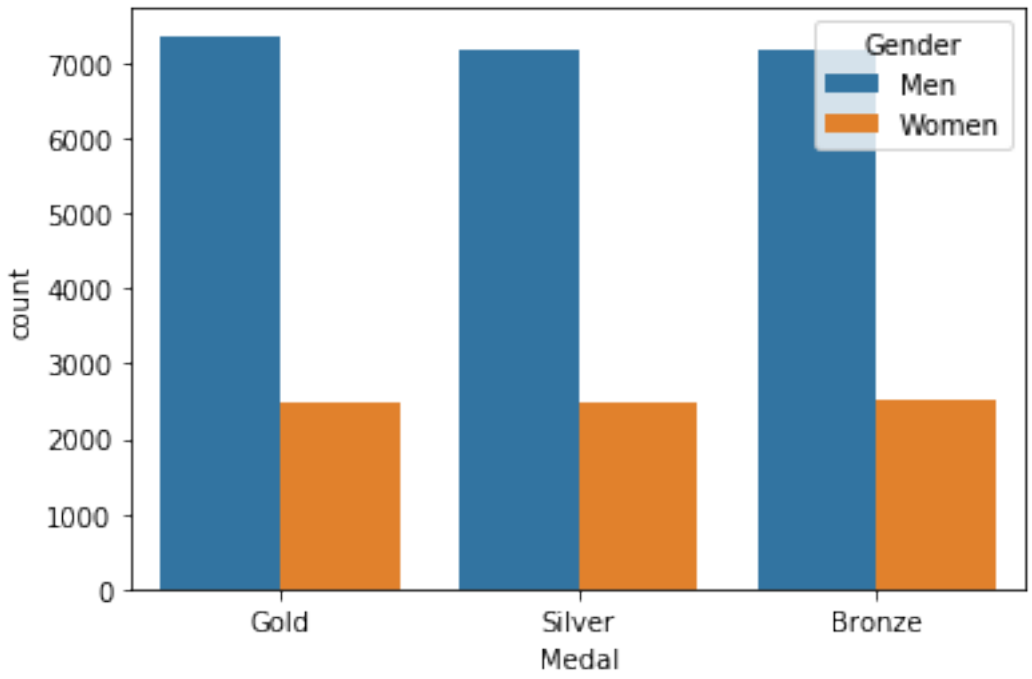
#### 4.11.1 `Countplot()`

- data : takes the df
- hue : categorical variable i.e. having a countable number classes
- order : dermines the sequence for categorical variable
- pallet : color set to use for different values

[41]:
```
import seaborn as sns
```

[42]:
```
# how many madels have been won by men and women in olympics history ?
# how may gold, silver and bronze madel are won by each gender ?
```

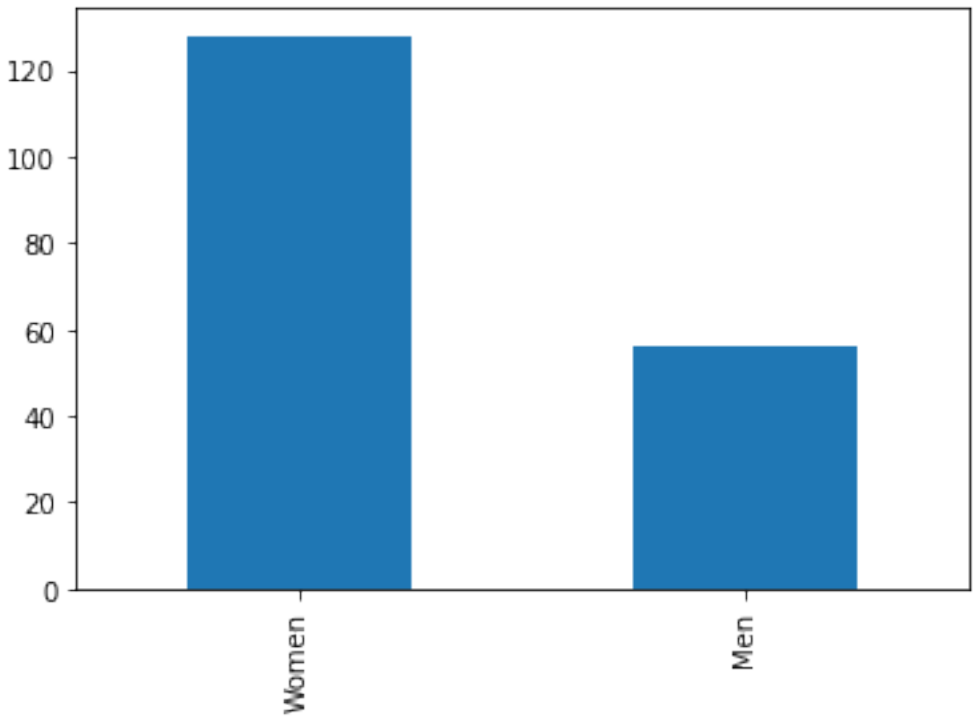
[43]:
```
sns.countplot(data=ds, x='Medal', hue='Gender');
```



