ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Introduction to Information Systems and Applications

Course Unit 2: Data processing with python

M. Tzagarakis, V. Daskalou

School of Business Administration

Department of Economics

Pandas

Why Pandas?

- Open-source python library for data analysis
- Include easy-to-use data structures and data analysis tools
- Used in a wide range of fields in academic and commercial domains, such as:
  - statistics,
  - analytics,
  - finance,
  - Economics
- Documentation:
  - https://pandas.pydata.org/pandas-docs/stable/index.html
  - 10 Minutes to pandas
  - Tutorials

Issues related to installation

- Better use pandas with Anaconda Python package
  - recall anaconda installation from Intro to Python
- Pandas library uses most of the functionalities of NumPy:
  - NumPy stands for Numerical Python
  - Is a library consisting of multidimensional array objects and a collection of routines for processing those arrays

```
In [1]: import pandas as pd

In [2]: import numpy as np

In [3]: import matplotlib.pyplot as plt
```

Data types

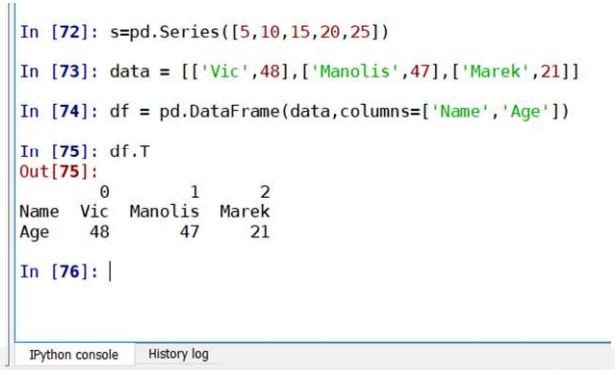
- Three (3) data types: Series, DataFrame, Panel
- Series:
  - 1 dimension
  - Homogeneous data, Size Immutable, Values of Data Mutable
- DataFrame:
  - 2 dimensions (tubular data: index or rows and columns)
  - Heterogeneous data, Size Mutable, Data Mutable
- Panel:
  - 3 dimensions
  - Heterogeneous data, Size Mutable, Data Mutable

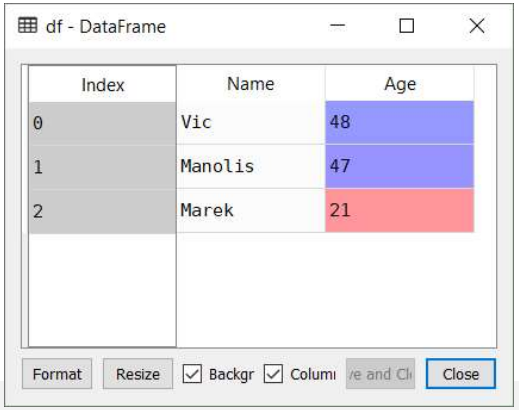
Examples of data types (1)

```
In [4]: s=pd.Series([5,10,15,20,25])
In [5]: s
Out[5]:
0     5
1    10
2    15
3    20
4    25
dtype: int64
```

Create a DataFrame from a list

```
In [6]: data = [['Vic',48],['Manolis',47],['Marek',21]]
In [7]: df = pd.DataFrame(data,columns=['Name','Age'])
In [8]: df
Out[8]:
Name Age
0 Vic 48
1 Manolis 47
2 Marek 21
```

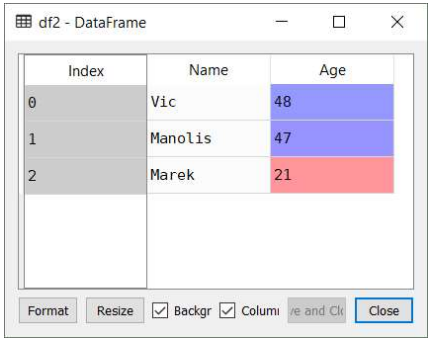




Examples of data types (2)

