1:print()

```
1.print(x,y,sep=,end=)
sep='\n' new line
sep='/' same line and sep with / symbol
```

In [4]:
```
sx='hello';sy='python';sz=10
print(sx,sy)
print(sx+sy) # + oly work for str
print(sx,sy,sz)
print(sx,sy,sz,sep='\n') # break to 3 lines

```

```
hello python
hellopython
hello python 10
hello
python
10
```

In [5]:
```
print(sx,sy,sz,sep='//',end=';')
```

```
hello//python//10;
```

In [7]:
```
#example:repeat str
print((sx,sy)*3)
```

```
('hello', 'python', 'hello', 'python', 'hello', 'python')
```

In [11]:
```
"""example 3: %s, list
1.%s =str
2.%d =digit
"""
name='john'; age=23
#print("%s is %d year old."%(age,name)) #error 順序要跟
print("%s is %d year old."%(name,age))
```

```
john is 23 year old.
```

In [18]:
```
data=("apple",'bana',55.43)
print(type(data))
print("%s are %s fruits,priced %.2f"%data) #%.2f:float to 2 decimal point
```

```
<class 'tuple'>
apple are bana fruits,priced 55.43
```

```
list=[] #mutuable
tuple=() #immutuable
dict={}
```

In [20]:
```
"""print list"""
mylist=[10,20,30] #print mylist[1]
print( ' apple pie costs %d.' %mylist[1])
```

```
 apple pie costs 20.
```

2.input()

```
<data_type>(input())
```

In [21]:
```
bonus=int(input("input the bonus: "))
salary=100+bonus
print("salary is ", salary,end=';')
```

```
input the bonus: 50
salary is  150;
```

3.beak many lines

```
slash \
blackslash /
```

In [22]:
```
"""ex1：break in element"""
a=10+20\
   +30\
   +40
print(a)
```

```
100
```

In [26]:
```
"""ex2:break in result"""
a=1000
s="it is too many lines, they are xxxxxx"
print(s,
     a,
     sep='++++++',
     end=';')
```

```
it is too many lines, they are xxxxxx++++++1000;
```

4 math symbols

```
1: divide /
2: reminder %
3:  expoent **;  a=a**2, square
4: not equal to !=
```

```
5.used in loc, &(and),(|)or , can't type and or directly
```

In [28]:
```
a=10;b=3
print(a%b) #10/3=3....1
print(a**b) #10*10*10
```

```
1
1000
```

5.string

In [32]:
```
"""ex1:simple access"""
str='python'
num=105
print(str[1])
print(str*3)
print(str,"is a ",num,)
```

```
y
pythonpythonpython
python is a  105
```

In [ ]:
```
"""DML
1.string[start:end:step] #substring
2.s.replace('a', '')/s.strip('<remove thing>')
3.String.find()=java contains(),find() doesn't find a match, it returns -1
  配合 if else
4. remove string space
a)x.replace(' ','')
b)strip(),仲有lstrip(),rstrip()
"""
```

In [39]:
```
weight='150lbs'
w1=weight[0:2]
w2=weight.replace('lbs','')

print(w1) # return 2-0=2 chars,
print(w2)

if weight.find('lbs')!=-1:
    print("find the lbs char")
else:
    print("not find")
```

```
15
150
find the lbs char
```

In [5]:
```
#remove space
str=" hello python "
str1=str.replace(' ','')
print(str1) #replace all space

str2=str.strip() #remove l/r only
print(str2)

str3=str.lstrip()
print(str3)
```

```
hellopython
hello python
hello python
```

6 list

```
list vs array
1.list contain diff datatype, array must a same datatype
  both can store non-unique elements

2. array is effient than list, + use numpy
```

```
1.[] start from 0, last is -1
2.access: list[start:end:step]
3.list.count(x) #num of list elements
4.list comprehension
#Input processed list to another list in a hight-efficienty way
link:
https://www.learncodewithmike.com/2020/01/python-comprehension.html
```

In [15]:

```
list=['A','B',3,'ee',9,'A',[1,2,3]]
print(list[-1])
print(list[1::2]) #even number
print(list.count('A'))
```

```
[1, 2, 3]
['B', 'ee', 'A']
2
```

In [18]:

```
"""non-list comprehension"""
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []
for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)

#ex2:
numbers = []
for x in range(10):
    numbers.append(x * 3)
print(numbers)
```

```
['apple', 'banana', 'mango']
[0, 3, 6, 9, 12, 15, 18, 21, 24, 27]
```

In [19]:
```
newlist2 = [x for x in fruits if "a" in x] #can add if in
print(newlist2)

#ex2:
numbers = [x * 3 for x in range(10)]
print(numbers)
```

```
['apple', 'banana', 'mango']
[0, 3, 6, 9, 12, 15, 18, 21, 24, 27]
```

In [20]:
```
"""ex3"""
numbers = [x * 3 for x in range(10) if x > 4]
print(numbers)
```

```
[15, 18, 21, 24, 27]
```

In [24]:
```
"""ex4"""
numbers = [50, 2, 12, 30, 27, 4]
result = [number * 2 for number in numbers]
print(result)

"""ex5"""
numbers2 = [50, 2, 12, 30, 27, 4]
result2 = [number for number in numbers2 if number > 10]
print(result2)

"""ex6"""
numbers3 = [50, 2, 12, 30, 27, 4]
result3 = [number if number > 10 else 0 for number in numbers3]
#put if to front is to change the list num first
print(result3)

```

```
[100, 4, 24, 60, 54, 8]
[50, 12, 30, 27]
[50, 0, 12, 30, 27, 0]
```

7: range+random

In [32]:
```
import random as rd
import numpy as np #np also has random method
```

In [ ]:
```

```

In [35]:
```
r1=rd.randrange(2, 20, 2) #return 1 random than are odd number
print(r1)

r2=rd.randint(0, 9) #randin=random integer
print(r2)
```

```
14
8
```

In [39]:
```
"""generate a list of sample
1.for loop + above 2 method
2.list comprehension
3.random.sample() function
"""
r1=[]
for i in range(0,5): #gen 5 num
    n=rd.randint(1,100) #value range 1-100
    #can change to randrange
    r1.append(n)
print(r1)

#in fact just change 1 to list comprehension
r2=[rd.randrange(1,100,2) for i in range(0,5,1) ] #1-100, odd number
print(r2)

#Generate 5 random numbers between 10 and 30
r3 = rd.sample(range(10, 30), 5)
print(r3)
```

```
[50, 24, 54, 7, 76]
[15, 49, 39, 19, 59]
[29, 14, 16, 21, 26]
```

chapter 8: 2-d for loop

In [5]:
```
"""8.1 understand nested list/array first
"""
#ex1:in theory 多，實際少用,single list
list=[[1,2,3,4],['a','b','c','d']]
print(list[0][1],list[1][2])
print()

for i in range(len(list)):
    for j in range(len(list[i])):
        print(list[i][j],sep='')
    print()
```

```
2 c

1
2
3
4

a
b
c
d
```

In [16]:
```
"""
ex2: 常見多數系matrix, one to many
process: 1a,1b,1c,1d  [0][0],[0][1],[0][2],[0][3],[0][4]
         2a,2b,2c,2d
"""
color=['red','green','blue','yello']
fruit=['apple','bal','cherry','nut']
for i in color:
    for j in fruit:
        print(i+"+"+j)
print()

for i in range(0,len(color),1):
    print(color[i],fruit[i],sep='---')
print()
```

```
red+apple
red+bal
red+cherry
red+nut
green+apple
green+bal
green+cherry
green+nut
blue+apple
blue+bal
blue+cherry
blue+nut
yello+apple
yello+bal
yello+cherry
yello+nut

red---apple
green---bal
blue---cherry
yello---nut
```

In [23]:
```
"""ex3:最實用，combine lists into one
    zip(list_x,list_y,z,....)
"""
name=['peter','john','amy']
age=[23,12,43]
score=[90,88,79]
info=list(zip(name,age,score))
info
#list() is a built-in funciton, don't use list=[1,2,4]
#del list    for error: list object is not callable
```

Out[23]:
```
[('peter', 23, 90), ('john', 12, 88), ('amy', 43, 79)]
```

chapter 9: zip()

In [24]:
```
"""ex1:dic +zip()
change list to dict(key:value)
"""

city = ["tw", "us", "hk"]
salary = [2.1, 3.8, 6.9]
d1=dict(zip(city,salary))
d1
```

Out[24]:
```
{'tw': 2.1, 'us': 3.8, 'hk': 6.9}
```

In [25]:
```
"""ex2:for loop
"""
names = ["A", "B", "C"]
level = [11, 23, 46]
ages = [45, 67, 82]
for x, y, z in zip(names, level, ages):
    print(x,y,z)
```

```
A 11 45
B 23 67
C 46 82
```

In [27]:
```
"""ex3:gen list"""
data = list(zip(range(3), 'ABCD')) #must in pair
data
```

Out[27]:
```
[(0, 'A'), (1, 'B'), (2, 'C')]
```

chapter 10: set

```
#note: set values will auto distinct
1.a - b  集合a中包含而集合b中不包含的元素
2.a | b    element in set a or b
3.a & b    insection, must in a and b
4. a ^ b   不同时包含于a和b的元素，只出現在set a or b
5.'orange' in basket  # return True/False
##################
s.add( x )   # add elements
s.update( x )
s.remove( x )
```

In [5]:
```
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(basket) #auto remove duplicate value
print("banana" in basket)
#print(basket[1]) # will error
```

```
{'pear', 'orange', 'banana', 'apple'}
True
```

In [6]:
```
"""print set() value """
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
# for each loop. print all
for i in basket:
    print(i)
```

```
pear
orange
banana
apple
```

In [10]:
```
#return specific value
for i in basket:
    if i.find('b')>=0:
        print(i)
```

```
banana
```

In [11]:
```
a = set('abracadabra')
print(a)
```

```
{'c', 'd', 'b', 'a', 'r'}
```

In [12]:
```
"""list comprehension
for loop + if
"""
list presentation

a1 = {x for x in 'abracadabra' if x not in 'abc'}
a1
```