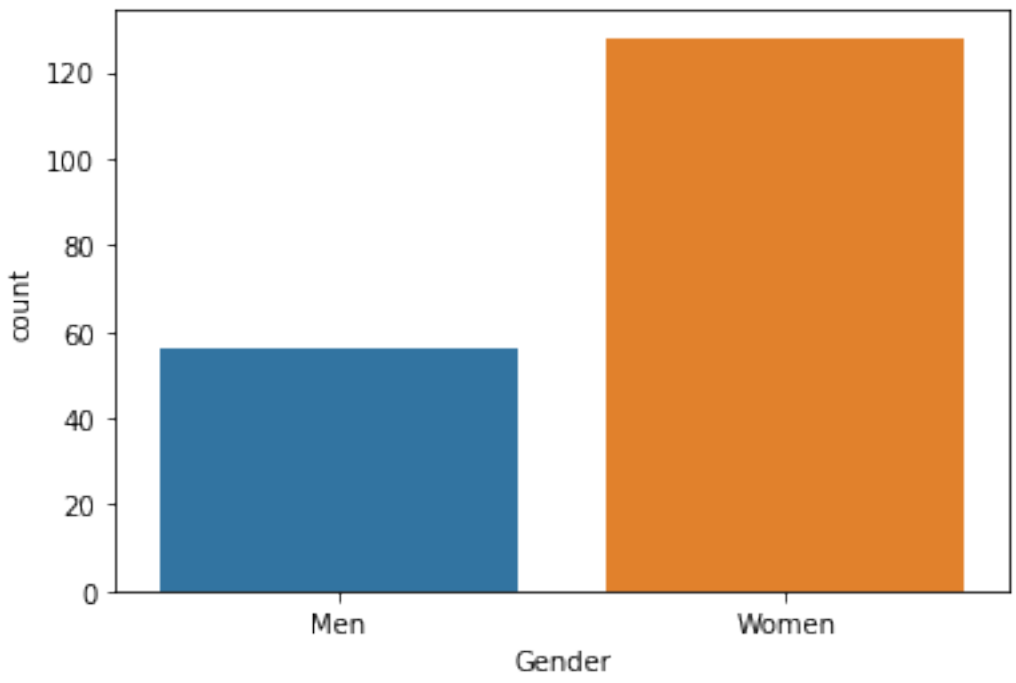

### 4.12 Problem Exersise

- plot the number of medals achieved by the Chinese team (men, women) in Beijing 2008 using
  - matplotlib
  - seaborn
- how can you use colormap to give the data more meaning ?
- plot the number of glod, silver and bronze medal for each gender
- how can you give the data more meaning, is there anything you can change ?

[44]:
```
# plot the number of medals achieved by the Chinese team (men, women) in␣
 ↪Beijing 2008 using
# matplotlib plot
ds21=ds[ (ds.Edition == 2008) & (ds.NOC == 'CHN') ]['Gender'].value_counts()
ds21.plot(kind='bar');
```




[45]:
```
# seaborn plot
ds22=ds[ (ds.Edition == 2008) & (ds.NOC == 'CHN') ]
sns.countplot(data=ds22, x='Gender');
```



[46]:
```
#how can you use colormap to give the data more meaning
ds21.plot(kind='bar',colormap='Paired',colormap=);
```

```
  File "<ipython-input-46-6d0ae70d6ac4>", line 2
    ds21.plot(kind='bar',colormap='Paired',colormap=);
                                                     ^
SyntaxError: invalid syntax
```

[ ]:
```
#plot the number of glod, silver and bronze medal for each gender
sns.countplot(data=ds22,
              x='Medal',
              hue='Gender',
              palette='bwr',
              order=['Gold', 'Silver', 'Bronze']);
```

## 5 Indexing

- The index object is an immutable array
- indexing allowes you to access a row or column using label
- index object has its own type

[53]:
```
ds.head()
```

[53]:

| | City | Edition | Sport | Discipline | Athlete | NOC | Gender | Event | Event_gender | Medal |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Athens | 1896 | Aquatics | Swimming | HAJOS, Alfred | HUN | Men | 100m freestyle | M | Gold |
| 1 | Athens | 1896 | Aquatics | Swimming | HERSCHMANN, Otto | AUT | Men | 100m freestyle | M | Silver |
| 2 | Athens | 1896 | Aquatics | Swimming | DRIVAS, Dimitrios | GRE | Men | 100m freestyle for sailors | M | Bronze |
| 3 | Athens | 1896 | Aquatics | Swimming | MALOKINIS, Ioannis | GRE | Men | 100m freestyle for sailors | M | Gold |
| 4 | Athens | 1896 | Aquatics | Swimming | CHASAPIS, Spiridon | GRE | Men | 100m freestyle for sailors | M | Silver |

[54]:
```
type(ds.index)
```

[54]:
```
pandas.core.indexes.range.RangeIndex
```

[55]:
```
ds.index[100]
```

[55]:
```
100
```

[56]:
```
# index is immutable
ds.index[100] = 5  #error!
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-56-01b5fd8fcd8b> in <module>
         1 # index is immutable
   ----> 2 ds.index[100] = 5  #error!


       ~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in␣
↪__setitem__(self, key, value)
      3936
      3937     def __setitem__(self, key, value):
   -> 3938         raise TypeError("Index does not support mutable operations")
      3939
      3940     def __getitem__(self, key):


       TypeError: Index does not support mutable operations
```

### 5.1 set_index()

[57]:
```
# set Athlete series as our index
ds.set_index('Athlete').head(3) # this is non-persistent though
```

[57]:
```
                     City  Edition     Sport Discipline  NOC Gender  \
Athlete
HAJOS, Alfred      Athens     1896  Aquatics   Swimming  HUN    Men
HERSCHMANN, Otto   Athens     1896  Aquatics   Swimming  AUT    Men
DRIVAS, Dimitrios  Athens     1896  Aquatics   Swimming  GRE    Men

                                    Event Event_gender   Medal
Athlete
HAJOS, Alfred                  100m freestyle            M    Gold
HERSCHMANN, Otto               100m freestyle            M  Silver
DRIVAS, Dimitrios  100m freestyle for sailors            M  Bronze
```