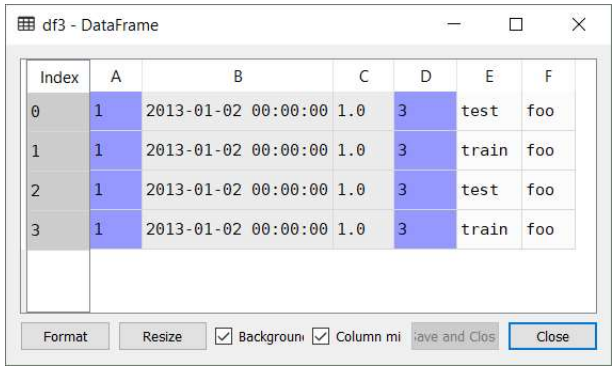
Create a DataFrame from a dictionary

```
In [11]: data2={'Name':['Vic','Manolis','Marek'],'Age':[48,47,21]}
In [12]: df2=pd.DataFrame(data=data2)
In [13]: df2
Out[13]:
Name Age
0 Vic 48
1 Manolis 47
2 Marek 21
```



Create a DataFrame of different column types

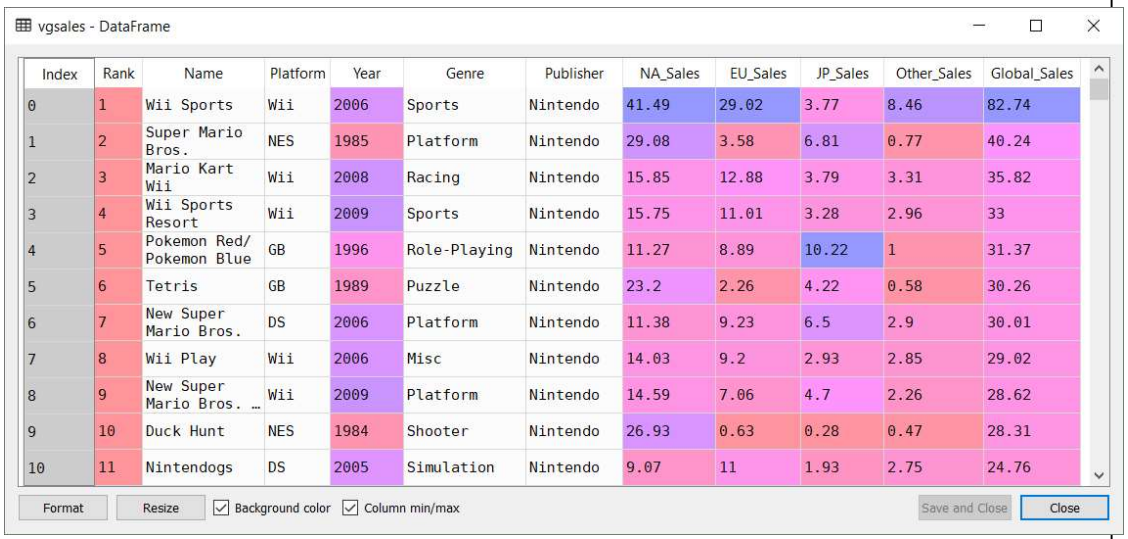
```
In [15]: df3 = pd.DataFrame({ 'A' : 1.,
    ...: 'B' : pd.Timestamp('20130102'),
    ...: 'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
    ...: 'D' : np.array([3] * 4,dtype='int32'),
    ...: 'E' : pd.Categorical(["test","train","test","train"]),
    ...: 'F' : 'foo' })
In [16]: df3.dtypes
Out[16]:
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```



More: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html#pandas.DataFrame

Read from a CSV file