Out[12]:
```
{'d', 'r'}
```

In [14]:
```
"""must use symbol but not text like ''
"""
a = set('abracadabra')
b = set('alacazam')
print("a-b is:", a-b) # value in a but no in b
print("a^b is:", a^b) # data can only in a or b, can not both
print("a|b is :",a|b) # union in a or b
print("a&b is : ", a&b) #intersection, must in both a and b
```

```
a-b is: {'d', 'r', 'b'}
a^b is: {'r', 'b', 'l', 'd', 'm', 'z'}
a|b is : {'l', 'c', 'd', 'b', 'm', 'z', 'a', 'r'}
a&b is :  {'a', 'c'}
```

chapter 11 :read files

```
the chapters of basic knowledge have gone. Start to 實戰。
1.data files: txt xlsx csv
2.usually we need to change the txt, csv to xlsx.
~~~~~~~
3.csv=microsoft excel comma  seperated values file
  xlsx=microsoft excel worksheet
  txt=text document
```

In [32]:
```
import pandas as pd
# inplace =True  meas change to exiting df, default is False
```

In [15]:
```
"""use pandas module's methods, but not function,
   so it is pd.to_read(var), but to_read(var)
#######
1.
given path:C:\Users\fengs\Desktop\py_project\csv_file
not use \, but /
2.or add (r'')
"""
#df_p=pd.read_csv('C:\Users\fengs\Desktop\py_project\csv_file\pokemon_data.csv') #unicode error
#df_p=pd.read_csv('C:/Users/fengs/Desktop/py_project/Upy_project/csv_file/pokemon_data.csv')
```

```
  File "C:\Users\fengs\AppData\Local\Temp/ipykernel_3956/338393444.py", line 8
    """
       ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 117-118: truncated \UXXXXXXXX escape
```

chapter 11.1 foundational info of DF

In [3]:
```
df_csv=pd.read_csv(r'C:\Users\fengs\Desktop\py_project\csv_file\pokemon_data.csv')
df_csv.head()
```

Out[3]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False

In [63]:
```
#find df's row &columns
#(number of rows, number of columns).
print(df_csv.shape)
print(df_csv.shape[0],df_csv.shape[1])
print(df_csv.count(0))
```

```
(800, 12)
800 12
#             800
Name          800
Type 1        800
Type 2        414
HP            800
Attack        800
Defense       800
Sp. Atk       800
Sp. Def       800
Speed         800
Generation    800
Legendary     800
dtype: int64
```

In [19]:
```
df_csv.columns  #it is an attribute but not a function or method
```

Out[19]:
```
Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk',
       'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

In [21]:
```
#df.info()=show table DDL
df_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   #           800 non-null    int64 
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   HP          800 non-null    int64 
 5   Attack      800 non-null    int64 
 6   Defense     800 non-null    int64 
 7   Sp. Atk     800 non-null    int64 
 8   Sp. Def     800 non-null    int64 
 9   Speed       800 non-null    int64 
 10  Generation  800 non-null    int64 
 11  Legendary   800 non-null    bool  
dtypes: bool(1), int64(8), object(3)
memory usage: 69.7+ KB
```

In [22]:
```
#df_describe()=show stat data
df_csv.describe()
```

Out[22]:

	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.00000
mean	362.813750	69.258750	79.001250	73.842500	72.820000	71.902500	68.277500	3.32375
std	208.343798	25.534669	32.457366	31.183501	32.722294	27.828916	29.060474	1.66129
min	1.000000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000	1.00000
25%	184.750000	50.000000	55.000000	50.000000	49.750000	50.000000	45.000000	2.00000
50%	364.500000	65.000000	75.000000	70.000000	65.000000	70.000000	65.000000	3.00000
75%	539.250000	80.000000	100.000000	90.000000	95.000000	90.000000	90.000000	5.00000
max	721.000000	255.000000	190.000000	230.000000	194.000000	230.000000	180.000000	6.00000

In [20]:
```
print(df_csv.head(2))
print(df_csv.tail(2))
```

```
   #       Name Type 1  Type 2  HP  Attack  Defense  Sp. Atk  Sp. Def  Speed  \
0  1  Bulbasaur  Grass  Poison  45      49       49       65       65     45
1  2    Ivysaur  Grass  Poison  60      62       63       80       80     60

   Generation  Legendary
0           1      False
1           1      False
       #               Name   Type 1 Type 2  HP  Attack  Defense  Sp. Atk  \
798  720  HoopaHoopa Unbound  Psychic   Dark  80     160       60      170
799  721           Volcanion     Fire  Water  80     110      120      130

     Sp. Def  Speed  Generation  Legendary
798      130     80           6       True
799       90     70           6       True
```

Chapter 11.2 query DF

In [29]:
```
"""select cols,
1.df[] only for single col
2.df[[x,y]] for multi-col, as it only return one arg
"""
print(df_csv[['#','Name','Type 1']].head(2))
print(df_csv[['#','Name','HP']].tail(2))
```

```
   #       Name Type 1
0  1  Bulbasaur  Grass
1  2    Ivysaur  Grass
       #                Name  HP
798  720  HoopaHoopa Unbound  80
799  721           Volcanion  80
```

In [37]:
```
#between 2 values
df_csv.loc[df_csv['#'].between(99, 101)]
```

Out[37]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
107	99	Kingler	Water	NaN	55	130	115	50	50	75	1	False
108	100	Voltorb	Electric	NaN	40	30	50	55	55	100	1	False
109	101	Electrode	Electric	NaN	60	50	70	80	80	140	1	False

In [4]:
```
"""loc[row,cols]
loc[].range(start,end,step)
most practical function of DF
2.loc=location
iloc=index location
"""
r1=df_csv.loc[range(0,len(df_csv),2),['#','Name','Sp. Atk']].head(3) #even 雙，odd 單
r1
```

Out[4]:

	#	Name	Sp. Atk
0	1	Bulbasaur	65
2	3	Venusaur	100
4	4	Charmander	60

```
ref:
df3 = df.loc[(df['Date'] > 'Feb 06, 2019') & (df['Open'] > 62), ['Date', 'Open']]
```

In [5]:
```
"""loc + where condition
df.loc[(c1)&(c2)&(c3)....,['col1','col2']]
1.each condition needs to be bracketed
2.use []=='string', but not =, '=' means assign to

"""
r2=df_csv.loc[(df_csv['#'].between(3,30)) & (df_csv['Type 1']=='Fire'),['#','Name','Type 1']]
r2.head(3)
```

Out[5]:

	#	Name	Type 1
4	4	Charmander	Fire
5	5	Charmeleon	Fire
6	6	Charizard	Fire

In [6]:
```
r3=df_csv.loc[((df_csv['#'].between(3,30)) & (df_csv['Type 1']=='Fire'))|(df_csv['Speed']>150),
              ['#','Name','Type 1','Speed']]
r3
```

Out[6]:

	#	Name	Type 1	Speed
4	4	Charmander	Fire	65
5	5	Charmeleon	Fire	80
6	6	Charizard	Fire	100
7	6	CharizardMega Charizard X	Fire	100
8	6	CharizardMega Charizard Y	Fire	100
315	291	Ninjask	Bug	160
431	386	DeoxysSpeed Forme	Psychic	180

In [7]:
```
"""more advaced query
1.df[].str.contain('A|B') # like
2.~df[].str.contain('A|B')
3.join
"""
r4=df_csv.loc[(df_csv['#'].between(20,120)) & (df_csv['Attack']>=100) & (df_csv['Name'].str.contains('s')),
              ['Name','#','Attack','Type 1']]
r4
```

Out[7]:

	Name	#	Attack	Type 1
33	Sandslash	28	100	Ground
84	Rapidash	78	100	Fire
124	KangaskhanMega Kangaskhan	115	125	Normal

In [ ]:
```
#add new columns base on conditon
df['Is_eligible'] = np.where(df['Age'] >= 18, True, False)
```

chapter 11.3 sorting DF

```
"""sorting
DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort',
                      na_position='last', ignore_index=False, key=None)[source]
"""
ref:df.sort_values(by=['Brand'], inplace=True, ascending=False)
```

In [ ]:
```
#example: sort many cols
df.sort(['a', 'b'], ascending=[True, False]) #or use 0,1
```

In [8]:
```
r5=df_csv.loc[df_csv['#'].between(50,100),['#','Name','Type 1','HP','Attack']]
r5.sort_values(by=['#'],inplace=True,ascending=False)
r5.head(5)
```

Out[8]:

	#	Name	Type 1	HP	Attack
108	100	Voltorb	Electric	40	30
107	99	Kingler	Water	55	130
106	98	Krabby	Water	30	105
105	97	Hypno	Psychic	85	73
104	96	Drowzee	Psychic	60	48

In [9]:
```
sales_data = pd.DataFrame({"name":["William","Emma","Sofia","Markus","Edward","Thomas","Ethan","Olivia","Arun","Anik
,"region":["East","North","East","South","West","West","South","West","West","East","South"]
,"sales":[50000,52000,90000,34000,42000,72000,49000,55000,67000,65000,67000]
,"expenses":[42000,43000,50000,44000,38000,39000,42000,60000,39000,44000,45000]})
sales_data
```

Out[9]:

	name	region	sales	expenses
0	William	East	50000	42000
1	Emma	North	52000	43000
2	Sofia	East	90000	50000
3	Markus	South	34000	44000
4	Edward	West	42000	38000
5	Thomas	West	72000	39000
6	Ethan	South	49000	42000
7	Olivia	West	55000	60000
8	Arun	West	67000	39000
9	Anika	East	65000	44000
10	Paulo	South	67000	45000

In [10]:
```
sales_data.sort_values(by=['sales','expenses'],inplace=True,ascending=[False,True])
sales_data
```

Out[10]:

	name	region	sales	expenses
2	Sofia	East	90000	50000
5	Thomas	West	72000	39000
8	Arun	West	67000	39000
10	Paulo	South	67000	45000
9	Anika	East	65000	44000
7	Olivia	West	55000	60000
1	Emma	North	52000	43000
0	William	East	50000	42000
6	Ethan	South	49000	42000
4	Edward	West	42000	38000
3	Markus	South	34000	44000

11.5 custom ordering

In [12]:
```
#try to sorting by type 1
r6=df_csv.loc[df_csv['#'].between(25,100),['Name','#','HP','Type 1']]
r6.head()
```

Out[12]:

	Name	#	HP	Type 1
30	Pikachu	25	35	Electric
31	Raichu	26	60	Electric
32	Sandshrew	27	50	Ground
33	Sandslash	28	75	Ground
34	Nidoran (Female)	29	55	Poison

In [14]:
```
#find distinct value of a column,
#sort by this custom cols value
df_csv['Type 1'].unique()
```