[58]:
```
ds.head(3)
```

[58]:
```
     City  Edition     Sport Discipline            Athlete  NOC Gender  \
0  Athens     1896  Aquatics   Swimming      HAJOS, Alfred  HUN    Men
```

```
1  Athens     1896  Aquatics   Swimming   HERSCHMANN, Otto  AUT    Men
2  Athens     1896  Aquatics   Swimming  DRIVAS, Dimitrios  GRE    Men

                       Event Event_gender   Medal
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
```

[59]:
```
ds3=ds.copy()   #copy DS into DS3
```

[67]:
```
ds3.set_index('Athlete',inplace=True);  #persistence change
```

[68]:
```
ds3.head(3)  # only ds3 is changed
```

[68]:
```
                     City  Edition     Sport Discipline  NOC Gender  \
Athlete
HAJOS, Alfred      Athens     1896  Aquatics   Swimming  HUN    Men
HERSCHMANN, Otto   Athens     1896  Aquatics   Swimming  AUT    Men
DRIVAS, Dimitrios  Athens     1896  Aquatics   Swimming  GRE    Men

                                     Event Event_gender   Medal
Athlete
HAJOS, Alfred                100m freestyle            M    Gold
HERSCHMANN, Otto             100m freestyle            M  Silver
DRIVAS, Dimitrios  100m freestyle for sailors            M  Bronze
```

[70]:
```
ds.head(3)  # not DS
```

[70]:
```
     City  Edition     Sport Discipline            Athlete  NOC Gender  \
0  Athens     1896  Aquatics   Swimming      HAJOS, Alfred  HUN    Men
1  Athens     1896  Aquatics   Swimming   HERSCHMANN, Otto  AUT    Men
2  Athens     1896  Aquatics   Swimming  DRIVAS, Dimitrios  GRE    Men

                       Event Event_gender   Medal
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
```

### 5.2 reset_index()

revert a dataset into its default integetr based index

[71]:
```
ds3.head(3)
```

[71]:
```
                     City  Edition     Sport Discipline  NOC Gender  \
Athlete
HAJOS, Alfred      Athens     1896  Aquatics   Swimming  HUN    Men
HERSCHMANN, Otto   Athens     1896  Aquatics   Swimming  AUT    Men
DRIVAS, Dimitrios  Athens     1896  Aquatics   Swimming  GRE    Men

                                     Event Event_gender   Medal
```

```
Athlete
HAJOS, Alfred                   100m freestyle            M    Gold
HERSCHMANN, Otto                100m freestyle            M  Silver
DRIVAS, Dimitrios  100m freestyle for sailors            M  Bronze
```

[72]:
```
ds3.reset_index(inplace=True)
```

[74]:
```
ds3.head(3)
```

[74]:
```
             Athlete    City  Edition     Sport Discipline  NOC Gender  \
0      HAJOS, Alfred  Athens     1896  Aquatics   Swimming  HUN    Men
1   HERSCHMANN, Otto  Athens     1896  Aquatics   Swimming  AUT    Men
2  DRIVAS, Dimitrios  Athens     1896  Aquatics   Swimming  GRE    Men

                        Event Event_gender   Medal
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
```

### 5.3 sort_index()

sorted index is faster to be accessed, especially when accessing a subset

[76]:
```
ds3.head(3)
```

[76]:
```
             Athlete    City  Edition     Sport Discipline  NOC Gender  \
0      HAJOS, Alfred  Athens     1896  Aquatics   Swimming  HUN    Men
1   HERSCHMANN, Otto  Athens     1896  Aquatics   Swimming  AUT    Men
2  DRIVAS, Dimitrios  Athens     1896  Aquatics   Swimming  GRE    Men

                        Event Event_gender   Medal
0              100m freestyle            M    Gold
1              100m freestyle            M  Silver
2  100m freestyle for sailors            M  Bronze
```

[83]:
```
#set the index to Athlete
ds3.set_index('Athlete',inplace=True)
ds3.head(3)
```

[83]:
```
                    City  Edition      Sport       Discipline  NOC Gender  \
Athlete
BAROEV, Khasan   Beijing     2008  Wrestling  Wrestling Gre-R  RUS    Men
LOPEZ, Mijain    Beijing     2008  Wrestling  Wrestling Gre-R  CUB    Men
PATRIKEEV, Yuri  Beijing     2008  Wrestling  Wrestling Gre-R  ARM    Men

                      Event Event_gender   Medal
Athlete
BAROEV, Khasan   96 - 120kg            M  Silver
LOPEZ, Mijain    96 - 120kg            M    Gold
PATRIKEEV, Yuri  96 - 120kg            M  Bronze
```

[84]:
```
ds3.sort_index(inplace=True, ascending=True);
ds3.head()
```

[84]:
```
                            City  Edition       Sport   Discipline  NOC  \
Athlete
AABYE, Edgar                Paris     1900  Tug of War   Tug of War  ZZX
AALTONEN, Arvo Ossian     Antwerp     1920    Aquatics     Swimming  FIN
AALTONEN, Arvo Ossian     Antwerp     1920    Aquatics     Swimming  FIN
AALTONEN, Paavo Johannes   London     1948  Gymnastics  Artistic G.  FIN
AALTONEN, Paavo Johannes   London     1948  Gymnastics  Artistic G.  FIN

                         Gender              Event Event_gender   Medal
Athlete
AABYE, Edgar                Men         tug of war            M    Gold
AALTONEN, Arvo Ossian       Men  400m breaststroke            M  Bronze
AALTONEN, Arvo Ossian       Men  200m breaststroke            M  Bronze
AALTONEN, Paavo Johannes    Men   team competition            M    Gold
AALTONEN, Paavo Johannes    Men       pommel horse            M    Gold
```

### 5.4 loc[]

- loc is a label based indexing
- raises key error when items are not found
- make sure searched label must be in index column