```
In [21]: vgsales=pd.read_csv('vgsales.csv')
In [22]: vgsales.dtypes
Out[22]:
Rank int64
Name object
Platform object
Year float64
Genre object
Publisher object
NA_Sales float64
EU_Sales float64
JP_Sales float64
Other_Sales float64
Global_Sales float64
dtype: object
```



Index	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33
4	5	Pokemon Red/ Pokemon Blue	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1	31.37
5	6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
6	7	New Super Mario Bros.	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
7	8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
8	9	New Super Mario Bros. …	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
9	10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31
10	11	Nintendogs	DS	2005	Simulation	Nintendo	9.07	11	1.93	2.75	24.76

View top and bottom rows of frame

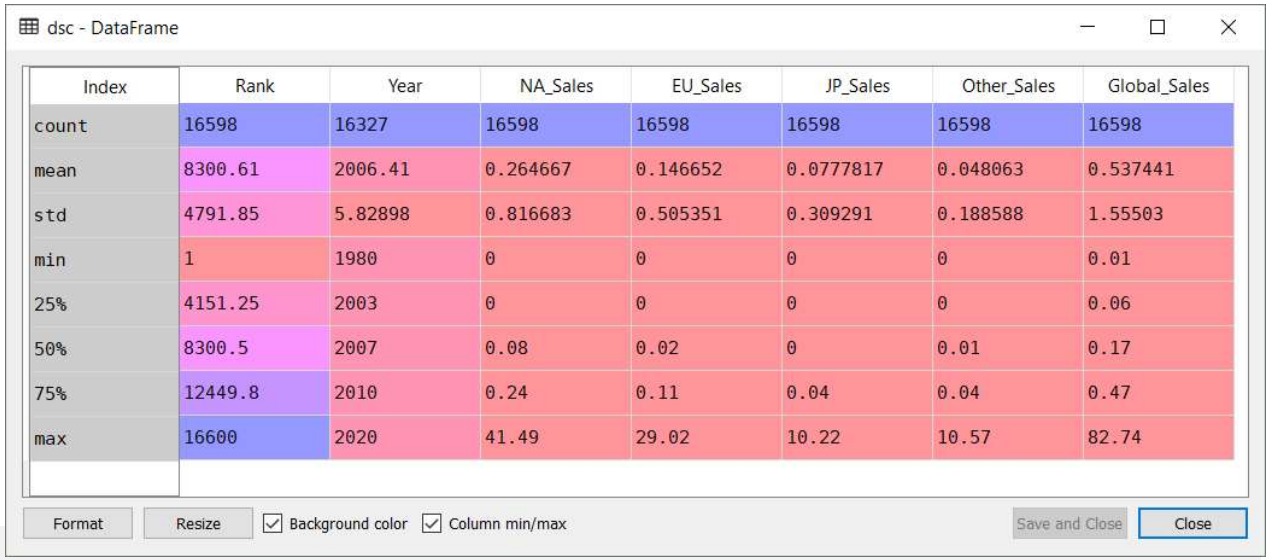
```
In [23]: vgsales.head()
Out[23]:
   Rank                      Name      ...       Other_Sales  Global_Sales
0     1                Wii Sports      ...              8.46         82.74
1     2         Super Mario Bros.      ...              0.77         40.24
2     3            Mario Kart Wii      ...              3.31         35.82
3     4         Wii Sports Resort      ...              2.96         33.00
4     5  Pokemon Red/Pokemon Blue      ...              1.00         31.37

[5 rows x 11 columns]
In [24]: vgsales.tail(5)
Out[24]:
        Rank     ...      Global_Sales
16593  16596     ...              0.01
16594  16597     ...              0.01
16595  16598     ...              0.01
16596  16599     ...              0.01
16597  16600     ...              0.01

[5 rows x 11 columns]
```

Index, columns, and underlying data

```
In [25]: vgsales.index [check if returns a range object]
Out[25]: RangeIndex(start=0, stop=16598, step=1)

In [26]: vgsales.columns
Out[26]:
Index(['Rank', 'Name', 'Platform', 'Year', 'Genre', 'Publisher', 'NA_Sales',
       'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales'],
      dtype='object')
In [32]: vgsales.values
Out[32]:
array([[1, 'Wii Sports', 'Wii', ..., 3.77, 8.46, 82.74],
       [2, 'Super Mario Bros.', 'NES', ..., 6.81, 0.77, 40.24],
       [3, 'Mario Kart Wii', 'Wii', ..., 3.79, 3.31, 35.82],
       ...,
       [16598, 'SCORE International Baja 1000: The Official Game', 'PS2',
        ..., 0.0, 0.0, 0.01],
       [16599, 'Know How 2', 'DS', ..., 0.0, 0.0, 0.01],
       [16600, 'Spirits & Spells', 'GBA', ..., 0.0, 0.0, 0.01]],
      dtype=object)
```

Descriptive statistics

