LABSHEET_2(PANDAS)

```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d',])
s = pd.Series(data)
print(s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data,index=[100,101,102,103])
print(s)
```

```
100    a
101    b
102    c
103    d
dtype: object
```

```
import pandas as pd
import numpy as np
data = {'a' : 0.,'b':1.,'c':2.}
s = pd.Series(data)
print(s)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

```
import pandas as pd
import numpy as np
data = {'a' : 0.,'b':1.,'c':2.}
s = pd.Series(data,index=['a','b','c','d'])
print(s)
```

```
a    0.0
b    1.0
c    2.0
d    NaN
dtype: float64
```

```
import pandas as pd
import numpy as np
s = pd.Series(5,index=[0,1,2,3])
print(s)
```

```
0    5
1    5
2    5
3    5
dtype: int64
```

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
#retrieve the first element
print(s[0])
```

```
1
```

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
print(s)
#retrieve the first three element
print(s[:3])
```

```
a    1
b    2
c    3
d    4
e    5
```

```
dtype: int64
a    1
b    2
c    3
dtype: int64
```

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
#retrieve a single element
print(s['a'])
```

```
1
```

```
#retrieve multiple element
print(s[['a','c','d']])
```

```
a    1
c    3
d    4
dtype: int64
```

```
import pandas as pd
df = pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

```
data = [1,2,3,4,5]
df = pd.DataFrame(data)
print(df)
```

```
   0
0  1
1  2
2  3
3  4
4  5
```

```
data = (['alex',10],['bob',12],['clarke',13])
df = pd.DataFrame(data,columns =['name','age'])
print(df)
```

```
     name  age
0    alex   10
1     bob   12
2  clarke   13
```

```
data = (['alex',10],['bob',12],['clarke',13])
df = pd.DataFrame(data,columns =['name','age'],dtype=float)
print(df)
```

```
     name   age
0    alex  10.0
1     bob  12.0
2  clarke  13.0
<ipython-input-20-f26b38ceada8>:2: FutureWarning: Could not cast to float64, falling back to object. This behavior is deprecated. In
  df = pd.DataFrame(data,columns =['name','age'],dtype=float)
```

```
data = {'name':['tom','jack','steve','ricky'],'age' :[28,34,29,42]}
df = pd.DataFrame(data,index =['rank1','rank2','rank3','rank4'])
print(df)
```

```
         name  age
rank1    tom   28
rank2   jack   34
rank3  steve   29
rank4  ricky   42
```

```
import pandas as pd
data = [{'a': 1,'b':2},{'a': 10,'b':20,'c':30}]
print(pd.DataFrame(data))
```

```
    a   b     c
0   1   2   NaN
1  10  20  30.0
```

```
import pandas as pd
#the following examples shows how to create a dataframe by passing a list of dictionaries and the row indicies
data = [{'a': 1,'b':2},{'a': 10,'b':20,'c':30}]
print(pd.DataFrame(data,index = ['first','second']))
```

```
            a   b     c
first       1   2   NaN
second     10  20  30.0
```

```
data = [{'a': 1,'b':2},{'a': 10,'b':20,'c':30}]
#with two columns indices values same as dictionary keys
df1 = pd.DataFrame(data,index =['first','second'],columns=['a','b'])
#with two column indices,with one index,with othername
df2 = pd.DataFrame(data,index =['first','second'],columns=['a','b1'])
print(df1)
print(df2)
```

```
            a   b
first       1   2
second     10  20
            a  b1
first       1 NaN
second     10 NaN
```

```
df3 = pd.DataFrame(data,index =['first','second'],columns=['a','b1','b'])
df3
```

	a	b1	b
first	1	NaN	2
second	10	NaN	20

```
d = {'one':pd.Series([1,2,3],index =['a','b','c']),'two':pd.Series([1,2,3,4],index =['a','b','c','d'])}
df=pd.DataFrame(d)
df
```

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

```
import pandas as pd
d = {'one':pd.Series([1,2,3],index =['a','b','c']),'two':pd.Series([1,2,3,4],index =['a','b','c','d'])}
df=pd.DataFrame(d)
#adding a new column to an existing dataframe object with column label by passing
print("adding a new column by passing as Series:")
df['three']=pd.Series([10,20,30],index=['a','b','c'])
print(df)
```