[86]:
```
ds3.loc['BOLT, Usain']
```

[86]:
```
                City  Edition      Sport Discipline  NOC Gender          Event  \
Athlete
BOLT, Usain  Beijing     2008  Athletics  Athletics  JAM    Men           100m
BOLT, Usain  Beijing     2008  Athletics  Athletics  JAM    Men           200m
BOLT, Usain  Beijing     2008  Athletics  Athletics  JAM    Men  4x100m relay

             Event_gender Medal
Athlete
BOLT, Usain            M  Gold
BOLT, Usain            M  Gold
BOLT, Usain            M  Gold
```

### 5.5 iloc[]

- integer based indexing
- it supports traditional pythonic slicing
- works still if the index is set to an alphanumeric attribute

[89]:
```
ds3.head(1)
```

[89]:
```
                City  Edition       Sport  Discipline  NOC Gender       Event  \
Athlete
```

```
AABYE, Edgar  Paris     1900  Tug of War  Tug of War  ZZX    Men  tug of war

             Event_gender Medal
Athlete
AABYE, Edgar            M  Gold
```

[91]:
```
ds3.iloc[0]    # notice index value is not included
```

[91]:
```
City                Paris
Edition              1900
Sport           Tug of War
Discipline      Tug of War
NOC                    ZZX
Gender                 Men
Event           tug of war
Event_gender             M
Medal                 Gold
Name: AABYE, Edgar, dtype: object
```

[97]:
```
ds3.reset_index(inplace=True)
```

[98]:
```
ds3.iloc[[0,2,5]]  # multiple rows with an index list
```

[98]:
```
                   Athlete     City  Edition       Sport   Discipline  NOC  \
0             AABYE, Edgar    Paris     1900  Tug of War   Tug of War  ZZX
2    AALTONEN, Arvo Ossian  Antwerp     1920    Aquatics     Swimming  FIN
5  AALTONEN, Paavo Johannes   London     1948  Gymnastics  Artistic G.  FIN

  Gender                 Event Event_gender   Medal
0    Men             tug of war            M    Gold
2    Men     200m breaststroke            M  Bronze
5    Men  individual all-round            M  Bronze
```

[100]:
```
ds3.iloc[2:10:2] # pythocinc slicing [start : end : step]
```

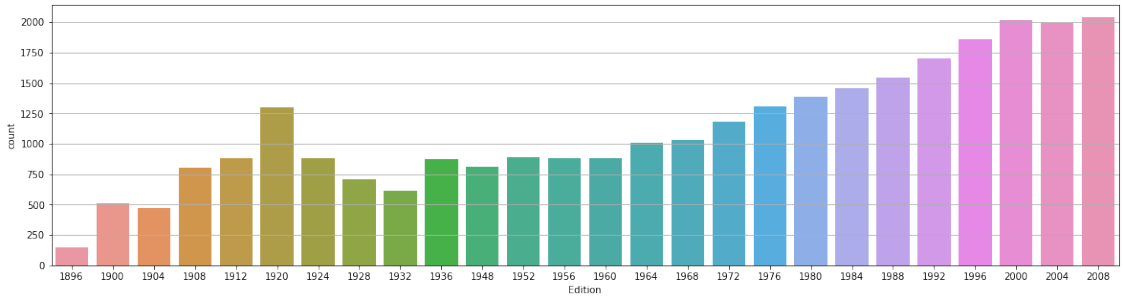
[100]:
```
                   Athlete      City  Edition       Sport   Discipline  NOC  \
2    AALTONEN, Arvo Ossian   Antwerp     1920    Aquatics     Swimming  FIN
4  AALTONEN, Paavo Johannes    London     1948  Gymnastics  Artistic G.  FIN
6  AALTONEN, Paavo Johannes  Helsinki     1952  Gymnastics  Artistic G.  FIN
8          AAMODT, Ragnhild   Beijing     2008    Handball     Handball  NOR

  Gender              Event Event_gender   Medal
2    Men  200m breaststroke            M  Bronze
4    Men       pommel horse            M    Gold
6    Men   team competition            M  Bronze
8  Women           handball            W    Gold
```

### 5.6 Problem Excersise

1. plot the total number of medals awarded to each of the Olympic games throughout history
2. which countries did not win any medals in 2008

3. how many countries were there

[116]:
```
#plot the total number of medals awarded to each of the Olympic games␣
 ↪throughout history
plt.subplots(figsize=(20,5))
plt.grid(True)
sns.countplot(data=ds, x='Edition');
```




[175]:
```
# which countries did not win any medals in 2008

noc = set(ds.NOC.tolist())    # list of all countries throught history
noc_08 = set(ds[ ds.Edition == 2008 ].NOC.tolist() ) #list of all countries won␣
 ↪a medal at 2008
sorted(list(noc.difference(noc_08))) # differece
```

[175]:
```
['AHO',
 'ANZ',
 'BAR',
 'BDI',
 'BER',
 'BOH',
 'BWI',
 'CIV',
 'CRC',
 'DJI',
 'ERI',
 'EUA',
 'EUN',
 'FRG',
 'GDR',
 'GHA',
 'GUY',
 'HAI',
 'HKG',
 'IOP',
 'IRQ',
 'ISV',
```

```
 'KSA',
 'KUW',
 'LIB',
 'LUX',
 'MKD',
 'MOZ',
 'NAM',
 'NIG',
 'PAK',
 'PAR',
 'PER',
 'PHI',
 'PUR',
 'QAT',
 'RU1',
 'SCG',
 'SEN',
 'SRI',
 'SUR',
 'SYR',
 'TAN',
 'TCH',
 'TGA',
 'UAE',
 'UGA',
 'URS',
 'URU',
 'YUG',
 'ZAM',
 'ZZX']
```