```
In [29]: dsc=vgsales.describe()

In [30]: dsc
    ...:
Out[30]:
                Rank          Year      ...        Other_Sales  Global_Sales
count  16598.000000  16327.000000      ...       16598.000000  16598.000000
mean    8300.605254   2006.406443      ...           0.048063      0.537441
std     4791.853933      5.828981      ...           0.188588      1.555028
min        1.000000   1980.000000      ...           0.000000      0.010000
25%     4151.250000   2003.000000      ...           0.000000      0.060000
50%     8300.500000   2007.000000      ...           0.010000      0.170000
75%    12449.750000   2010.000000      ...           0.040000      0.470000
max    16600.000000   2020.000000      ...          10.570000     82.740000

[8 rows x 7 columns]
```



Index	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16598	16327	16598	16598	16598	16598	16598
mean	8300.61	2006.41	0.264667	0.146652	0.0777817	0.048063	0.537441
std	4791.85	5.82898	0.816683	0.505351	0.309291	0.188588	1.55503
min	1	1980	0	0	0	0	0.01
25%	4151.25	2003	0	0	0	0	0.06
50%	8300.5	2007	0.08	0.02	0	0.01	0.17
75%	12449.8	2010	0.24	0.11	0.04	0.04	0.47
max	16600	2020	41.49	29.02	10.22	10.57	82.74

Getting data

Getting a column

```
In [11]: vgsales['Name']
Out[11]:
0 Wii Sports
1 Super Mario Bros.
2 Mario Kart Wii
3 Wii Sports Resort
4 Pokemon Red/Pokemon Blue
5 Tetris
6 New Super Mario Bros.
7 Wii Play
8 New Super Mario Bros. Wii
9 Duck Hunt
10 Nintendogs
11 Mario Kart DS
12 Pokemon Gold/Pokemon Silver

...
16596 Know How 2
16597 Spirits & Spells
Name: Name, Length: 16598, dtype: object
```

Getting rows with slices

```
In [12]: vgsales[3:5]
Out[12]:
   Rank                      Name  ...  Other_Sales  Global_Sales
3     4          Wii Sports Resort  ...         2.96         33.00
4     5  Pokemon Red/Pokemon Blue  ...         1.00         31.37

[2 rows x 11 columns]
```

Select data by label: loc

```
# extract a subset of dataframe. Specify slices for rows and columns.
# more about DataFrame.loc https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html
print(vgsales.loc[0:5,'Name':'Year'])
#loc with only label (row) returns a series object
x=vgsales.loc[0]
print(x)
#loc with a list of labels for rows returns a dataframe
y=vgsales.loc[[0,5]]
print(y)
#loc with single label for row and column returns a cell, the Name of row 0
z=vgsales.loc[0,'Name']
#Slice with labels for row and single label for column. Return a Series object
a=vgsales.loc[3:7,'Name']
# loc specific columns for all rows
b=vgsales.loc[:,['Name','Global_Sales']]
#conditional
c=vgsales.loc[vgsales['Rank']<5]
z=vgsales.loc[((vgsales['Rank']>=1) & (vgsales['Rank']<=100))]
#convert dataframe and use to_string() to print it as a whole
print(z.to_string())
#Using the isin() method for filtering
d=vgsales[vgsales['Platform'].isin(['Wii','DS','PS4’])]
```

Selection by position: iloc (1/2)

Return the row in a specific position

```
In [4]: vgsales.iloc[5]
Out[4]:
Rank                    6
Name               Tetris
Platform               GB
Year                 1989
Genre              Puzzle
Publisher        Nintendo
NA_Sales             23.2
EU_Sales             2.26
JP_Sales             4.22
Other_Sales          0.58
Global_Sales        30.26
Name: 5, dtype: object
```

Specify slices with integers