Out[14]:
```
array(['Grass', 'Fire', 'Water', 'Bug', 'Normal', 'Poison', 'Electric',
       'Ground', 'Fairy', 'Fighting', 'Psychic', 'Rock', 'Ghost', 'Ice',
       'Dragon', 'Dark', 'Steel', 'Flying'], dtype=object)
```

```
pandas not equal to list of values
df[~df['column_name'].isin(['value_1', 'value_2'])]

#failed
r6['Type 1']!=('Water','Normal','Grass')
```

In [58]:

```
"""1.map a new cols
   2.Use CategoricalDtype Cast data to an ordered category type
   3.As commented, in newer pandas,
     Series has a replace method to do this more elegantly:
     s = df['m'].replace({'March':0, 'April':1, 'Dec':3})

link: df.map() vs df.apply()
https://zhuanlan.zhihu.com/p/100064394
https://chowdera.com/2020/11/20201106205416849i.html
"""
main=['Water','Normal','Grass']
#link:https://www.statology.org/case-statement-pandas/
r6['order']=np.where(r6['Type 1']=='Water',1,
                     np.where(r6['Type 1']=='Normal',2,
                     np.where(r6['Type 1']=='Grass',3,
                     4    #mean not in  3 above
                                )))
r6
```

Out[58]:

	Name	#	HP	Type 1	order
30	Pikachu	25	35	Electric	4
31	Raichu	26	60	Electric	4
32	Sandshrew	27	50	Ground	4
33	Sandslash	28	75	Ground	4
34	Nidoran (Female)	29	55	Poison	4
...	...	...	...	...	...
104	Drowzee	96	60	Psychic	4
105	Hypno	97	85	Psychic	4
106	Krabby	98	30	Water	1
107	Kingler	99	55	Water	1
108	Voltorb	100	40	Electric	4

79 rows × 5 columns

In [59]:
```
r6.sort_values(by=['order','Name'],inplace=True,ascending=[True,True])
r6
```

Out[59]:

	Name	#	HP	Type 1	order
98	Cloyster	91	50	Water	1
94	Dewgong	87	90	Water	1
60	Golduck	55	80	Water	1
107	Kingler	99	55	Water	1
106	Krabby	98	30	Water	1
...	...	...	...	...	...
54	Venomoth	49	70	Bug	4
53	Venonat	48	60	Bug	4
108	Voltorb	100	40	Electric	4
42	Vulpix	37	38	Fire	4
46	Zubat	41	40	Poison	4

79 rows × 5 columns

In [69]:
```
####show all columns execept one
#r6.loc[:,r6.columns!='order']

#not show some columbs
#df.loc[:, ~df.columns.isin(['col1', 'col2'])]
r6.loc[:,~r6.columns.isin(['order','HP'])].head()
```

Out[69]:

	Name	#	Type 1
98	Cloyster	91	Water
94	Dewgong	87	Water
60	Golduck	55	Water
107	Kingler	99	Water
106	Krabby	98	Water

11.5.2 :create a ordering function

```
if row A == B: 0

if rowA > B: 1

if row A < B: -1
```

In [68]:
```
#create lists to DF
df=pd.DataFrame({
    'A':[2,3,1],
    'B':[2,1,3]
   })
df
```

Out[68]:

	A	B
0	2	2
1	3	1
2	1	3

In [53]:
```
def f(row):
    if row['A'] == row['B']:
        val = 0
    elif row['A'] > row['B']:
        val = 1
    else:
        val = -1
    return val
df['C'] = df.apply(f, axis=1) #axis=1 AS diff ROW compare
df
```

Out[53]:

	A	B	C
0	2	2	0
1	3	1	1
2	1	3	-1

In [23]:
```
import numpy as np
```

In [43]:
```
r7=df_csv.loc[(df_csv['#'].between(1,10)) & (~df_csv['Type 1'].isin(['Grass','Water']))]
r7
```

Out[43]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
5	5	Charmeleon	Fire	NaN	58	64	58	80	65	80	1	False
6	6	Charizard	Fire	Flying	78	84	78	109	85	100	1	False
7	6	CharizardMega Charizard X	Fire	Dragon	78	130	111	130	85	100	1	False
8	6	CharizardMega Charizard Y	Fire	Flying	78	104	78	159	115	100	1	False
13	10	Caterpie	Bug	NaN	45	30	35	20	20	45	1	False

chapter 13.numpy

chapter 13.1： background

```
numpy use n-dimension array to replace python list,MAIN concepts include{ ndim,shape,dtype}
1234 :shape(4,); axis=0

1234
5678
shape(2,4); axis has 0,1 0:y-aixs(cols) 1:x-aix(rows)
```

```
Q1:why use numpy???
In Python we have lists that serve the purpose of arrays, but they are slow to process.
NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
Arrays are very frequently used in data science, where speed and resources are very important.

Q2:Why is NumPy Faster Than Lists?
```

```
NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.This behavior is called locality of reference in computer science.
This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

```

Chapter 13.2 basic syntax

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import sys
```

In [ ]:
```
numpy.array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)

link:
https://thispointer.com/python-numpy-create-a-ndarray-from-list-tuple-or-list-of-lists-using-numpy-array/
```

In [5]:
```
#1.create list/tuple array
np1=np.array([1,2,3]) #list[]
np2=np.array((4,5,6)) #tupe(), immutuble,one datatype
print(np1)
print(np2)
print(type(np1),type(np2)) # bot convert to ndarray
```

```
[1 2 3]
[4 5 6]
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

In [8]:
```
a1=np.array([1,2,3],dtype=int)
a2=np.array([1,2,3],dtype=float)
print(a1)
print(a2)
```

```
[1 2 3]
[1. 2. 3.]
```

In [7]:
```
#2.arrang(start,stop,step,dtype=None)
"""similar list range,range is built-in python class
   arange is a function of third-party library(Numpy)
"""
print(np.arange(1, 8, 3))
print( np.arange(1, 10.1, 3)) #get 4f, include 10.1
```

```
[1 4 7]
[ 1.  4.  7. 10.]
```

In [11]:
```
"""
numpy.linspace(start, stop, num=50,
               endpoint=True, retstep=False,
               dtype=None, axis=0)
-> start  : [optional] start of interval range. By default start = 0
-> stop   : end of interval range
-> restep : If True, return (samples, step). By deflut restep = False
-> num    : [int, optional] No. of samples to generate
-> dtype  : type of output array
"""
x = np.linspace(0, 2, 5) #5 is return 5 num
print(x) #default return float

y=np.linspace(0,8,2) #return 2 nums
print(y)
```

```
[0.  0.5 1.  1.5 2. ]
[0. 8.]
```

In [23]:
```
#3.create nd-array
"""
arr1 = np.array([1, 2, 3, 4, 5]) #1d
arr2 = np.array([[1, 2, 3], [4, 5, 6]]) #2d

arr3 = np.array([[[1, 2, 3], [4, 5, 6]],
                 [[1, 2, 3], [4, 5, 6]]
                ])
"""
print('arr3 is :',arr3)
print()

#error:if [1,2,3,4]
#3*4 matrix
arr4=np.array([[1,2,4,3],['d','e','f','G'],['ten','ele','twe','13th']])
print('arr4 is:',arr4)
print("dimentsion: ",arr4.ndim) #as not more than 2 layer nested array
print("matrix/(row: col) is :",arr4.shape)
print("total number of ele is:",arr4.size)
```

```
arr3 is : [[[1 2 3]
  [4 5 6]]

 [[1 2 3]
  [4 5 6]]]

arr4 is: [['1' '2' '4' '3']
 ['d' 'e' 'f' 'G']
 ['ten' 'ele' 'twe' '13th']]
dimentsion:  2
matrix/(row: col) is : (3, 4)
total number of ele is: 12
```

In [30]:
```
"""4.change to multi-list to array
numpy.reshape(a, newshape, order='C')
order:'C','K','A'
"""
a1=np.arange(8) #(sart,end,step)
print(a1)
a1=np.arange(1,17).reshape(4,4) #(row,col)
print(a1)
```

```
[0 1 2 3 4 5 6 7]
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

In [37]:
```
#5 access array values
#a1[row,col]=[<start:end:step:>,<:::>] start from 0
print(a1[1,1:3]) #1st row,1,2 ele
print(a1[1:3,2]) # 1&2 row, 2nd col
print(a1[::2,::2])
print(a1[:,2])
print(a1[1,:])
```

```
[6 7]
[ 7 11]
[[ 1  3]
 [ 9 11]]
[ 3  7 11 15]
[5 6 7 8]
```

chapter 13.3 initialize diff types of arrays

In [43]:
```
#1.all 1s/0s matrix
#numpy.ones(shape, dtype = None, order = 'C')
#shape =(row,col)
print(np.zeros(3,))
print(np.zeros((2,3)))
print(np.ones((3,3)))

```

```
[0. 0. 0.]
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

In [44]:
```
#2.any other number martrix
#numpy.full(shape, fill_value, dtype = None, order = ‘C’)
np.full((2,3),12)
```

Out[44]:
```
array([[12, 12, 12],
       [12, 12, 12]])
```

In [45]:
```
#full_like, change copy a matrix shape
arr2 = np.arange(16, dtype=float).reshape(4, 4)
print("\n\narr2 before full_like : \n", arr2)

# using full_like
print("\narr2 after full_like : \n", np.full_like(arr2, -3))
```

```
arr2 before full_like :
 [[ 0.  1.  2.  3.]
 [ 4.  5.  6.  7.]
 [ 8.  9. 10. 11.]
 [12. 13. 14. 15.]]

arr2 after full_like :
 [[-3. -3. -3. -3.]
 [-3. -3. -3. -3.]
 [-3. -3. -3. -3.]
 [-3. -3. -3. -3.]]
```

chapter 14 numpy generate randon data

In [47]:
```
"""
1.random decimal number
np.random.rand(d0,d2,d3) :d means dimension
#value[0,1]
"""
print(np.random.rand(2))
print(np.random.rand(2,3))#(row,cols)
```

```
[0.18144125 0.18824898]
[[0.55449679 0.15667761 0.77931262]
 [0.7523837  0.38728601 0.72929654]]
```

In [51]:
```
"""
2.np.random.randint(min,max,size,dtype)
random.randint(low, high=None, size=None, dtype=int)
#size(row,col) d-array
"""
print("1st arr: \n",
     np.random.randint(7,size=(4))) #0-7,7 exclude
print("2nd arr: \n",
     np.random.randint(low=2,high=8,size=(2,3))
     )
```

```
1st arr:
 [6 4 0 2]
2nd arr:
 [[3 3 2]
 [5 2 3]]
```