## 6 Groupby

- split df into groups based on criteria
- apply function to each group independently
- combines the results to group by objec (dict like)

[176]:
```
type(ds.groupby('Edition'))
```

[176]:
```
pandas.core.groupby.generic.DataFrameGroupBy
```

The group by object holds a dict list structure, keyed by the group by claus and valued by the matched df

[186]:
```
for group_key , group_value in ds.groupby('Edition'):
    print(f'key = {group_key} : val_type = {type(group_value)}')
```

```
key = 1896 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1900 : val_type = <class 'pandas.core.frame.DataFrame'>
```

```
key = 1904 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1908 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1912 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1920 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1924 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1928 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1932 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1936 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1948 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1952 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1956 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1960 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1964 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1968 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1972 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1976 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1980 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1984 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1988 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1992 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 1996 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 2000 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 2004 : val_type = <class 'pandas.core.frame.DataFrame'>
key = 2008 : val_type = <class 'pandas.core.frame.DataFrame'>
```

Understanding group by with multiple colum names, create a composite primary key and groups for each unique combination of keyed attributes

## 6.1 groupby computation

### 6.1.1 `size()`

counts size of a every group

[188]:
```
ds.groupby('Edition').size()
```

[188]:
```
Edition
1896     151
1900     512
1904     470
1908     804
1912     885
1920    1298
1924     884
1928     710
1932     615
1936     875
1948     814
1952     889
1956     885
```

```
1960     882
1964    1010
1968    1031
1972    1185
1976    1305
1980    1387
1984    1459
1988    1546
1992    1705
1996    1859
2000    2015
2004    1998
2008    2042
dtype: int64
```

### 6.1.2 agg([...])

calculate multiple stats for each group in one computation

[199]:
```
ds.groupby( ['Edition', 'NOC', 'Medal'] ).agg({'Edition' :␣
↪['min','max','count']})
```

[199]:

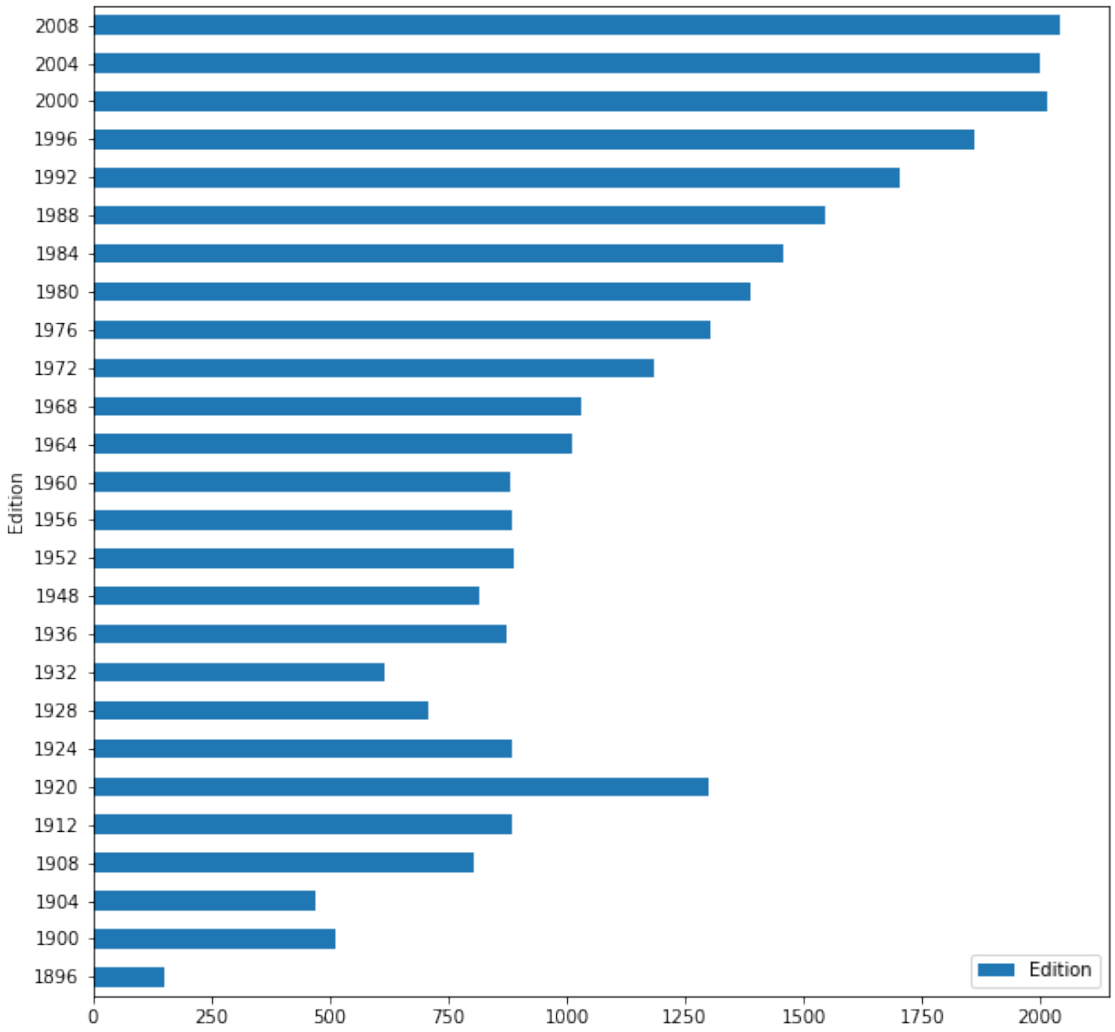
| | | | Edition | | |
|---|---|---|---|---|---|
| | | | min | max | count |
| Edition | NOC | Medal | | | |
| 1896 | AUS | Gold | 1896 | 1896 | 2 |
| | AUT | Bronze | 1896 | 1896 | 2 |
| | | Gold | 1896 | 1896 | 2 |
| | | Silver | 1896 | 1896 | 1 |
| | DEN | Bronze | 1896 | 1896 | 3 |
| | | Gold | 1896 | 1896 | 1 |
| | | Silver | 1896 | 1896 | 2 |
| | FRA | Bronze | 1896 | 1896 | 2 |
| | | Gold | 1896 | 1896 | 5 |
| | | Silver | 1896 | 1896 | 4 |
| | GBR | Bronze | 1896 | 1896 | 2 |
| | | Gold | 1896 | 1896 | 2 |
| | | Silver | 1896 | 1896 | 3 |
| | GER | Bronze | 1896 | 1896 | 2 |
| | | Gold | 1896 | 1896 | 26 |
| | | Silver | 1896 | 1896 | 5 |
| | GRE | Bronze | 1896 | 1896 | 22 |
| | | Gold | 1896 | 1896 | 10 |
| | | Silver | 1896 | 1896 | 20 |
| | HUN | Bronze | 1896 | 1896 | 3 |
| | | Gold | 1896 | 1896 | 2 |
| | | Silver | 1896 | 1896 | 1 |
| | SUI | Gold | 1896 | 1896 | 1 |

```
            Silver    1896  1896     2
        USA Bronze    1896  1896     2
            Gold      1896  1896    11
            Silver    1896  1896     7
        ZZX Bronze    1896  1896     2
            Gold      1896  1896     2
            Silver    1896  1896     2
...                     ...   ...   ...
2008    SUI Gold      2008  2008     3
        SVK Bronze    2008  2008     1
            Gold      2008  2008     4
            Silver    2008  2008     5
        SWE Bronze    2008  2008     2
            Silver    2008  2008     5
        THA Gold      2008  2008     2
            Silver    2008  2008     2
        TJK Bronze    2008  2008     1
            Silver    2008  2008     1
        TOG Bronze    2008  2008     1
        TPE Bronze    2008  2008     4
        TRI Silver    2008  2008     5
        TUN Gold      2008  2008     1
        TUR Bronze    2008  2008     3
            Gold      2008  2008     1
            Silver    2008  2008     4
        UKR Bronze    2008  2008    16
            Gold      2008  2008    10
            Silver    2008  2008     5
        USA Bronze    2008  2008    81
            Gold      2008  2008   125
            Silver    2008  2008   109
        UZB Bronze    2008  2008     3
            Gold      2008  2008     1
            Silver    2008  2008     2
        VEN Bronze    2008  2008     1
        VIE Silver    2008  2008     1
        ZIM Gold      2008  2008     1
            Silver    2008  2008     3

[2356 rows x 3 columns]
```