```
In [6]: vgsales.iloc[0:3,0:3]
Out[6]:
Rank Name Platform
0 1 Wii Sports Wii
1 2 Super Mario Bros. NES
2 3 Mario Kart Wii Wii
```

Selection by position: iloc (2/2)

Specify lists of specific integer position locations (can specify also only rows or columns)

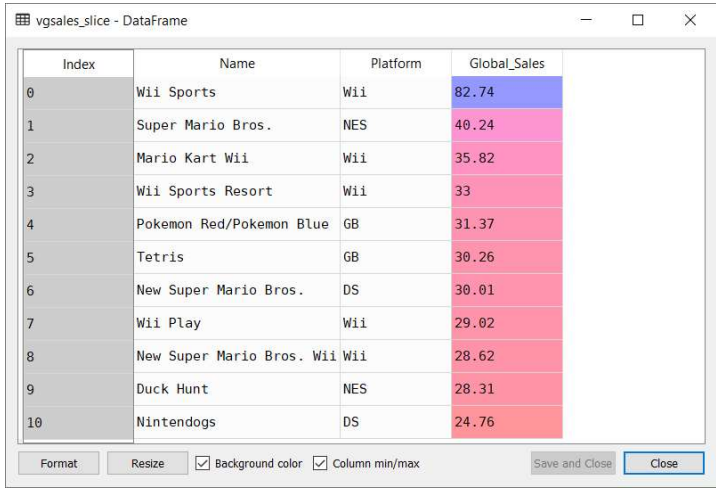
```
In [7]: vgsales.iloc[[1,2,4],[0,2]]
Out[7]:
   Rank Platform
1     2      NES
2     3      Wii
4     5       GB
```

Specify an exact cell position

```
In [8]: vgsales.iloc[1][1]

Out[8]: 'Super Mario Bros.'
```

Sorting

```
In [111]: vgsales_slice.sort_values(by='Name')
Out[111]:
                          Name Platform  Global_Sales
9                    Duck Hunt      NES         28.31
2                Mario Kart Wii      Wii         35.82
6        New Super Mario Bros.       DS         30.01
8    New Super Mario Bros. Wii      Wii         28.62
10                   Nintendogs       DS         24.76
4     Pokemon Red/Pokemon Blue       GB         31.37
1            Super Mario Bros.      NES         40.24
5                      Tetris       GB         30.26
7                    Wii Play      Wii         29.02
0                  Wii Sports      Wii         82.74
3            Wii Sports Resort      Wii         33.00
```



Setting

Create a DataFrame from a list of list

```
In [12]:
data=pd.DataFrame([[1,2,3],[4,5,6]],
columns=['A','B','C'])

In [13]: data
    ...:
Out[13]:
A B C
0 1 2 3
1 4 5 6
```

Create a new column in DataFrame (with value 1)

```
In [14]: data['E']=1

In [15]: data
Out[15]:
A B C E
0 1 2 3 1
1 4 5 6 1
```

Set value to a column (value 2 to column E)

```
In [16]: data.loc[:,'E']=2
In [17]: data
Out[17]:
A B C E
0 1 2 3 2
1 4 5 6 2
```

Set a value to a specific cell (value 10 to cell [0,0])

```
In [18]: data.iloc[0,0]=10
In [19]: data
Out[19]:
A B C E
0 10 2 3 2
1 4 5 6 2
```

Set values based on conditions (set 0 to cells with value >2)

```
In [20]: data[data>2]=0
In [21]: data
Out[21]:
A B C E
0 0 2 0 2
1 0 0 0 2
```

Operations

- Statistics
- Apply
- Aggregate
- Histogramming
- Grouping
- Pivot tables

Operations: Statistics

Create a DataFrame from a list of list (3 rows, 3 columns)

```
data=pd.DataFrame([[1,2,3],[4,5,6],[7,8,9]],
columns=['A','B','C'])
data
Out[24]:
   A  B  C
0  1  2  3
1  4  5  6
2  7  8  9
```

Calculate mean of values in requested axis (by default axis 0 is index)