In [58]:
```
"""most practical use to gen list data
3.choice(a, size=None, replace=True, p=None)
NOTE:replace=False :not repeat value
a=array
"""
print("1st arr: \n",
      np.random.choice(range(2,10),size=(5),replace=False)
     )
print("2nd arr: \n",
      np.random.choice(range(3,50),size=(2,4),replace=False)
     )
```

```
1st arr:
 [7 9 8 3 6]
2nd arr:
 [[13 33 24 40]
 [46  8 16 30]]
```

real case on plot

In [67]:
```
boolean=[True,False]
gender=["男","女"]
color=["white","black","yellow"]
data=pd.DataFrame({
    "height":np.random.randint(150,190,100),
    "weight":np.random.randint(40,90,100),
    "smoker":[boolean[x] for x in np.random.randint(0,2,100)],
    "gender":[gender[x] for x in np.random.randint(0,2,100)],
    "age":np.random.randint(15,90,100),
    "color":[color[x] for x in np.random.randint(0,len(color),100) ]
}
)
```

In [ ]:
```
"""gender列的男替换为1，女替换为0，
Series.map() vs apply()
"""
data["gender"] = data["gender"].map({"男":1, "女":0})
data["gender"] = data["gender"].map(gender_map)
#
data["age"] = data["age"].apply(apply_age,args=(-3,))
```

chapter 14.2 numpy math

given random arrays

In [59]:
```
a=np.random.choice(range(2,10),size=(5),replace=False)
b=a.copy()
b
```

Out[59]:
```
array([4, 2, 7, 6, 9])
```

In [67]:
```
#1given
#random draw [1,10] to build 2*3
a=np.random.choice(range(1,10),size=(2,3),replace=False)

"""
#[2,13] all element to build 2*3
#error:b=np.arange(2,13).reshape(2,3)
#need to arange=2*3=6
"""
b=np.arange(5,11).reshape(2,3) #range total must=6

print("a array is:\n",a)
print("b array is:\n",b)
```

```
a array is:
 [[8 4 3]
 [7 2 5]]
b array is:
 [[ 5  6  7]
 [ 8  9 10]]
```

In [73]:
```
#1 add
c= a+1
print("c is :\n",c)
print()

#expoent 次方
d=a**2
print("d is : \n",d)
print()

#extract specific element
e=(a[0,:]+1) # fist row,all cols
print("e is :\n",e)
```

```
c is :
 [[9 5 4]
 [8 3 6]]

d is :
 [[64 16  9]
 [49  4 25]]

e is :
 [9 5 4]
```

In [77]:
```
"""Agg of arrays
"""
#ex1:
print(a)
print()
f=np.sum(a)
print(f)
print()

#ex2:expod
#所有相乘
g=np.prod(a)
print("prod is :",g)

#min/max
#(0=y)min:each col
#(1=x)max：each row
h=[np.min(a,axis=0),np.max(a,axis=1)]
print("h is \n",h)

"""others stat
1.cumsum 累計加
2.cumprod 累計乘
3.ptp :diff=max-min
4.percentitle(arr,75) 第75% value
"""
```

```
[[8 4 3]
 [7 2 5]]

29

prod is : 6720
h is
 [array([7, 2, 3]), array([8, 7])]
```

14.3 sorting array

In [81]:
```
#randint has not replace agr
#elements may repeat
x=np.random.randint(2,15,size=(3,5))
x
```

Out[81]:
```
array([[ 3, 13, 11,  9,  9],
       [ 7,  6, 10,  6,  4],
       [ 7,  3, 11,  9,  3]])
```

In [78]:
```

```

Out[78]:
```
array([[10,  4, 13,  9],
       [ 6, 11,  3, 14],
       [ 2, 12,  8,  7]])
```

In [88]:
```
#replace=False(not repeat)
a=np.random.choice(range(2,15),size=(3,4),replace=False)
print("given arrays is:\n",a)
print()

#sprt by row
b=np.sort(a,axis=1)
print("sort by row ascending:\n",b)
print()

b1=-np.sort(-a,axis=1)
print("sort by row descending:\n",b1)
print()

c=np.sort(a,axis=0)
print("sort by cols:\n", c)
```

```
given arrays is:
 [[ 8  7 12  2]
 [14  5 11  9]
 [ 6 13 10  3]]

sort by row ascending:
 [[ 2  7  8 12]
 [ 5  9 11 14]
 [ 3  6 10 13]]

sort by row descending:
 [[12  8  7  2]
 [14 11  9  5]
 [13 10  6  3]]

sort by cols:
 [[ 6  5 10  2]
 [ 8  7 11  3]
 [14 13 12  9]]
```

In [ ]:
```

```

In [ ]:
```
1
```

chapter 15. outport DF to excel

In [ ]:
```
1
```

Part 2:

chapter 16: OS module

In [1]:
```
1 """get the jupyter files info:
2 eg:path ,file names
3 """
4 import os
```

In [3]:
```
1 #cwd: current working directly
2 os.getcwd()
```

Out[3]:
```
'C:\\Users\\fengs'
```

In [7]:
```
"""
default:list cwd files
dir=directory
"""
#os.listdir()
os.listdir('C:/Users/fengs/Desktop/pd_real')
```

Out[7]:
```
['data.xlsx',
 'forget',
 'gd',
 'gd_2',
 'gd_3',
 'GeeksforGeeks',
 'mathplot',
 'mathplot.pdf',
 'matplotlib_tutorial-master.zip',
 'Misc',
 'py_loc.pdf',
 'py_road.pdf',
 'README.md',
 'SalesAnalysis']
```

In [9]:
```
import glob
# All files and directories ending with .pdf and that don't begin with a dot:
print(glob.glob("C:/Users/fengs/Desktop/pd_real/*.pdf"))
```

```
['C:/Users/fengs/Desktop/pd_real\\mathplot.pdf', 'C:/Users/fengs/Desktop/pd_real\\py_loc.pdf', 'C:/Users/fengs/Desktop/pd_real\\py_road.pdf']
```

In [11]:
```
# All files and directories ending with .pdf with depth of 2 folders, ignoring names beginning with a dot:
print(glob.glob("C:/Users/fengs/Desktop/pd_real/*.pdf"))
```

```
['C:/Users/fengs/Desktop/pd_real\\mathplot.pdf', 'C:/Users/fengs/Desktop/pd_real\\py_loc.pdf', 'C:/Users/fengs/Desktop/pd_real\\py_road.pdf']
```

In [16]:
```
#print paths of files as well
pdf_files = []
for file in glob.glob("C:/Users/fengs/Desktop/pd_real/*.pdf"):
    #pdf_files.append(file)
     print(file)
```

```
C:/Users/fengs/Desktop/pd_real\mathplot.pdf
C:/Users/fengs/Desktop/pd_real\py_loc.pdf
C:/Users/fengs/Desktop/pd_real\py_road.pdf
```

chapter 16.2 formal ways

In [17]:
```
path = "C:/Users/fengs/Desktop/pd_real"
dir_list = os.listdir(path)

print("Files and directories in '", path, "' :")

# prints all files
print(dir_list)
```

```
Files and directories in ' C:/Users/fengs/Desktop/pd_real ' :
['data.xlsx', 'forget', 'gd', 'gd_2', 'gd_3', 'GeeksforGeeks', 'mathplot', 'mathplot.pdf', 'matplotlib_tutorial-master.
zip', 'Misc', 'py_loc.pdf', 'py_road.pdf', 'README.md', 'SalesAnalysis']
```

In [19]:
```
#print files names only
path='C:/Users/fengs/Desktop/pd_real/mathplot'
for x in os.listdir(path):
    if x.endswith(".csv"):
        # Prints only csv file present in My Folder
        print(x)
```

```
fifa_data.csv
gas_prices.csv
```

chapter 17:pandas plot

```
there are 3 commonly used plots:
1.pandas plot :simple,easy to use
2.mathplotlib plot: complicated, but more flexible
3.seaborn plot :optional to know
```

```
Although they may appear similar, these modules have unique purposes and functionalities.

The Pandas module is used for working with tabular data. It allows us to work with data in table form,
such as in CSV or SQL database formats. We can also create tables of our own,
and edit or add columns or rows to tables.
Pandas provides us with some powerful objects like DataFrames and Series which
are very useful for working with and analyzing data.

The Numpy module is mainly used for working with numerical data.
It provides us with a powerful object known as an Array.
With Arrays, we can perform mathematical operations on multiple values
in the Arrays at the same time, and also perform operations between different Arrays,
similar to matrix operations.

Last, but not least, the Matplotlib module is used for data visualization.
It provides functionality for us to draw charts and graphs,
so that we can better understand and present the data visually.

These modules have different purposes and functionality they excel at,
and together they allow us to analyze,
manipulate and visualize data in very useful ways.
```

In [21]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

```
df.plot(kind = '<graph_type>', x = 'x_tilte', y = 'y_axis',color='',ls='',marker='<o/.>',xlim,xticks)
1.kind='line','bar','hist','bar','barh(橫條)','scatter','pie'
2.x/y=names of axis
3.ls=linestyle,<->實線,<-->虛線,<:>點線,<-*>虛點線
4.lw=lineweigth,
5.figsize=[10,5],[8,4]: x,y axis length ratio

  ~~~~
```

```
df.plot is friendly with tabular data,
so CAN combine loc
eg:one row one line
df.iloc[1].plot()
df.iloc[2].plot()
```

example_1: school marks

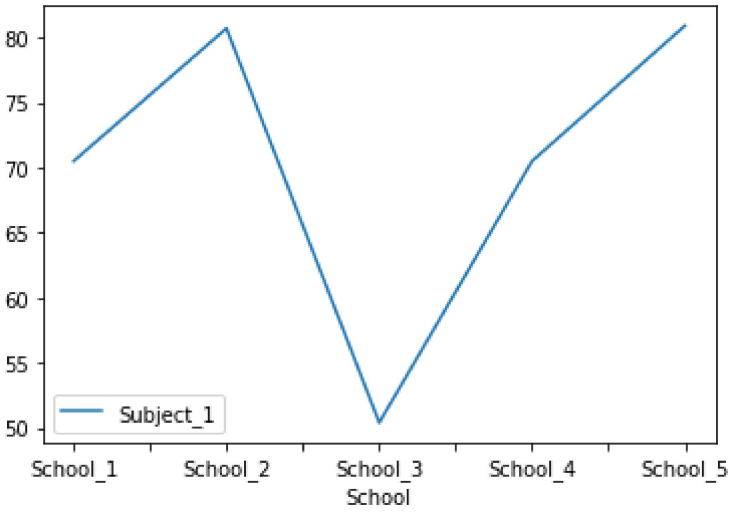
In [31]:

```
"""ref:example
"""
df = pd.DataFrame({
    'Subject_1': [70.5, 80.7, 50.4, 70.5, 80.9],
    'Subject_2': [40.24, 50.9, 70.6, 80.1, 50.9],
    'Subject_3': [30, 50.5, 70.8, 90.88, 30],
    'School': ['School_1', 'School_2', 'School_3', 'School_4', 'School_5'],
    'Date': ['2021/06/01', '2021/06/02', '2021/06/03', '2021/06/04', '2021/06/05']
})
df
```

Out[31]:

	Subject_1	Subject_2	Subject_3	School	Date
0	70.5	40.24	30.00	School_1	2021/06/01
1	80.7	50.90	50.50	School_2	2021/06/02
2	50.4	70.60	70.80	School_3	2021/06/03
3	70.5	80.10	90.88	School_4	2021/06/04
4	80.9	50.90	30.00	School_5	2021/06/05

In [32]:
```
"""
1.line plot with x and y values defined
 抽其中一個object 出來，自己define x,y axis values
 the x,y 100% same as the column name, row value
 自己識食
"""
#compare schools's sub_1 values
df.plot(x='School', y='Subject_1');
```



real excel case for gas_price

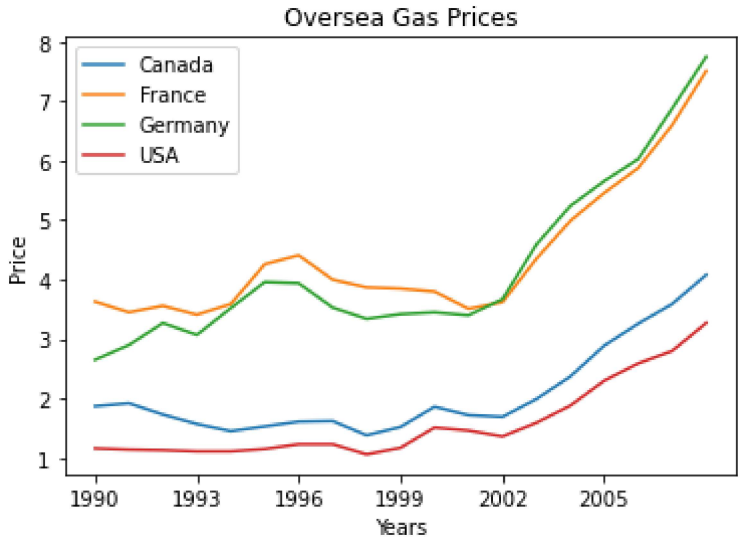
In [25]:
```
"""
PATH='/ FOR folder, \file.xxx'
"""
df_gas=pd.read_csv('C:/Users/fengs/Desktop/pd_real/mathplot\gas_prices.csv')
df_gas.head()
```

Out[25]:

	Year	Australia	Canada	France	Germany	Italy	Japan	Mexico	South Korea	UK	USA
0	1990	NaN	1.87	3.63	2.65	4.59	3.16	1.00	2.05	2.82	1.16
1	1991	1.96	1.92	3.45	2.90	4.50	3.46	1.30	2.49	3.01	1.14
2	1992	1.89	1.73	3.56	3.27	4.53	3.58	1.50	2.65	3.06	1.13
3	1993	1.73	1.57	3.41	3.07	3.68	4.16	1.56	2.88	2.84	1.11
4	1994	1.84	1.45	3.59	3.52	3.70	4.36	1.48	2.87	2.99	1.11

In [83]:
```
#try to use pandas.plot only, no matplotlib
#CANNOT g1=df.plot g2=df.plot, it will come out 2 graphs
#link:https://realpython.com/pandas-plot-python/
df_gas.plot(x='Year',y=['Canada','France','Germany','USA'], #must know:to draw multi-lines
            xticks=range(df_gas['Year'].min(),df_gas['Year'].max(),3),
            xlabel='Years',ylabel='Price',title='Oversea Gas Prices'
           )

```

Out[83]: <AxesSubplot:title={'center':'Oversea Gas Prices'}, xlabel='Years', ylabel='Price'>

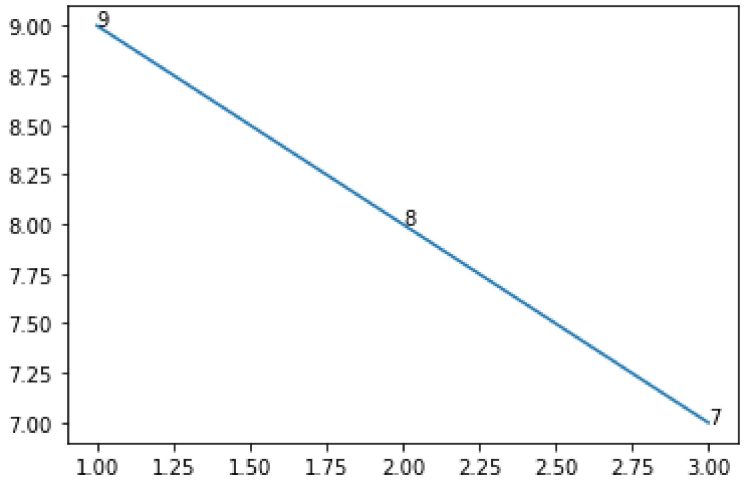


In [38]:
```
"""prepare the x,y axis values"""
print(df_gas['Year'].min(),df_gas['Year'].max())
```

```
1990 2008
```

demo show point values

In [52]:
```
import matplotlib.pyplot as plt

x=[1,2,3]
y=[9,8,7]

plt.plot(x,y)
for a,b in zip(x, y):
    plt.text(a, b, str(b))
plt.show()
```



chapter 18 :Mathplotlib graphics

```
syntax:
 plt.plot(x,y,color='r,g,b',marker=<o/.>,ls=<-/--/:>,label='')

1.figsize is size on container
   # Creating a new figure with width = 5 inches
   # and height = 4 inches
   fig = plt.figure(figsize =(5, 4))
```

```
2.subplot=container分成幾多個小正方形，用來在一個containers
          draw N graphics,如果1個就不用理，default(1,1,1)
3.label=line name
4.ls=line style
```

In [58]:
```
import matplotlib.pyplot as plt
df_gas=pd.read_csv('C:/Users/fengs/Desktop/pd_real/mathplot\gas_prices.csv')
```

In [82]:
```
df_gas.head(2)
```

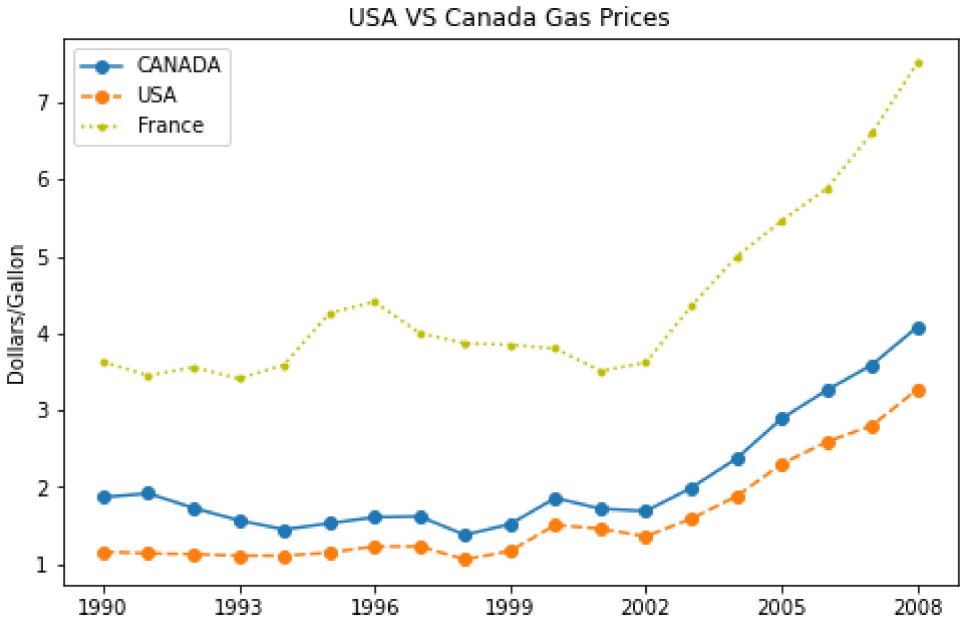
Out[82]:

	Year	Australia	Canada	France	Germany	Italy	Japan	Mexico	South Korea	UK	USA
0	1990	NaN	1.87	3.63	2.65	4.59	3.16	1.0	2.05	2.82	1.16
1	1991	1.96	1.92	3.45	2.90	4.50	3.46	1.3	2.49	3.01	1.14

18.1 :EX1: Multi-line

In [85]:

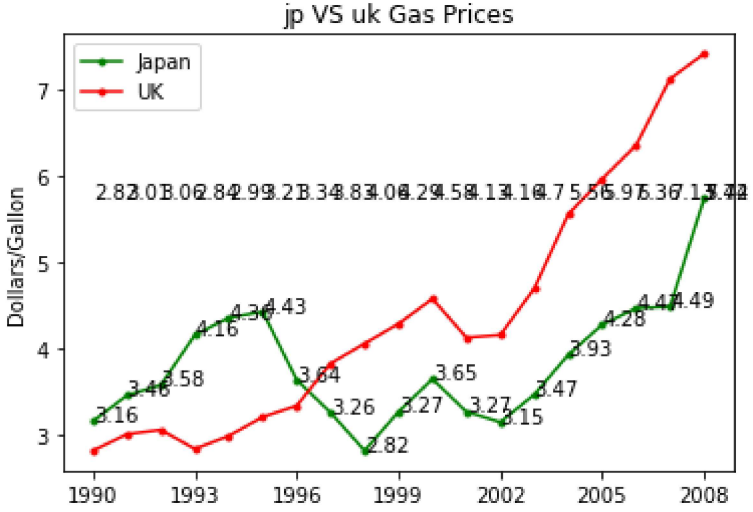
```
#1.commonly shared by lines
plt.figure(figsize=(8,5)) #x=8 inch ,y =5 inch
plt.title('USA VS Canada Gas Prices')
plt.ylabel('Dollars/Gallon')

#x-value intervals
plt.xticks(df_gas['Year'][::3])

#2.no :x=[],y=[] arg, directly x,y varible
plt.plot(df_gas['Year'],df_gas['Canada'],ls='-',marker='o' ,
         label='CANADA')
plt.plot(df_gas['Year'],df_gas['USA'],ls='--',marker='o' ,
         label='USA')
#shortform yellow line, dot mark,: 虛線
plt.plot(df_gas['Year'],df_gas['France'],'y.:',
         label='France')