## 6.2 Problem Excersise

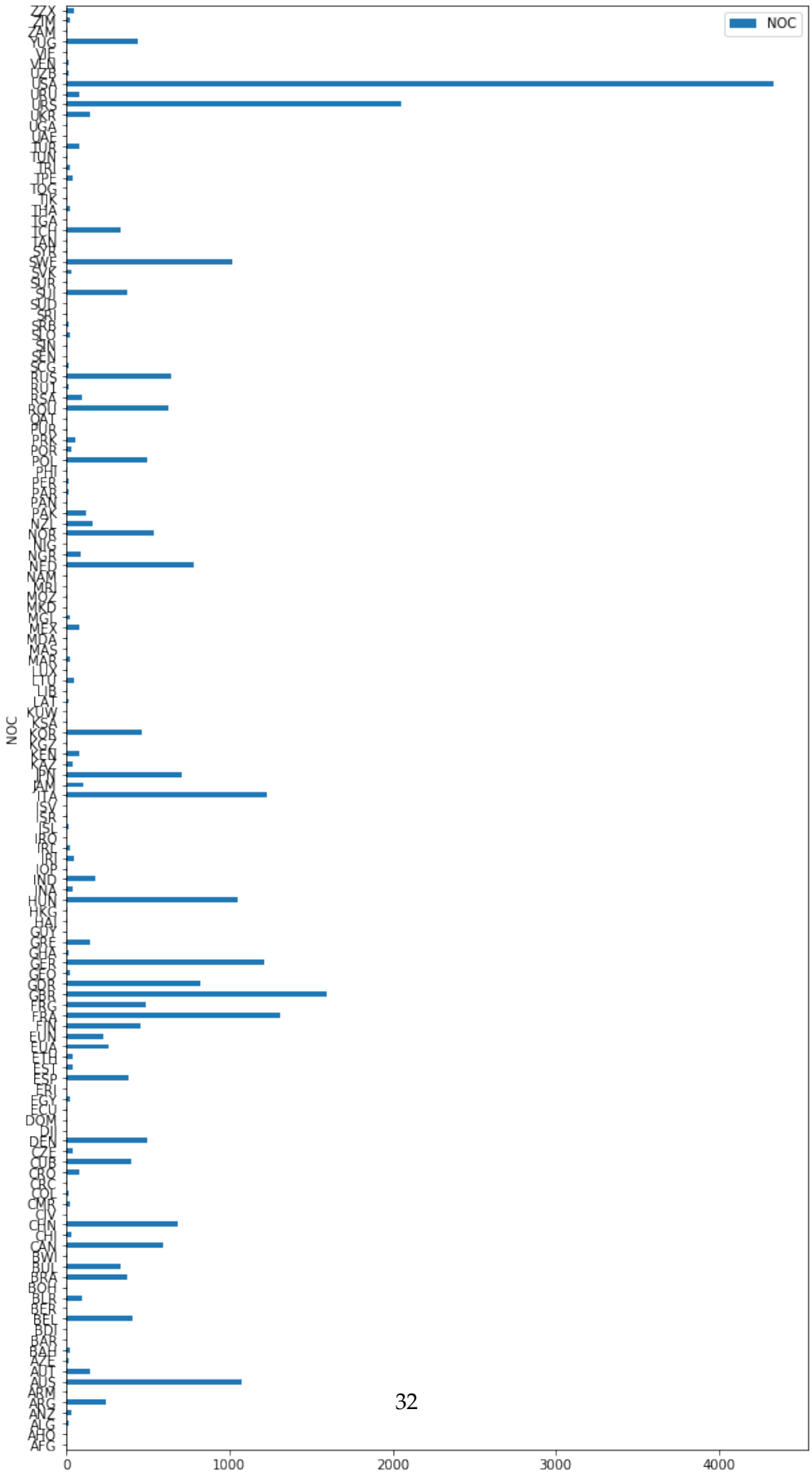
1. list the total number of medal given to every year throughout history
2. list total number of medal won by any country throughout history