```
data.mean()
Out[25]:
A    4.0
B    5.0
C    6.0
dtype: float64
```

Calculate mean of values in requested axis (1 is by rows)

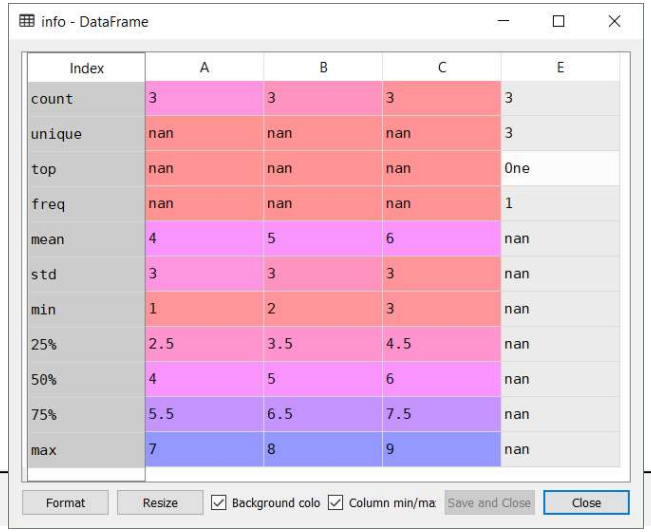
```
data.mean(1)
Out[26]:
0    2.0
1    5.0
2    8.0
dtype: float64
```

Calculate sum of values in requested axis (by default axis 0 is index)

```
In [27]: data.sum()
Out[27]:
A    12
B    15
C    18
dtype: int64
```

Calculate descriptive statistics to all type of columns

```
In [30]: data['E']=['One','Two','Three']
In [31]: data
Out[31]:
A B C E
0 1 2 3 One
1 4 5 6 Two
2 7 8 9 Three
In [32]: info=data.describe(include='all')
```



Operations: apply

applymap() applies a function to every single element in the entire dataframe.

More about apply functions: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.apply.html

```
import pandas as pd
# create a function called add10
def add10(x):
    # that, if x is a string,
    if type(x) is str:
        # just returns it untouched
        return x
    # but, if not, return it by adding 10
    elif x:
        return x+10
    # and leave everything else
    else:
        return
#create a dataframe
data=pd.DataFrame([[1,2,3],[4,5,6],[7,8,9]], columns=['A','B','C’])
#apply function add10 to each element of dataframe data
data10=data.applymap(add10)
```

Operations: aggregate

Aggregate using one or more operations over the specified axis. agg is an alias for aggregate. Use the alias.

```
In [43]: data
Out[43]:
   A  B  C
0  1  2  3
1  4  5  6
2  7  8  9
```

Aggregate these functions over the rows.

```
In [44]: data.agg(['sum', 'min'])
Out[44]:
      A   B   C
sum  12  15  18
min   1   2   3
```

Aggregate over the columns.

```
In [45]: data.agg("mean", axis="columns")
Out[45]:
0    2.0
1    5.0
2    8.0
dtype: float64
```

Different aggregations per column.

```
In [46]: data.agg({'A' : ['sum', 'min'], 'B' : ['min', 'max']})
Out[46]:
        A    B
max   NaN  8.0
min   1.0  2.0
sum  12.0  NaN
```

Operations: Histogramming

Series.value_counts(): Returns object containing counts of unique values.

```
In [55]: data
Out[55]:
   A  B  C    E
0  7  2  3  Foo
1  4  5  6  Bar
2  7  8  9  Foo

In [56]: data['A'].value_counts()
Out[56]:
7    2
4    1
Name: A, dtype: int64

In [57]: data['E'].value_counts()
Out[57]:
Foo    2
Bar    1
Name: E, dtype: int64
```

Grouping

Get a slice of vgsales DataFrame from row 0 to 10 and the columns Name, Platform and Global_Sales

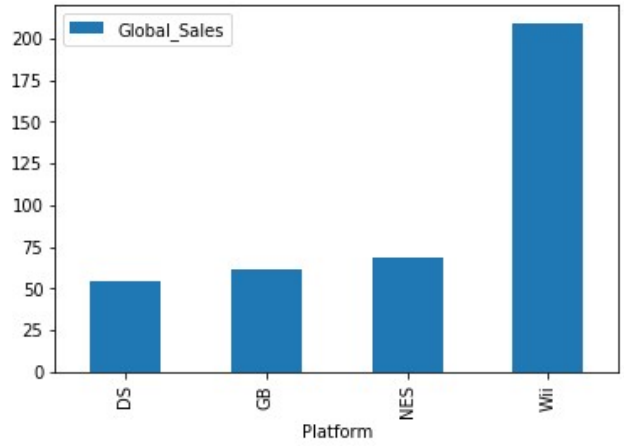
```
In [64]: vgsales_slice=vgsales.loc[0:10,['Name','Platform','Global_Sales']]
In [65]: vgsales_slice
Out[65]:
                         Name Platform  Global_Sales
0                  Wii Sports      Wii         82.74
1           Super Mario Bros.      NES         40.24
2              Mario Kart Wii      Wii         35.82
3           Wii Sports Resort      Wii         33.00
4    Pokemon Red/Pokemon Blue       GB         31.37
5                      Tetris       GB         30.26
6       New Super Mario Bros.       DS         30.01
7                    Wii Play      Wii         29.02
8   New Super Mario Bros. Wii      Wii         28.62
9                   Duck Hunt      NES         28.31
10                 Nintendogs       DS         24.76
```

Grouping and then applying the sum() function to the resulting groups.