#3.legend=傳說/圖像說明
plt.legend() #to display the line label
plt.show()

"""note
1.can y[['1','2'....]],but not sugg, as mark label problem
2.type change is not (kind='x'),but pltbar()/plthist()/
"""
```



Out[85]: "note\n1.can y[['1','2'....]],but not sugg, as mark label problem\n2.type change is not (kind='x'),but pltbar()/plthist()/\n"

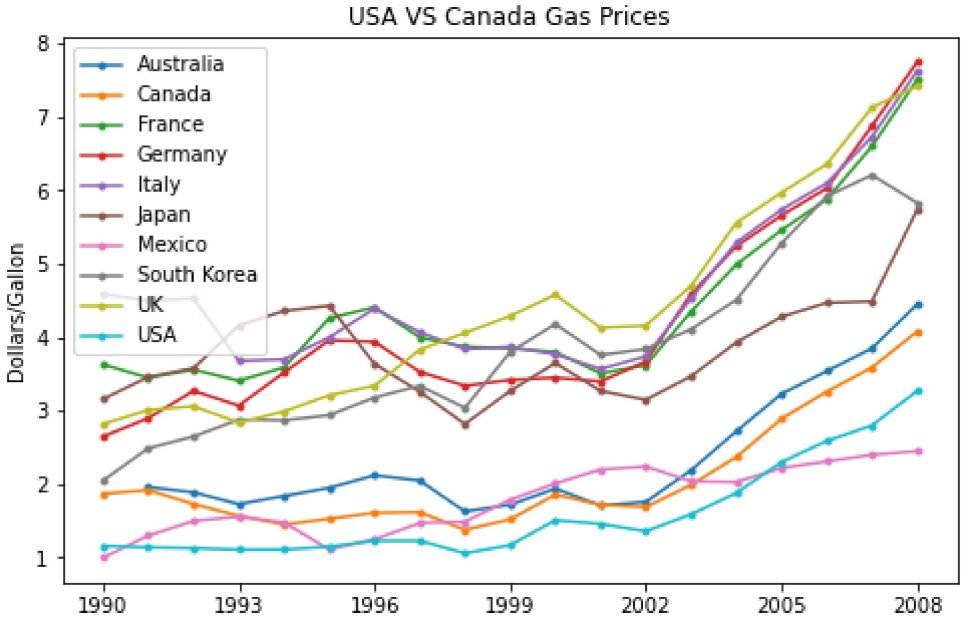
18.1.1 print point values of lines

In [105]:
```
# plt.figure(figsize=(8,5)) #x=8 inch ,y =5 inch
plt.title('jp VS uk Gas Prices')
plt.ylabel('Dollars/Gallon')
plt.xticks(df_gas['Year'][::3])

x1=df_gas['Year']
y1=df_gas['Japan']
y2=df_gas['UK']

plt.plot(x1,y1,'g.-',label='Japan')
plt.plot(x1,y2,'r.-',label='UK')

for a1,b1 in zip(x1,y1):
     plt.text(a1,b1,str(b1))

for a2,b2 in zip(x1,y2):
     plt.text(a2,b1,str(b2))

plt.legend() #to display the line label
plt.show()
```



18.2 quick way to show all lines

In [89]:
```
#1.commonly shared by lines
plt.figure(figsize=(8,5)) #x=8 inch ,y =5 inch
plt.title('USA VS Canada Gas Prices')
plt.ylabel('Dollars/Gallon')

#x-value intervals
plt.xticks(df_gas['Year'][::3])

#for each loop print all countries
for country in df_gas:
    if country!='Year' :
        plt.plot(df_gas['Year'],df_gas[country],marker='.',label=country)

#3.legend=傳說/圖像說明
plt.legend() #to display the line label
plt.show()
```



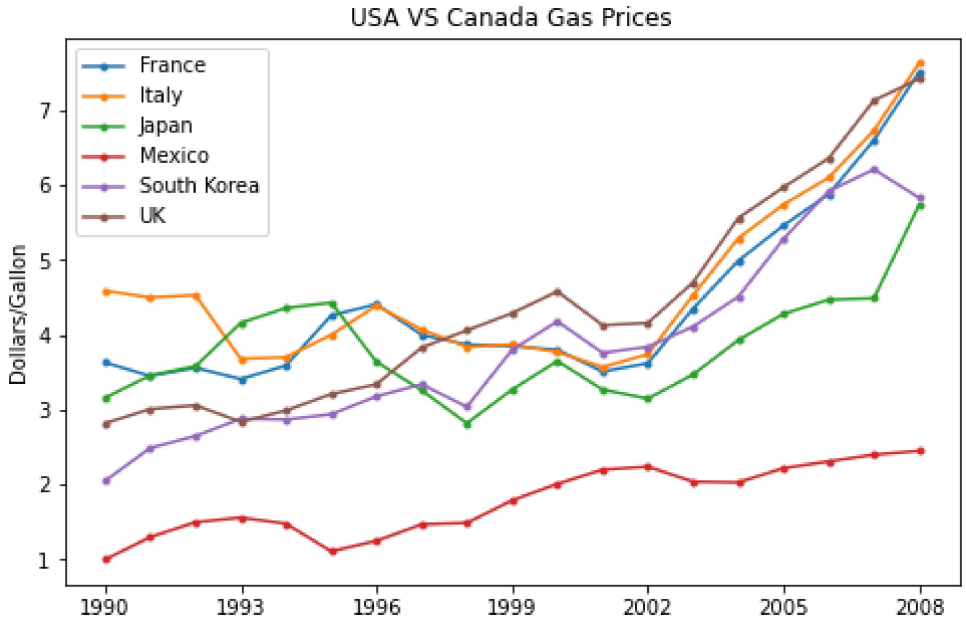
In [86]:
```
for country in df_gas:
    print(country)
```

```
Year
Australia
Canada
France
Germany
Italy
Japan
Mexico
South Korea
UK
USA
```

18.3 only not plot some lines/columns

In [92]:

```
#1.commonly shared by lines
plt.figure(figsize=(8,5)) #x=8 inch ,y =5 inch
plt.title('USA VS Canada Gas Prices')
plt.ylabel('Dollars/Gallon')

#x-value intervals
plt.xticks(df_gas['Year'][::3])

#make a not print list
country_not_look=['Australia','USA','Canada','Germany','USA']

#for each loop print all countries
for country in df_gas:
    #not use isin(); us in/not in
    if (country!='Year')&(country not in country_not_look):
        plt.plot(df_gas['Year'],df_gas[country],marker='.',label=country)

#3.Legend=傳說/圖像說明
plt.legend() #to display the line label
plt.show()
```



chapter 19 pie chart

```
1.bar is similar to line, but pie is another thing
2.bar need to change value to pct
syntax:
plt.pie()
a)labels
b)autopct:% of each parts
         autopct='%.2f':2 float
         autopct='%.2f%%': show % as well
c)explode(0,0,0.2,0)
          ~4 parts, 3rd part 凸其0.2格
          ~突出重點or 不要太靠近
d)pctdistince=0.6：pct 距離圓心0.6
```

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [10]:
```
#ref:df_gas=pd.read_csv('C:/Users/fengs/Desktop/pd_real/mathplot\gas_prices.csv')
df_fifa=pd.read_csv(r'C:/Users/fengs/Desktop/pd_real/mathplot\fifa_data.csv')
df_fifa.head()
```

Out[10]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	Ba
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	J
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Pari C
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Mar
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Mar

5 rows × 89 columns

chapter 19.1 simple pie of football

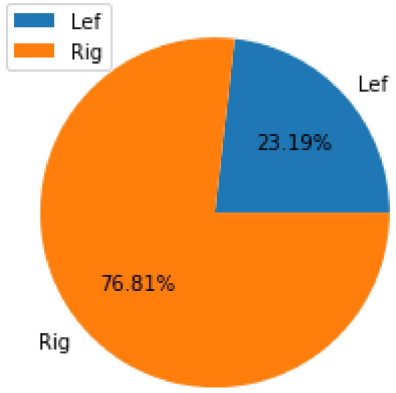
In [3]:
```
#1.df_fifa.columns
print(df_fifa['Preferred Foot'].unique())
#return the count value only,[0] fist
df_fifa.loc[df_fifa['Preferred Foot']=='Left'].count()[0]
```

```
['Left' 'Right' nan]
```

Out[3]:
```
4211
```

```
matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None,
                      pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0, radius=1,
                      counterclock=True, wedgeprops=None, textprops=None, center=(0, 0),
                         rotatelabels=False, *, normalize=True, data=None)
```

In [5]:
```
left=df_fifa.loc[df_fifa['Preferred Foot']=='Left'].count()[0]
right=df_fifa.loc[df_fifa['Preferred Foot']=='Right'].count()[0]

#[left,right] as only can fullin x, 1-d array
#'%.2f'=2 float, '%.2f%%'=add % sm
plt.pie([left,right],labels=['Lef','Rig'],autopct='%.2f%%')

plt.title=('foot preferences of players')
plt.legend()
plt.show()
```



chaper 19.2 more real case of footabll

In [21]:
```
"""background
1.df_fifa['Weight'].head()
 159lbs is str but not number, need to substr
 but some are number, some ar nan<null>
 bad data sameple

 # storing first 3 letters of name as username
 df['UserName'] = df['Name'].str[:3]

 #strip()=remove space,
 strip('xx')=remove char
"""
#1:update the Weight column
#'folat' object has no attribute 'strip'
#list comprehension, only get the true value
df_fifa['Weight']=[x.strip('lbs') if type(x)==str else x # 放前面可以令else做額外
                   for x in df_fifa['Weight']
                   ]
df_fifa['Weight'][0]  # get first number
#'159' is str but not number
```

Out[21]: '159'

```
use for loop+ if in dataframe
1.df['new column name'] = df['column name'].apply(
    lambda x: 'value if condition is met' if x condition
    else 'value if condition is not met')

```

Convert list of NaNs and strings to int?

In [ ]:
```

```

In [57]:
```
"""problem 1: 'list' object has no attribute 'astype'
一定是df=df.astype(int/float)
"""
# df_fifa['Weight']=[x.strip('lbs') if type(x)==str else x # 放前面可以令else做額外
#                    for x in df_fifa['Weight']
#                    ].astype[float]
#AttributeError: 'list' object has no attribute 'astype'


