Funtions in Python

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

In [1]:

```
def num():
    return 'hello world!'
```

In [19]:

```
car_data = {"mercedes": "1,2,3vr", "honda":"v8.90 exsit"}
user_name = input("enter your name: ")
choice = input("what car do you want: ")
for car in car_data.keys():
    if car == choice:
        print(f"{user_name} we have {choice}\ndetails: {car_data[car]}")
```

```
enter your name: caleb
what car do you want: honda
caleb we have honda
details: v8.90 exsit
```

In [17]:

```
def car_sale(car_repo):
    user_name = input("enter your name: ")
    choice = input("what car do you want: ")
    for choice in car_repo.keys():
        if car == choice:
            return f"{user_name} we have {choice} details: {car_data[car]}"
```

In [18]:

```
car_sale(car_data)
```

```
enter your name: caleb
what car do you want: honda
```

Out[18]:

```
'caleb we have honda details: v8.90 exsit'
```

In [ ]:

```

```

In [2]:

```
num()
```

Out[2]:

```
'hello world!'
```

In [3]:

```
def add(x):
    return 2+x
```

In [4]:

```
add(3)
```

Out[4]:

5

In [5]:

```
import math as m
```

In [6]:

```
m.sqrt(4)
```

Out[6]:

2.0

Introduction to

- numpy
- pandas
- matplotlib
- Data Understanding using the above modules

In [1]:

```
# importing libraries
import pandas as pd
```

In [25]:

```
house = {"door_type":[1,2,3,4,5,6,12,8,9,0],
        "light":[6,5,4,34,11,1,23,45,67,89],
        "roof": ['high','low','mid','cid','kee', 'u','me','nine', 'ten', 'comp
        'price': [100,200,300,400,500,600,700,800,900,1000]}
```

In [26]:

```
data = pd.DataFrame(data = house, columns = house.keys())
```

In [27]:

```
1 data
```

Out[27]:

	door_type	light	roof	price
0	1	6	high	100
1	2	5	low	200
2	3	4	mid	300
3	4	34	cid	400
4	5	11	kee	500
5	6	1	u	600
6	12	23	me	700
7	8	45	nine	800
8	9	67	ten	900
9	0	89	complet	1000

In [7]:

```
1 data.head(10)
```

Out[7]:

	door_type	light	roof	price
0	1	6	high	100
1	2	5	low	200
2	3	4	mid	300
3	4	34	cid	400
4	5	11	kee	500
5	6	1	u	600
6	12	23	me	700
7	8	45	nine	800
8	9	67	ten	900
9	0	89	complet	1000

In [30]:

```
1 data.tail(2)
```

Out[30]:

	door_type	light	roof	price
8	9	67	ten	900
9	0	89	complet	1000

In [31]:

```
1 data.shape #to get the nunmber of rows and columns in the dataset
```

Out[31]:

```
(10, 4)
```

In [33]:

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   door_type  10 non-null     int64 
 1   light      10 non-null     int64 
 2   roof       10 non-null     object
 3   price      10 non-null     int64 
dtypes: int64(3), object(1)
memory usage: 448.0+ bytes
```

In [34]:

```
1 data.describe()
```

Out[34]:

	door_type	light	price
count	10.000000	10.000000	10.000000
mean	5.000000	28.500000	550.000000
std	3.800585	30.178175	302.765035
min	0.000000	1.000000	100.000000
25%	2.250000	5.250000	325.000000
50%	4.500000	17.000000	550.000000
75%	7.500000	42.250000	775.000000
max	12.000000	89.000000	1000.000000

In [31]:

```
1 a =3
2 a = 5
3 print(a)
```

```
5
```

In [33]:

```
data.drop('roof', axis =1, inplace = True)
data
```

Out[33]:

	door_type	light	price
0	1	6	100
1	2	5	200
2	3	4	300
3	4	34	400
4	5	11	500
5	6	1	600
6	12	23	700
7	8	45	800
8	9	67	900
9	0	89	1000

In [34]:

```
x = data.drop(['price','door_type'], axis =1)
```

In [ ]:

```
data = data.drop('roof', axis =1)
```

In [66]:

```
x
```

Out[66]:

	light
0	6
1	5
2	4
3	34
4	11
5	1
6	23
7	45
8	67
9	89

In [58]:

```
1 data[["price", "door_type"]].describe()
```

Out[58]:

	price	door_type
count	10.000000	10.000000
mean	550.000000	5.000000
std	302.765035	3.800585
min	100.000000	0.000000
25%	325.000000	2.250000
50%	550.000000	4.500000
75%	775.000000	7.500000
max	1000.000000	12.000000

In [62]:

```
1 type(data.price.values)
```

Out[62]:

```
numpy.ndarray
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [35]:

```
1 data
```

Out[35]:

	door_type	light	price
0	1	6	100
1	2	5	200
2	3	4	300
3	4	34	400
4	5	11	500
5	6	1	600
6	12	23	700
7	8	45	800
8	9	67	900
9	0	89	1000

In [8]:

```
1 #this shows/displays the first five rows in a dataset
2 data.head(4)
```

Out[8]:

	door_type	roof	price
0	1	high	100
1	2	low	200
2	3	mid	300
3	4	cid	400

In [7]:

```
1 #this shows/displays the last five rows in a dataset
2 data.tail()
```

Out[7]:

	door_type	roof	price
5	6	u	600
6	12	me	700
7	8	nine	800
8	9	ten	900
9	0	complet	1000

In [19]:

```
#to display the shape of the dataset
data.shape
```

Out[19]:

```
(10, 4)
```

In [20]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   door_type  10 non-null     int64 
 1   light      10 non-null     int64 
 2   roof       10 non-null     object
 3   price      10 non-null     int64 
dtypes: int64(3), object(1)
memory usage: 448.0+ bytes
```

In [21]:

```
#this displays a short statistics of the dataset
data.describe()
```

Out[21]:

	door_type	light	price
count	10.000000	10.000000	10.000000
mean	5.000000	28.500000	550.000000
std	3.800585	30.178175	302.765035
min	0.000000	1.000000	100.000000
25%	2.250000	5.250000	325.000000
50%	4.500000	17.000000	550.000000
75%	7.500000	42.250000	775.000000
max	12.000000	89.000000	1000.000000

In [37]:

```
data.columns
```

Out[37]:

```
Index(['door_type', 'light', 'price'], dtype='object')
```

In [38]:

```
x = list(data.columns)
print(x)
```

```
['door_type', 'light', 'price']
```

In [44]:

```
x = 'door types', 'lightings', 'prices'
```

In [45]:

```
data.columns = x
```

In [46]:

```
data
```

Out[46]:

	door types	lightings	prices
0	1	6	100
1	2	5	200
2	3	4	300
3	4	34	400
4	5	11	500
5	6	1	600
6	12	23	700
7	8	45	800
8	9	67	900
9	0	89	1000

In [53]:

```
1 data[["door types"]]
```

Out[53]:

	door types
0	1
1	2
2	3
3	4
4	5
5	6
6	12
7	8
8	9
9	0

In [55]:

```
1 data[['lightings', 'prices']]
```

Out[55]:

	lightings	prices
0	6	100
1	5	200
2	4	300
3	34	400
4	11	500
5	1	600
6	23	700
7	45	800
8	67	900
9	89	1000

In [28]:

```
1 data
```

Out[28]:

	door_type	light	roofing	price
0	1	6	high	100
1	2	5	low	200
2	3	4	mid	300
3	4	34	cid	400
4	5	11	kee	500
5	6	1	u	600
6	12	23	me	700
7	8	45	nine	800
8	9	67	ten	900
9	0	89	complet	1000

In [60]:

```
1 data.iloc[2:8,1:-1]
```

Out[60]:

	lightings
2	4
3	34
4	11
5	1
6	23
7	45

In [30]:

```
1 y = data[['price']]
```

In [31]:

```
1 x
```

Out[31]:

	door_type	light	roofing
0	1	6	high
1	2	5	low
2	3	4	mid
3	4	34	cid
4	5	11	kee
5	6	1	u
6	12	23	me
7	8	45	nine
8	9	67	ten
9	0	89	complet

In [32]:

```
1 y
```

Out[32]:

	price
0	100
1	200
2	300
3	400
4	500
5	600
6	700
7	800
8	900
9	1000

In [33]:

```
1 data
```

Out[33]:

	door_type	light	roofing	price
0	1	6	high	100
1	2	5	low	200
2	3	4	mid	300
3	4	34	cid	400
4	5	11	kee	500
5	6	1	u	600
6	12	23	me	700
7	8	45	nine	800
8	9	67	ten	900
9	0	89	complet	1000

In [34]:

```
1 import seaborn as sns
```

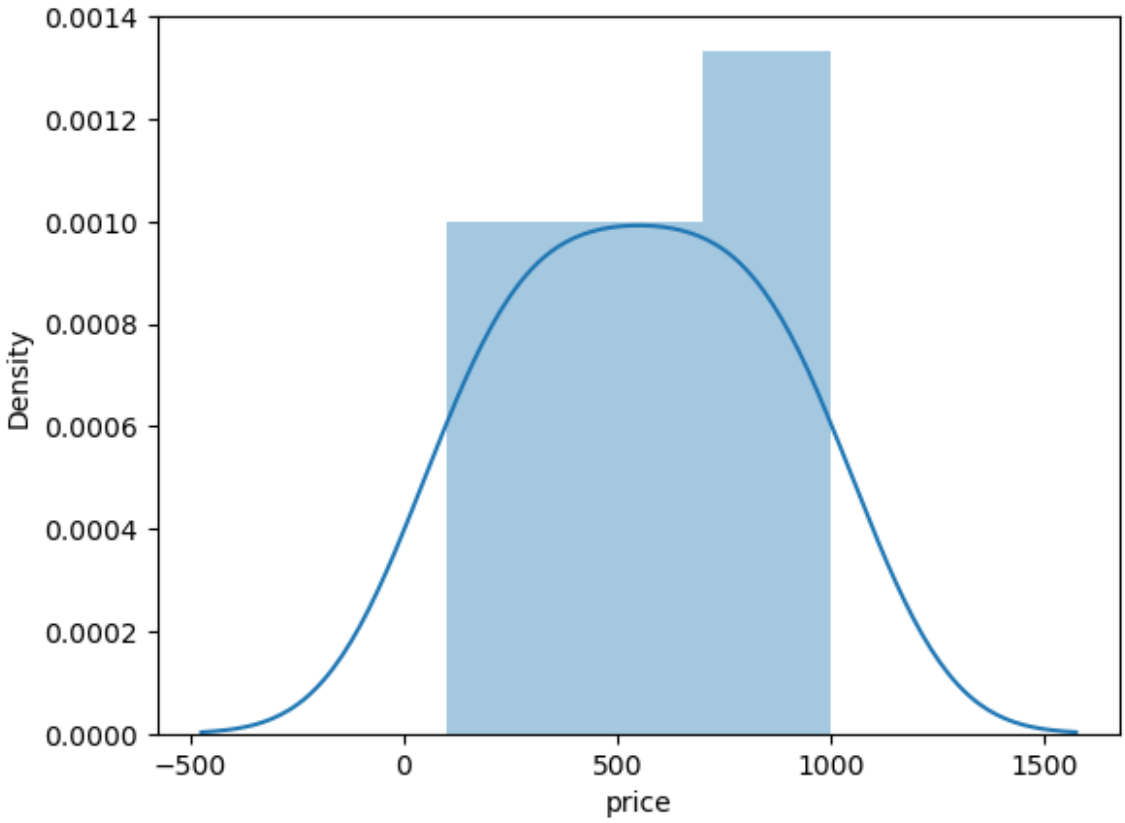
In [35]:

```
1 sns.distplot(data.price)
```

```
/home/C4LEB/anaconda3/lib/python3.9/site-packages/seaborn/distributi
ons.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use e
ither `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

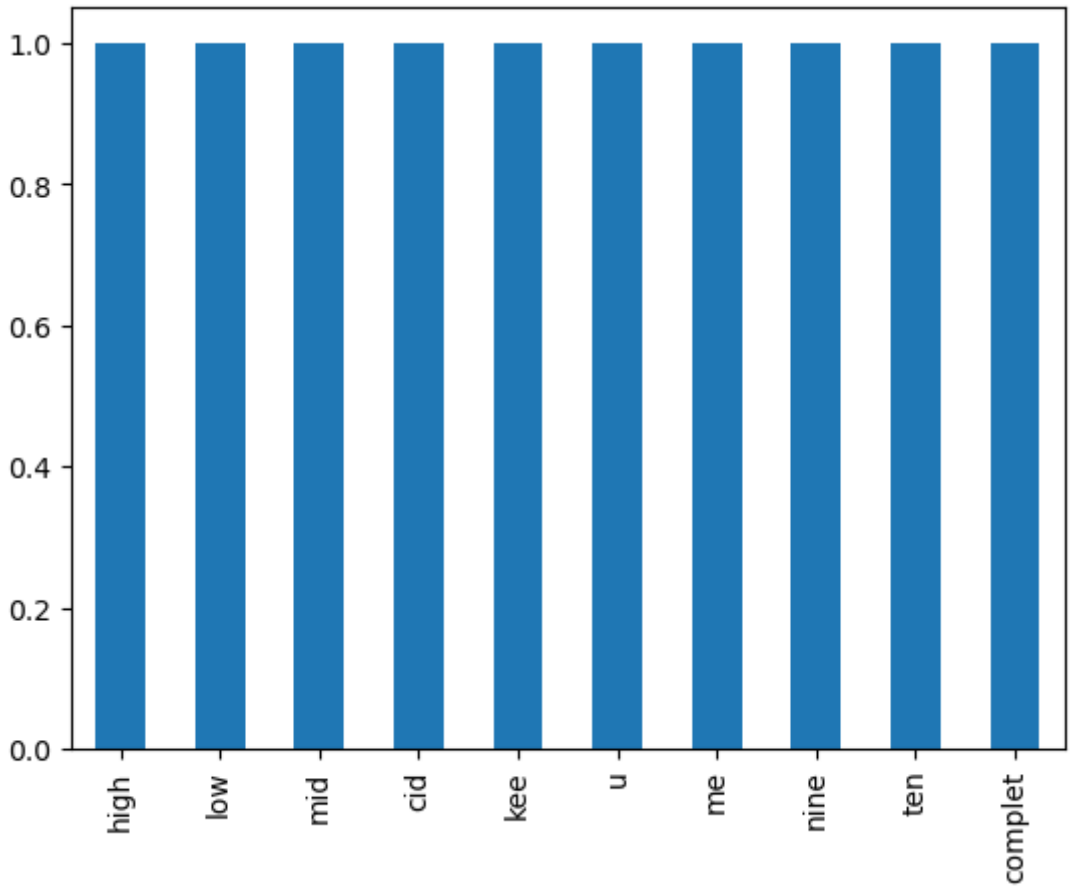
Out[35]:

```
<AxesSubplot:xlabel='price', ylabel='Density'>
```



In [36]:

```
1 data.roofing.value_counts().plot(kind = 'bar');
```



In [37]:

```
1 data
```

Out[37]:

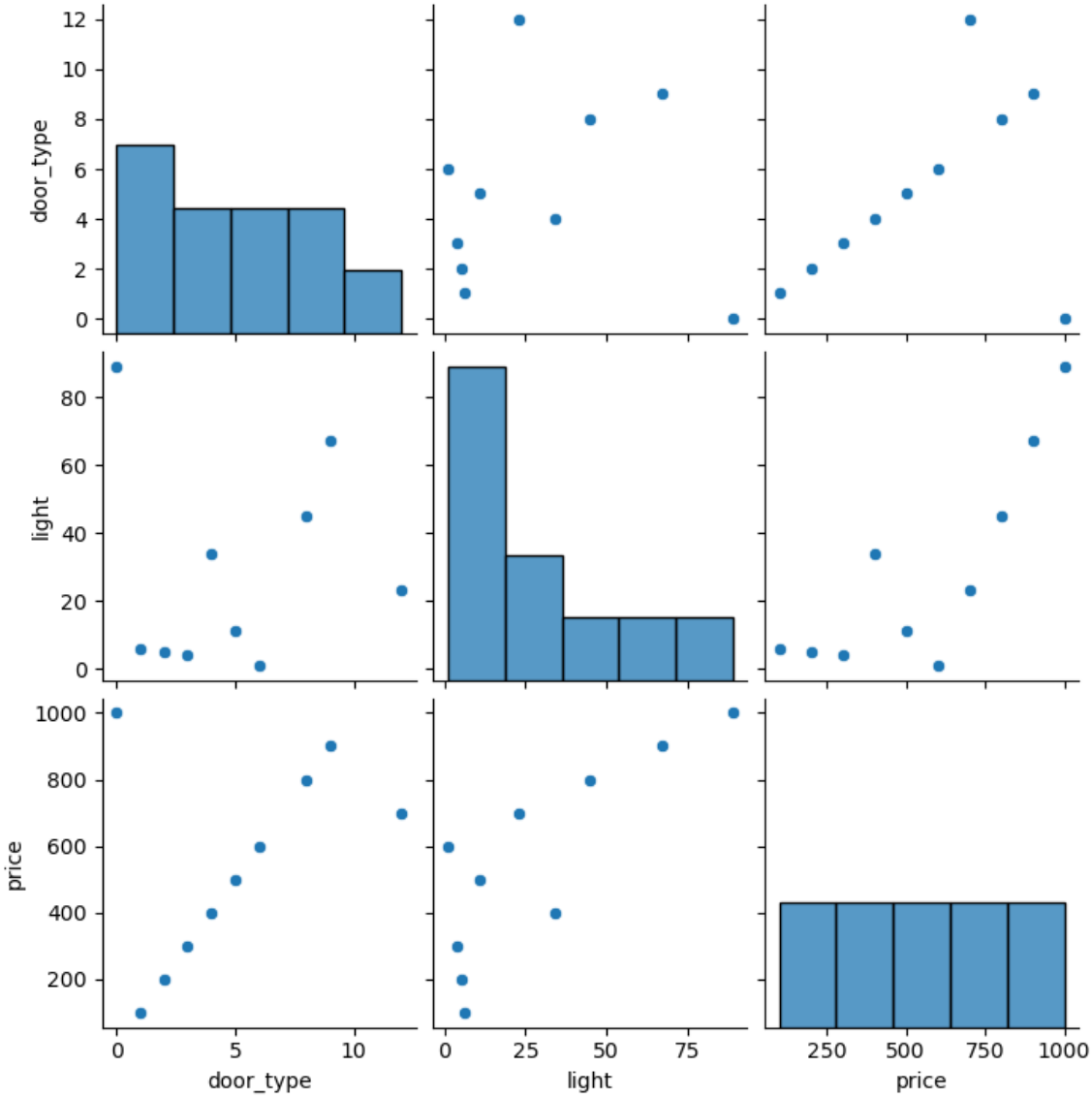
	door_type	light	roofing	price
0	1	6	high	100
1	2	5	low	200
2	3	4	mid	300
3	4	34	cid	400
4	5	11	kee	500
5	6	1	u	600
6	12	23	me	700
7	8	45	nine	800
8	9	67	ten	900
9	0	89	complet	1000

In [ ]:

```
1
```

In [39]:

```
1 sns.pairplot(data);
```



In [ ]:

```
1
```