```
In [66]: vgsales_slice.groupby('Platform').sum()
Out[66]:
Platform    Global_Sales
DS                 54.77
GB                 61.63
NES                68.55
Wii               209.20
```

Pivot tables (1/2)

```
In [120]: pv=vgsales_slice.pivot_table(index='Platform',aggfunc='sum')
In [124]: pv
Out[124]:
          Global_Sales
Platform
DS               54.77
GB               61.63
NES              68.55
Wii             209.20
In [125]: pv.plot(kind='bar')
Out[125]: <matplotlib.axes._subplots.AxesSubplot at 0x1be4d569630>
```



Pivot tables (1/2)

```
vgsales_slice2=vgsales.loc[0:20]

In [139]: pv=vgsales_slice2.pivot_table('NA_Sales',index='Publisher',
columns='Genre',aggfunc='sum',margins=True)
```

pv - DataFrame

Index	Action	Misc	Platform	Puzzle	Racing	Role-Playing	Shooter	Simulation	Sports	All
Microsoft Game Studios	nan	14.97	nan	nan	nan	nan	nan	nan	nan	14.97
Nintendo	nan	18.78	67.83	23.2	25.66	26.69	26.93	9.07	75.27	273.43
Take-Two Interactive	16.44	nan	nan	nan	nan	nan	nan	nan	nan	16.44
All	16.44	33.75	67.83	23.2	25.66	26.69	26.93	9.07	75.27	304.84

Format Resize Background color Column min/max Save and Close Close

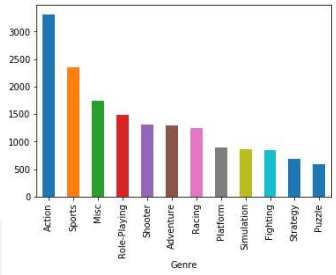
Categoricals

Pandas can include categorical data in a DataFrame.

```
In [144]:
vgsales['Platform']=vgsales['Platform'].astype('category')
In [145]:
vgsales['Year']=vgsales['Year'].astype('category')
In [146]:
vgsales['Genre']=vgsales['Genre'].astype('category')
In [147]: vgsales.dtypes
Out[147]:
Rank               int64
Name              object
Platform        category
Year            category
Genre           category
Publisher         object
NA_Sales         float64
EU_Sales         float64
JP_Sales         float64
Other_Sales      float64
Global_Sales     float64
Dtype: object
```

Grouping by a categorical column and then apply the size() function can give us a frequency table of a categorical column (histogram)