"""problems 2: some a nan so
failed to convert
ValueError: cannot convert float NaN to integer
#FAILED
df_fifa['Weight']=df_fifa['Weight'].astype(int)
"""
print()
```

note:find NaN rows

In [49]:
```
"""
df_fifa['Weight'].isna() only return T/F
"""
#S1:df.loc[<where conditions>,[cols]]
#df_fifa.loc[df_fifa['Weight'].isna(),['ID','Name','Age','Weight']].head()


"""
S2:no use pandas, but only add where
 #df_fifa[df_fifa['Weight'].isna()].head()
 #df_fifa[where conditions]

df_fifa[(df_fifa['Weight'].isna()) & (df_fifa.Age<30)] #valid
"""
df_fifa.loc[(df_fifa['Weight'].isna()) & (df_fifa.Age<30)
            ,df_fifa.columns.isin(['Age','ID','Name','Weight'])].head()
#isin is df attribute, if(x not in listx) when normal condition
```

Out[49]:

	ID	Name	Age	Weight
13237	195380	J. Barrera	29	NaN
13239	240437	A. Semprini	20	NaN
13240	209462	R. Bingham	24	NaN
13241	219702	K. Dankowski	21	NaN
13242	225590	I. Colman	23	NaN

note:conver nan in string cols to empty str

In [ ]:
```
df[['column1','column2']] = df[['column1','column2']].fillna('')
```

In [64]:
```
#1.fill in nan
#df_fifa['Weight']=df_fifa['Weight'].fillna('')

#2.check whether still nan row
#df_fifa[df_fifa['Weight'].isna()]

#3.change str to int
#ref:df['Price'] = df['Price'].astype(int)

type(df_fifa.dtypes) #pandas.core.series.Series
```

Out[64]:
```
pandas.core.series.Series
```

note:only not show some columns

In [54]:
```
"""
df.loc[<row condition>,<~df.columns.isin(['a','b'])>]

1/exclude one clos
 #df.loc[:, df.columns!='column1']

2/exclude some clos
df.loc[:, ~df.columns.isin(['a','b'])]
"""
df_fifa.loc[0:3,df_fifa.columns.isin(['Age','ID','Name','Weight'])]
```

Out[54]:

	ID	Name	Age	Weight
0	158023	L. Messi	31	159
1	20801	Cristiano Ronaldo	33	183
2	190871	Neymar Jr	26	150
3	193080	De Gea	27	168

note: drop cols in dataframe

In [ ]:
```
newdf = df.drop("age", axis='columns')
```

In [66]:
```
#delete dataframe
#del [[df_1,df_2]]
del [df_fifa]
```

19.2.1 start the football case

In [6]:
```
#1.del first
#del [df_fifa]
df_fifa=pd.read_csv(r'C:/Users/fengs/Desktop/pd_real/mathplot\fifa_data.csv')
df_fifa.head()
```

Out[6]:

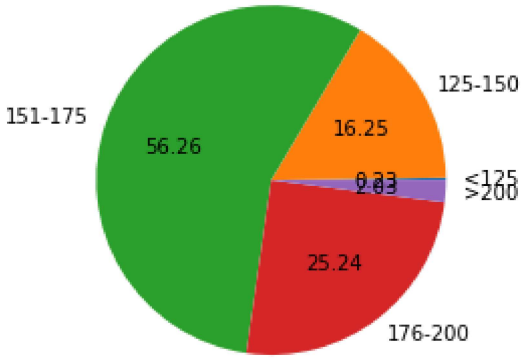
	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Cl
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	Barcelc
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	Juven
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Paris Sa Germ
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Manches Uni
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Manches C

5 rows × 89 columns

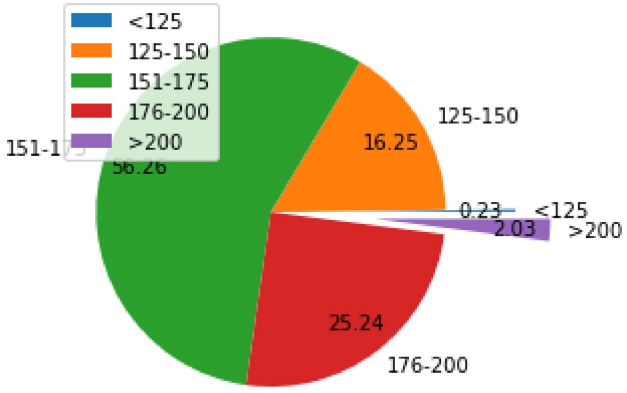
In [4]:
```
df_fifa['Weight']=[int(x.strip('lbs')) if type(x)==str else x # 放前面可以令else做額外
                   for x in df_fifa['Weight']
                   ]
print(df_fifa['Weight'][0])
print(df_fifa['Weight'].dtype)
```

```
159.0
float64
```

In [13]:
```
light=df_fifa.loc[df_fifa.Weight<125].count()[0]
medium=df_fifa.loc[df_fifa.Weight.between(125,150)].count()[0]
heavy=df_fifa.loc[df_fifa.Weight.between(151,175)].count()[0]
giant=df_fifa.loc[df_fifa.Weight.between(175,200)].count()[0]
sg=df_fifa.loc[df_fifa.Weight>200].count()[0]

#number list but not string name list,as put as x in plt.pie(x)
weights=[light,medium,heavy,giant,sg]
#labels names list
labels=['<125','125-150','151-175','176-200','>200']

plt.pie(weights,labels=labels,autopct='%0.2f')
plt.show()
```



In [24]:
```
"""fine-tune pie above
1.explode()
2.pctdistance=<0.6-0.8>
"""
plt.stype=('ggplot') # one color/display template
plt.title=('Weight of Player')

plt.pie(weights,labels=labels,autopct='%0.2f',pctdistance=0.8,explode=(0.4,0,0,0,0.6))
#as only <125 ,>200 are tight,數番上面weights[]排位

plt.legend() # give the color description
plt.show()
```



chapter 19.3 boxplot

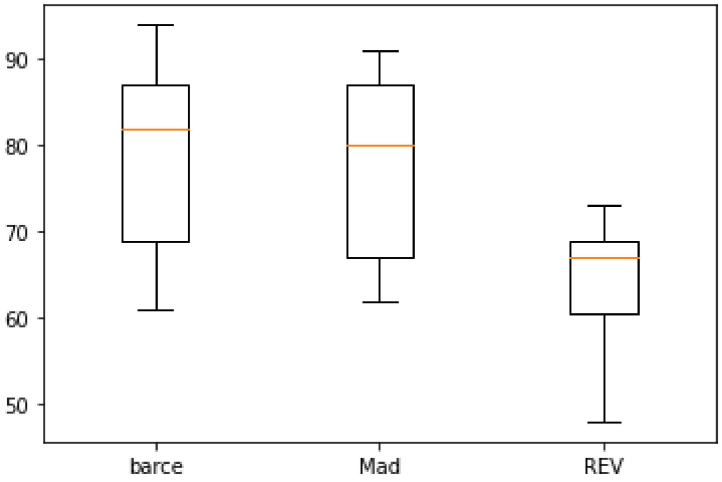
```
boxplot(箱形圖)compare 大家的 5個數字：
1.頭min · max,top 25%/75% · median,

display the five-number summary of a dataset
min/max/1st quartile/median/3rd quartile
```

19.3.1 basic example

In [ ]:
```
df=pf.DataFrame{'A':[1,1,2,2],
                'B':[''],

}
```

In [ ]:
```

```

In [30]:
```
"""syntax
plt.boxplot([d1,d2],....) [x,y] is data,if single just x,
"""
barcelona=df_fifa.loc[df_fifa.Club=='FC Barcelona']['Overall']
madrid=df_fifa.loc[df_fifa.Club=='Real Madrid']['Overall']
revs=df_fifa.loc[df_fifa.Club=='New England Revolution']['Overall']

plt.boxplot([barcelona,madrid,revs],labels=['barce','Mad','REV'])

plt.title=('Team Soccer Comparsion')
plt.ylabel=('Overall Rate')

plt.show()
```



chapter 19.x:create nested pie with subplot

In [ ]:
```

```

chpater 20:statistic Pandas

In [ ]:
```
1.video link:
    https://www.youtube.com/watch?v=txMdrV1Ut64
2.ref link
      https://www.geeksforgeeks.org/python-pandas-dataframe-groupby/
```

chapter 20.1 groupby

