dictionary

Dictionaries are used to store data values in **key:value** pair. We use "key" to access "value" in a dictionary, while use "index" to access "value" in List or numpy array. See link for simple introduction to dictionary. Using dictionary, we can construct and manage rather complex data structure.
Simple examples of dictionary and List are shown below.

In [1]:
```
import numpy as np

#  dictionary
myDict={"brand": "Ford", "model": "Mustang", "year": 1964}
#  access value with "key"
print(type(myDict))
print(myDict["brand"])
print(myDict["year"])

#  List
myList=[ "Ford", "Mustang", 1964]
# access value with index.  0 for brand, 1 for model, 2 for year
print(type(myList))
print(myList[0])
print(myList[2])

#  numpy array
myNpArray=np.array([231, 987, 645])
# access value with index.
print(type(myNpArray))
print(myNpArray[0])
print(myNpArray[2])
```

```
<class 'dict'>
Ford
1964
<class 'list'>
Ford
1964
<class 'numpy.ndarray'>
231
645
```

In [2]:
```
#. print dictionary, list, numpy array
print(myDict)
print(myList)
print(myNpArray)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
['Ford', 'Mustang', 1964]
[231 987 645]
```

loop through a dictionary

We can loop through a dictionary by using for loop.

In [3]:
```
myDict={"brand": "Ford", "model": "Mustang", "year": 1964}

# loop over keys.
for k in myDict:
    print("key",k, type(k))
    v=myDict[k]
    print("val",v, type(v))
```

```
key brand <class 'str'>
val Ford <class 'str'>
key model <class 'str'>
val Mustang <class 'str'>
key year <class 'str'>
val 1964 <class 'int'>
```

In [4]:
```
# loop over values
for v in myDict.values():
    print("val",v, type(v))
```

```
val Ford <class 'str'>
val Mustang <class 'str'>
val 1964 <class 'int'>
```

In [5]:
```
# get a list of keys
ks=myDict.keys()
print(ks)
print(type(ks))
```

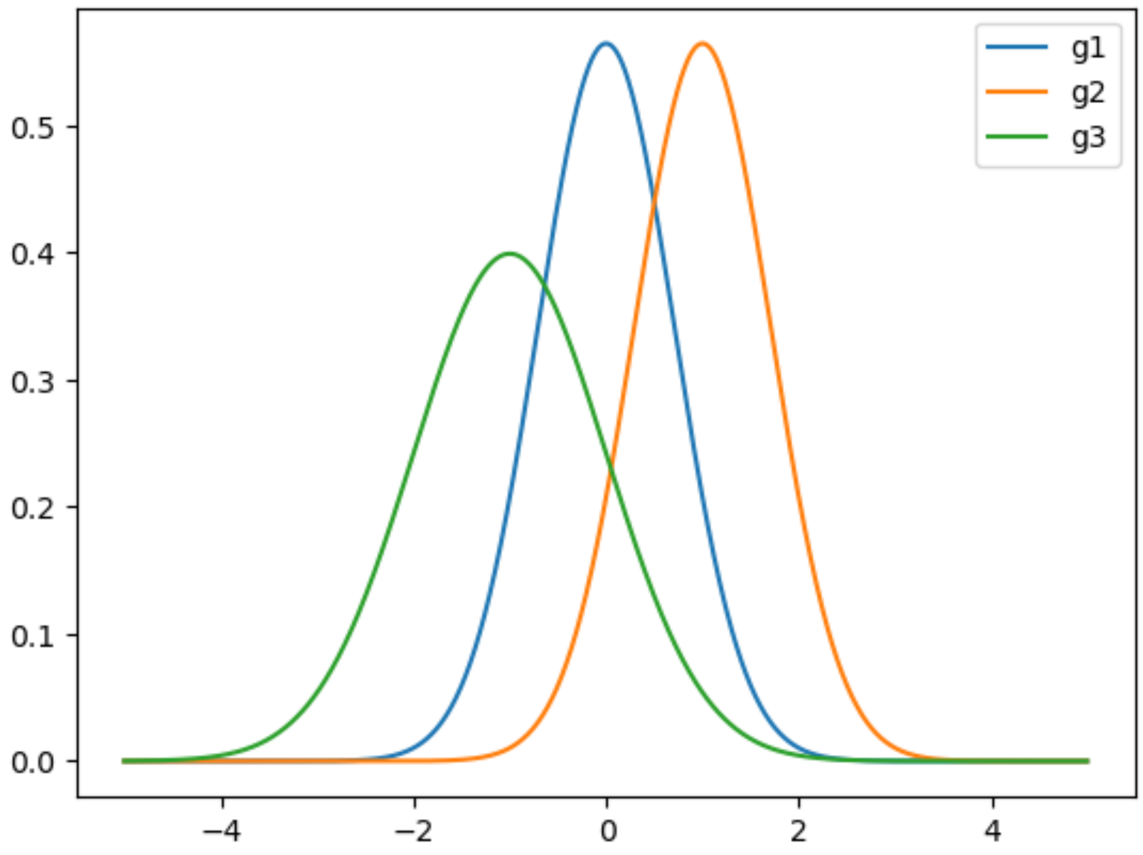
```
dict_keys(['brand', 'model', 'year'])
<class 'dict_keys'>
```

Constructing data structure using dictionary

In the follwoing example, we borrow a dataframe concept from Pandas
and use variable number of parameters in fucntion call. See link more about functions.

```
In [6]: import numpy as np
        import matplotlib.pyplot as plt

        def mygauss(x,**kwargs):
            # **kwargs: key word arguments packed into a dictionary
            # equation for gaussian from week3.
            print("mygauss:  type(kwards)",type(kwargs),kwargs)
            mu=kwargs["mu"]
            sig=kwargs["sigma"]
            a=1.0/np.sqrt(2.0*sig*np.pi)
            b=(x-mu)*(x-mu)/(2.0*sig)
            return a*np.exp(-b)

        def plotCurves(df):

            for k in df:
                if k=="x":
                    continue
                print("plot ",k)
                plt.plot(df["x"],df[k],label=k)
            plt.legend(loc="upper right")
            plt.show()

        # this program starts here...
        df={}      # creating an empty dictioany

        #. create 1st data set
        x=np.arange(-5.0, 5.0, 0.01)
        df["x"]=x

        #  store curves in a dictionary "df"
        df["g1"]=mygauss(x,mu=0.0,sigma=0.5)
        df["g2"]=mygauss(x,mu=1.0,sigma=0.5)
        df["g3"]=mygauss(x,mu=-1.0,sigma=1.0)

        # give the dataframe df to function plotCurves...
        plotCurves(df)
```

```
mygauss:  type(kwards) <class 'dict'> {'mu': 0.0, 'sigma': 0.5}
mygauss:  type(kwards) <class 'dict'> {'mu': 1.0, 'sigma': 0.5}
mygauss:  type(kwards) <class 'dict'> {'mu': -1.0, 'sigma': 1.0}
plot  g1
plot  g2
plot  g3
```



```
In [ ]:
```