```
In [154]:
vgsales.groupby('Genre').size().sort_values(ascending=Fa
lse)
Out[154]:
Genre
Action          3316
Sports          2346
Misc            1739
Role-Playing    1488
Shooter         1310
Adventure       1286
Racing          1249
Platform         886
Simulation       867
Fighting         848
Strategy         681
Puzzle           582
dtype: int64
In [155]:
vgsales_by_genre=vgsales.groupby('Genre').size().sort_va
lues(ascending=False)
In [156]: vgsales_by_genre.plot(kind='bar')
Out[156]: <matplotlib.axes._subplots.AxesSubplot at
0x1be4d68b5c0>
```

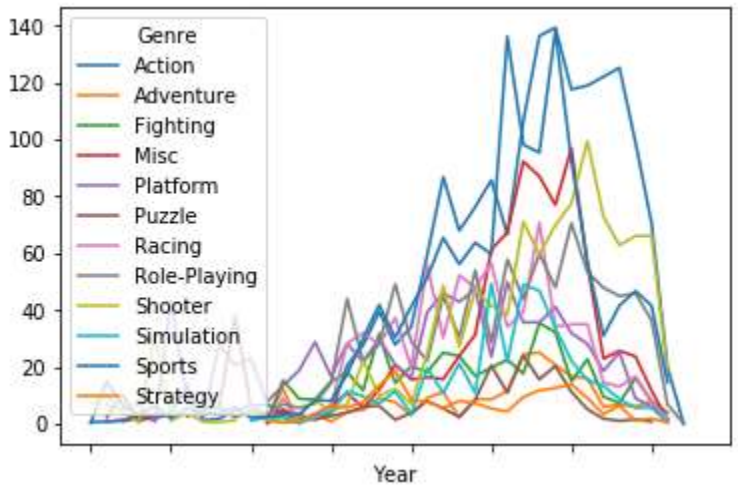


Plotting

https://pandas.pydata.org/pandas-docs/stable/visualization.html#visualization

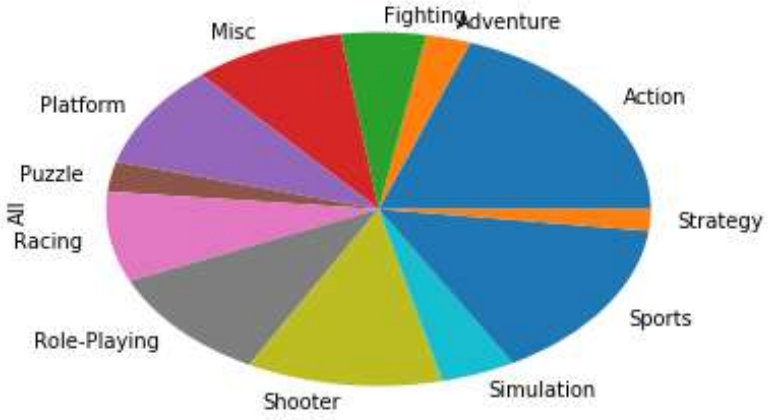
Create a pivot table of Global sales per Genre for every year and plot it

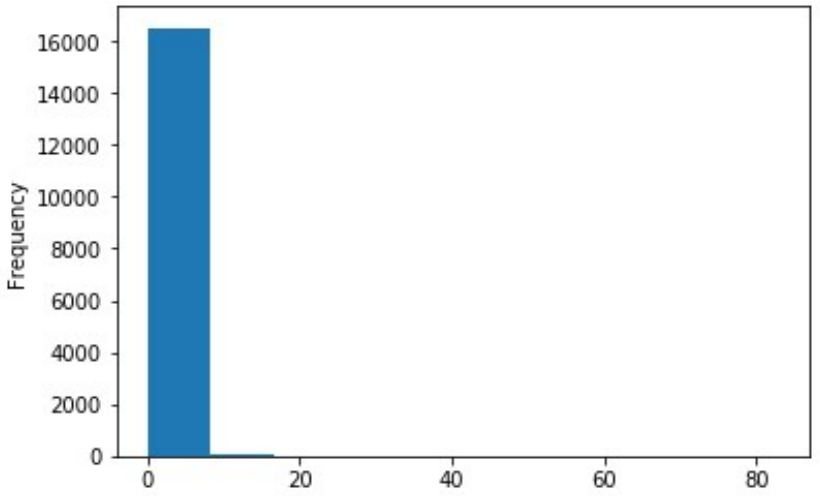
```
vgsales.pivot_table('Global_Sales', index='Year',
columns='Genre', aggfunc='sum').plot()
```



Plotting

```
pv=vgsales.pivot_table('Global_Sales', index='Year',
columns='Genre', aggfunc='sum',margins='all')

plt.figure()

pv.loc['All','Action':'Strategy'].plot.pie()

plt.figure()

vgsales['Global_Sales'].plot.hist()
```





Plotting

```
vgsales['Global_Sales'].plot.hist(logy=True)
```