```
Syntax: df.groupby(by=None, axis=0, level=None,
              as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)
```

EX1: NBA worker info

In [7]:
```
#ref:df_fifa=pd.read_csv(r'C:/Users/fengs/Desktop/pd_real/mathplot\fifa_data.csv')
df_nba=pd.read_csv(r'C:\Users\fengs\Desktop\pd_real\nba.csv')
df_nba.head()
```

Out[7]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

In [8]:
```
#show unique
print(df_nba['Team'].unique())
```

Out[8]:
```
array(['Boston Celtics', 'Brooklyn Nets', 'New York Knicks',
       'Philadelphia 76ers', 'Toronto Raptors', 'Golden State Warriors',
       'Los Angeles Clippers', 'Los Angeles Lakers', 'Phoenix Suns',
       'Sacramento Kings', 'Chicago Bulls', 'Cleveland Cavaliers',
       'Detroit Pistons', 'Indiana Pacers', 'Milwaukee Bucks',
       'Dallas Mavericks', 'Houston Rockets', 'Memphis Grizzlies',
       'New Orleans Pelicans', 'San Antonio Spurs', 'Atlanta Hawks',
       'Charlotte Hornets', 'Miami Heat', 'Orlando Magic',
       'Washington Wizards', 'Denver Nuggets', 'Minnesota Timberwolves',
       'Oklahoma City Thunder', 'Portland Trail Blazers', 'Utah Jazz',
       nan], dtype=object)
```

In [14]:
```
"""
1.df.groupby(['<col_name>'].<arg>())
#<arg>.():mean(),count()
"""
team=df_nba.groupby('Team') #return a ref_no
team.first().head()  #first() return 1st record of each group
```

Out[14]:

Team	Name	Number	Position	Age	Height	Weight	College	Salary
Atlanta Hawks	Kent Bazemore	24.0	SF	26.0	6-5	201.0	Old Dominion	2000000.0
Boston Celtics	Avery Bradley	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
Brooklyn Nets	Bojan Bogdanovic	44.0	SG	27.0	6-8	216.0	Oklahoma State	3425510.0
Charlotte Hornets	Nicolas Batum	5.0	SG	27.0	6-8	200.0	Virginia Commonwealth	13125306.0
Chicago Bulls	Cameron Bairstow	41.0	PF	25.0	6-9	250.0	New Mexico	845059.0

In [16]:
```
#return a specfici group value
team.get_group('New York Knicks').tail(3)
```

Out[16]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
43	Sasha Vujacic	New York Knicks	18.0	SG	32.0	6-7	195.0	NaN	947276.0
44	Derrick Williams	New York Knicks	23.0	PF	25.0	6-8	240.0	Arizona	4000000.0
45	Tony Wroten	New York Knicks	5.0	SG	23.0	6-6	205.0	Washington	167406.0

In [18]:
```
"""group by 2 cols
"""
#use <df_name>.groupby, but not pd.groupby()
duo_group=df_nba.groupby(['Team','Position'])
duo_group.first()
```

Out[18]:

Team	Position	Name	Number	Age	Height	Weight	College	Salary
	C	Al Horford	15.0	30.0	6-10	245.0	Florida	12000000.0
	PF	Kris Humphries	43.0	31.0	6-9	235.0	Minnesota	1000000.0
Atlanta Hawks	PG	Dennis Schroder	17.0	22.0	6-1	172.0	Wake Forest	1763400.0
	SF	Kent Bazemore	24.0	26.0	6-5	201.0	Old Dominion	2000000.0
	SG	Tim Hardaway Jr.	10.0	24.0	6-6	205.0	Michigan	1304520.0
...	...	...	...	...	...	...	...	...
	C	Marcin Gortat	13.0	32.0	6-11	240.0	North Carolina State	11217391.0
	PF	Drew Gooden	90.0	34.0	6-10	250.0	Kansas	3300000.0
Washington Wizards	PG	Ramon Sessions	7.0	30.0	6-3	190.0	Nevada	2170465.0
	SF	Jared Dudley	1.0	30.0	6-7	225.0	Boston College	4375000.0
	SG	Alan Anderson	6.0	33.0	6-6	220.0	Michigan State	4000000.0

149 rows × 7 columns

In [27]:
```
"""2.with agg function
groupby(['co2l'])['col2'].sum()/mean()
yet didnot add any where conditon
"""
df_nba.groupby(['Team'])['Team'].count().head()
```

Out[27]:
```
Team
Atlanta Hawks        15
Boston Celtics       15
Brooklyn Nets        15
Charlotte Hornets    15
Chicago Bulls        15
Name: Team, dtype: int64
```

In [25]:
```
#same as above
df_nba.value_counts(['Team']).head()
```

Out[25]:
```
Team
New Orleans Pelicans    19
Memphis Grizzlies       18
New York Knicks         16
Milwaukee Bucks         16
Atlanta Hawks           15
dtype: int64
```

Groupby+agg()

In [31]:
```
#find the max,min age of each team
df_nba.groupby(['Team'])['Age'].agg(['min','max']).head()
```

Out[31]:

Team	min	max
Atlanta Hawks	22.0	35.0
Boston Celtics	20.0	29.0
Brooklyn Nets	21.0	32.0
Charlotte Hornets	21.0	31.0
Chicago Bulls	21.0	35.0

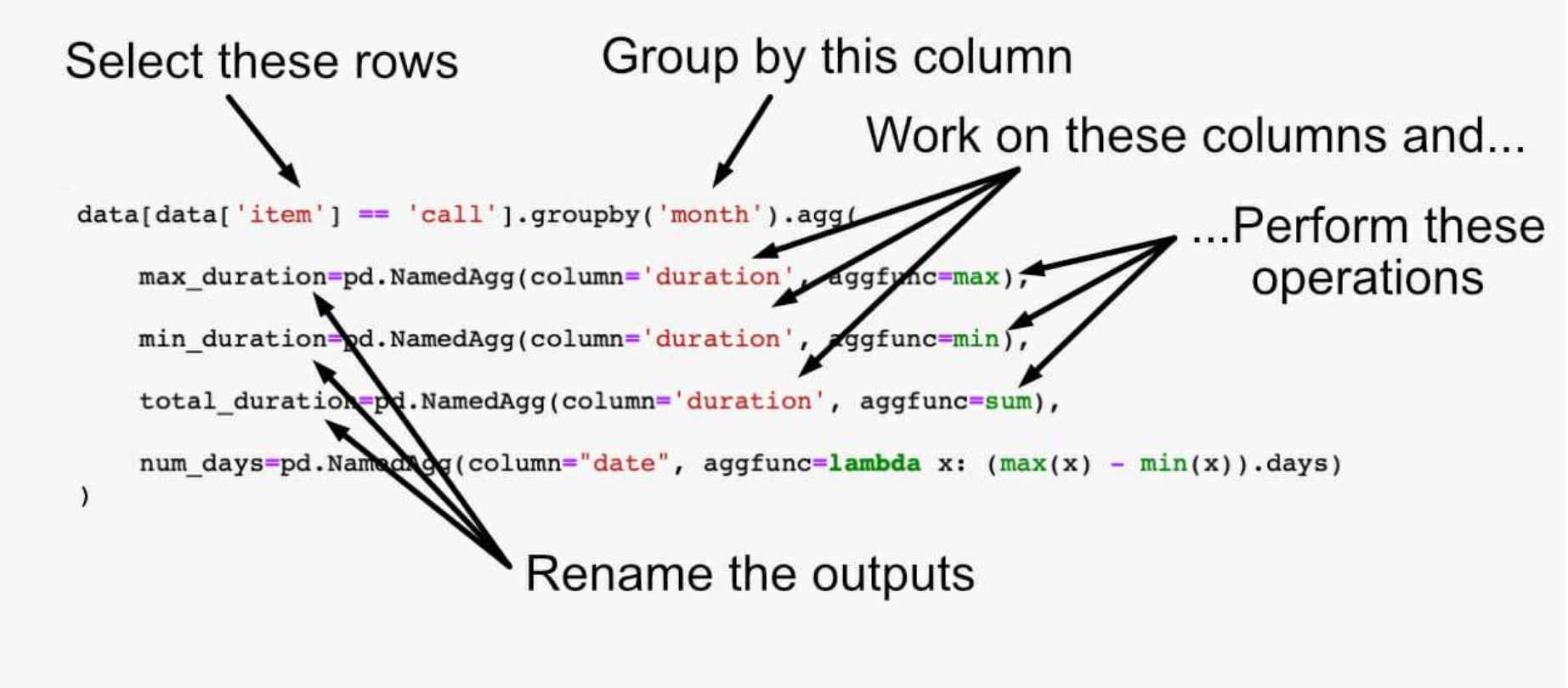
EX2: Phone Info(實用)

In [ ]:
```
useful link:
https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/
```

Note:Steps to insert image:

```
1.change cell to 'Markdown'but not code
2.Edit>inset image(bottom)
```

""" link: https://pandas.pydata.org/pandas-docs/version/0.22/generated (https://pandas.pydata.org/pandas-docs/version/0.22/generated) """



In [32]:
```
df_phone=pd.read_csv('C:/Users/fengs/Desktop/pd_real\phone_data.csv')
df_phone.head()
```

Out[32]:

	index	date	duration	item	month	network	network_type
0	0	15/10/14 06:58	34.429	data	2014-11	data	data
1	1	15/10/14 06:58	13.000	call	2014-11	Vodafone	mobile
2	2	15/10/14 14:46	23.000	call	2014-11	Meteor	mobile
3	3	15/10/14 14:48	4.000	call	2014-11	Tesco	mobile
4	4	15/10/14 17:27	4.000	call	2014-11	Tesco	mobile

In [34]:
```
"""
1.find the sum of duration of each month
#without agg
#need to consider sorting as well
"""
df_phone.groupby(['month'])['duration'].sum()
```

Out[34]:
```
month
2014-11    26639.441
2014-12    14641.870
2015-01    18223.299
2015-02    15522.299
2015-03    22750.441
Name: duration, dtype: float64
```

In [36]:
```
#get num of date each month
#in fact it count ref_no
df_phone.groupby('month')['date'].count()
```

Out[36]:
```
month
2014-11    230
2014-12    157
2015-01    205
2015-02    137
2015-03    101
Name: date, dtype: int64
```

In [43]:
```
"""
Q2.add where clause
df[where clause].groupby(['xx'])[<yy>].<agg>()
#note: it may combine with loc
"""
"""
1.valid
#df_phone.loc[df_phone['item']=='call']

2.return a memory serise num
#df_phone.loc[df_phone['item']=='call'].groupby('network')
#extract 1st entry of each group with first()
#df_phone.loc[df_phone['item']=='call'].groupby('network').first()

3.get values
#add ['<cal_cols>'].sum()
#df.loc[(w1)&(w2),['A','B','C']]
"""
#call only,sum of duration of each network
df_phone.loc[df_phone['item']=='call',['network','duration']].groupby('network').sum()
```

Out[43]:

network	duration
Meteor	7200.0
Tesco	13828.0
Three	36464.0
Vodafone	14621.0
landline	18433.0
voicemail	1775.0

In [49]:
```
"""
Q3:group by N cols,
loc[row,col]
"""
#if loc filter out cal cols,
#will error:'Column not found: date'
#df_phone.loc[:,['month','item']].groupby(['month','item'])['date'].count()

df_phone.loc[:,['month','item','date']].groupby(['month','item'])['date'].count()
```

Out[49]:
```
month    item
2014-11  call    107
         data     29
         sms      94
2014-12  call     79
         data     30
         sms      48
2015-01  call     88
         data     31
         sms      86
2015-02  call     67
         data     31
         sms      39
2015-03  call     47
         data     29
         sms      25
Name: date, dtype: int64
```

groupby +agg(Multi-stat)

```
# in realiy there will more than agg() cols
like the nba example, the min(),max() age of each team
```

```
#group by has sort/aixs args
DataFrame.groupby(by=None, axis=0, level=None, as_index=True,
                  sort=True, group_keys=True,
```

```
                  squeeze=NoDefault.no_default, observed=False,
                  dropna=True)

#sort: bool, default True
#axis{0 or 'index', 1 or 'columns'}, default 0,
  group by each row
 Split along rows (0) or columns (1).
#as_index:bool, default True
For aggregated output, return object with group labels as the index.
Only relevant for DataFrame input. as_index=False
is effectively "SQL-style" grouped output.
```

In [ ]:
```
"""
1.same meaning
"""
df_phone.groupby
```

In [ ]:
```

```

In [ ]:
```

```

chapter 21:pivot table

In [ ]:
```

```

In [ ]:
```

```

chapter 22:manipulate spreadsheets

In [ ]:
```

```

In [ ]:
```
1
```