[214]:
```
#list the total number of medal given to every year throughout history

ds.groupby('Edition').agg({'Edition':'count'}).
 ↪plot(kind='barh',figsize=(10,10));
```




[219]:
```
#list total number of medal won by any country throughout history
ds.groupby('NOC').agg({'NOC':'count'}).plot(kind='barh',figsize=(10,20))
```

[219]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a92266aa58>
```


NOC
0
1000
2000
3000
4000

### 6.3 Reshaping

[232]:
```
#athlete that took part in Beijing olympic 100m or 200m event
bj_100_200 = ds[ (ds.City == 'Beijing') & ((ds.Event == '100m') | (ds.Event ==␣
 ↪'200m'))]
bj_100_200
```

[232]:
```
          City  Edition      Sport Discipline                  Athlete  NOC  \
27551  Beijing     2008  Athletics  Athletics              DIX, Walter  USA
27552  Beijing     2008  Athletics  Athletics              BOLT, Usain  JAM
27553  Beijing     2008  Athletics  Athletics         THOMPSON, Richard  TRI
27554  Beijing     2008  Athletics  Athletics        FRASER, Shelly-ann  JAM
27555  Beijing     2008  Athletics  Athletics          SIMPSON, Sherone  JAM
27556  Beijing     2008  Athletics  Athletics           STEWART, Kerron  JAM
27569  Beijing     2008  Athletics  Athletics              DIX, Walter  USA
27570  Beijing     2008  Athletics  Athletics              BOLT, Usain  JAM
27571  Beijing     2008  Athletics  Athletics           CRAWFORD, Shawn  USA
27572  Beijing     2008  Athletics  Athletics           STEWART, Kerron  JAM
27573  Beijing     2008  Athletics  Athletics  CAMPBELL-BROWN, Veronica  JAM
27574  Beijing     2008  Athletics  Athletics            FELIX, Allyson  USA

      Gender Event Event_gender   Medal
27551    Men  100m            M  Bronze
27552    Men  100m            M    Gold
27553    Men  100m            M  Silver
27554  Women  100m            W    Gold
27555  Women  100m            W  Silver
27556  Women  100m            W  Silver
27569    Men  200m            M  Bronze
27570    Men  200m            M    Gold
27571    Men  200m            M  Silver
27572  Women  200m            W  Bronze
27573  Women  200m            W    Gold
27574  Women  200m            W  Silver
```

[233]:
```
g=bj_100_200.groupby(['NOC', 'Gender', 'Discipline', 'Event']).size()
g
```

[233]:
```
NOC  Gender  Discipline  Event
JAM  Men     Athletics   100m     1
                         200m     1
     Women   Athletics   100m     3
                         200m     2
TRI  Men     Athletics   100m     1
USA  Men     Athletics   100m     1
```

```
                          200m     2
     Women   Athletics    200m     1
dtype: int64
```

[236]:
```
ds_us = g.unstack(['Discipline','Event'])
ds_us
```

[236]:
```
Discipline Athletics
Event           100m 200m
NOC Gender
JAM Men          1.0  1.0
    Women        3.0  2.0
TRI Men          1.0  NaN
USA Men          1.0  2.0
    Women        NaN  1.0
```

### 6.4 stack()

- returns a dataframe or a series
- Pivot a level of the column level, returning a DF or Series with a new innermost level of row level
- Stacking makes your table taller

[237]:
```
ds_us
```

[237]:
```
Discipline Athletics
Event           100m 200m
NOC Gender
JAM Men          1.0  1.0
    Women        3.0  2.0
TRI Men          1.0  NaN
USA Men          1.0  2.0
    Women        NaN  1.0
```

[238]:
```
ds_us.stack()
```

[238]:
```
Discipline        Athletics
NOC Gender Event
JAM Men    100m         1.0
           200m         1.0
    Women  100m         3.0
           200m         2.0
TRI Men    100m         1.0
USA Men    100m         1.0
           200m         2.0
    Women  200m         1.0
```

### 6.5 unstack()

- opposite to stacking

- unstacking makes your table wider

[ ]:

# 7 Data Visualization

## 7.1 Heatmap

[243]:
```
# all medal won at 2008 olympic
ds_2008 = ds[ ds.Edition == 2008 ];

# group by NOC and Medal and count them, unstack the medal and fill N/A with 0
gp_2008 = ds_2008.groupby( ['NOC','Medal'] ).size().
 ↪unstack('Medal',fill_value=0);

gp_2008.head(10)
```

[243]:
| Medal | Bronze | Gold | Silver |
|---|---|---|---|
| NOC | | | |
| AFG | 1 | 0 | 0 |
| ALG | 1 | 0 | 1 |
| ARG | 31 | 20 | 0 |
| ARM | 6 | 0 | 0 |
| AUS | 76 | 31 | 42 |
| AUT | 2 | 0 | 1 |
| AZE | 4 | 1 | 2 |
| BAH | 1 | 0 | 4 |
| BEL | 0 | 1 | 4 |
| BLR | 17 | 8 | 5 |

[250]:
```
#sort medals and alter the column orders

srt_08 = gp_2008.sort_values(['Gold','Silver','Bronze'], ascending=False)[␣
 ↪['Gold','Silver','Bronze'] ]
srt_08.head(10)
```

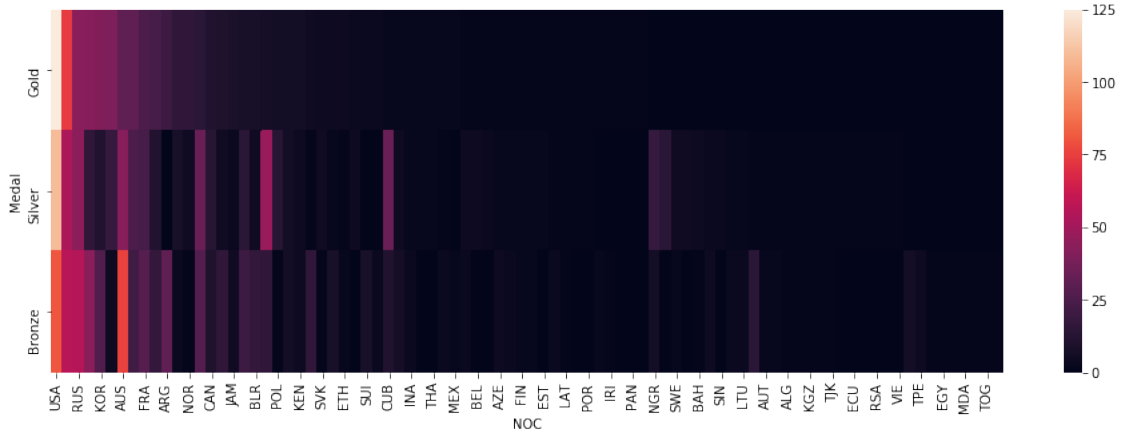
[250]:
| Medal | Gold | Silver | Bronze |
|---|---|---|---|
| NOC | | | |
| USA | 125 | 109 | 81 |
| CHN | 74 | 53 | 57 |
| RUS | 43 | 44 | 56 |
| GER | 42 | 16 | 43 |
| KOR | 41 | 11 | 26 |
| NED | 40 | 18 | 4 |
| AUS | 31 | 42 | 76 |
| GBR | 31 | 25 | 21 |
| FRA | 25 | 23 | 28 |
| JPN | 23 | 11 | 17 |

[252]:
```
srt_08.transpose()
```

[252]:
```
NOC     USA  CHN  RUS  GER  KOR  NED  AUS  GBR  FRA  JPN  ...  VIE  ARM  TPE  \
Medal                                                     ...
Gold    125   74   43   42   41   40   31   31   25   23  ...    0    0    0
Silver  109   53   44   16   11   18   42   25   23   11  ...    1    0    0
Bronze   81   57   56   43   26    4   76   21   28   17  ...    0    6    4

NOC     AFG  EGY  ISR  MDA  MRI  TOG  VEN
Medal
Gold      0    0    0    0    0    0    0
Silver    0    0    0    0    0    0    0
Bronze    1    1    1    1    1    1    1

[3 rows x 86 columns]
```

[254]:
```
plt.figure(figsize=(16,5),)
sns.heatmap(data=srt_08.transpose())
```

[254]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a91fa5f518>
```




## 7.2 Creating custom colormaps

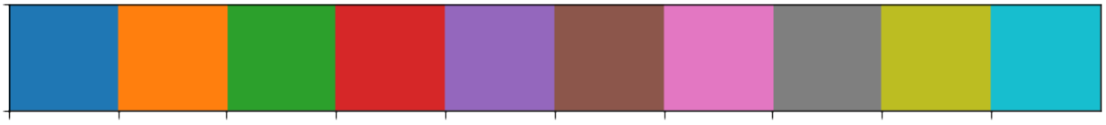
[255]:
```
from matplotlib.colors import ListedColormap
```

[257]:
```
sns.color_palette()
```

[257]:
```
[(0.12156862745098039, 0.4666666666666667, 0.7058823529411765),
 (1.0, 0.4980392156862745, 0.054901960784313725),
 (0.17254901960784313, 0.6274509803921569, 0.17254901960784313),
 (0.8392156862745098, 0.15294117647058825, 0.1568627450980392),
 (0.5803921568627451, 0.403921568627451, 0.7411764705882353),
 (0.5490196078431373, 0.33725490196078434, 0.29411764705882354),
 (0.8901960784313725, 0.4666666666666667, 0.7607843137254902),
```

```
(0.4980392156862745, 0.4980392156862745, 0.4980392156862745),
(0.7372549019607844, 0.7411764705882353, 0.13333333333333333),
(0.09019607843137255, 0.7450980392156863, 0.8117647058823529)]
```

[260]:
```
sns.palplot(sns.color_palette())
```



[264]:
```
#create a new pallet
gsb=['#FFD700','#C0C0C0','#cd7f32']
```

[265]:
```
plt.figure(figsize=(16,5),)
sns.heatmap(data=srt_08.transpose(),cmap=gsb)
```

[265]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a922db7ef0>
```