```
adding a new column by passing as Series:
   one  two  three
a  1.0    1   10.0
b  2.0    2   20.0
c  3.0    3   30.0
d  NaN    4    NaN
```

```
df['four']=df['one']+df['two']
df
```

	one	two	three	four
a	1.0	1	10.0	2.0
b	2.0	2	20.0	4.0
c	3.0	3	30.0	6.0
d	NaN	4	NaN	NaN

```
#using del function
print("deleting the first column using DEL function:")
del(df['one'])
print(df)
```

```
deleting the first column using DEL function:
   two  three  four
a    1   10.0   2.0
b    2   20.0   4.0
c    3   30.0   6.0
d    4    NaN   NaN
```

```
#using pop function
df.pop('two')
df
```

	three	four
a	10.0	2.0
b	20.0	4.0
c	30.0	6.0
d	NaN	NaN

```
d = {'one':pd.Series([1,2,3],index =['a','b','c']),'two':pd.Series([1,2,3,4],index =['a','b','c','d'])}
df=pd.DataFrame(d)
print(df.loc['b'])
```

```
one    2.0
two    2.0
Name: b, dtype: float64
```

```
d = {'one':pd.Series([1,2,3],index =['a','b','c']),'two':pd.Series([1,2,3,4],index =['a','b','c','d'])}
df=pd.DataFrame(d)
print(df.iloc[0])
```

```
one    1.0
two    1.0
Name: a, dtype: float64
```

```
df[2:4]
```

	one	two
c	3.0	3
d	NaN	4

```
#adding new rows t a DataFrame using the append function.This function will append the rows at the end
df = pd.DataFrame([[1,2],[3,4]],columns = ['a','b'])
df2 = pd.DataFrame([[5,6],[7,8]],columns = ['a','b'])
df = df.append(df2)
print(df)
```

```
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
<ipython-input-19-cc1082fcfe80>:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  df = df.append(df2)
```

```
#drop rows with label 0
df= df.drop(0)
df
```

	a	b
1	3	4
1	7	8

loading the data

```
import pandas as pd
df=pd.read_csv("/content/german_credit_data.csv")
df
```

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
0	0	67	male	2	own	NaN	little	1169	6	radio/TV
1	1	22	female	2	own	little	moderate	5951	48	radio/TV
2	2	49	male	1	own	little	NaN	2096	12	education
3	3	45	male	2	free	little	little	7882	42	furniture/equipment
4	4	53	male	2	free	little	little	4870	24	car
...	...	...	...	...	...	...	...	...	...	...
995	995	31	female	1	own	little	NaN	1736	12	furniture/equipment
996	996	40	male	3	own	little	little	3857	30	car
997	997	38	male	2	own	little	NaN	804	12	radio/TV
998	998	23	male	2	free	little	little	1845	45	radio/TV
999	999	27	male	2	own	moderate	moderate	4576	45	car

1000 rows × 10 columns

```
df.head()
```

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
0	0	67	male	2	own	NaN	little	1169	6	radio/TV
1	1	22	female	2	own	little	moderate	5951	48	radio/TV
2	2	49	male	1	own	little	NaN	2096	12	education
3	3	45	male	2	free	little	little	7882	42	furniture/equipment
4	4	53	male	2	free	little	little	4870	24	car

```
df.tail()
```

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
995	995	31	female	1	own	little	NaN	1736	12	furniture/equipment
996	996	40	male	3	own	little	little	3857	30	car
997	997	38	male	2	own	little	NaN	804	12	radio/TV
998	998	23	male	2	free	little	little	1845	45	radio/TV
999	999	27	male	2	own	moderate	moderate	4576	45	car

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Unnamed: 0        1000 non-null   int64 
 1   Age               1000 non-null   int64 
 2   Sex               1000 non-null   object
 3   Job               1000 non-null   int64 
 4   Housing           1000 non-null   object
 5   Saving accounts   817 non-null    object
 6   Checking account  606 non-null    object
 7   Credit amount     1000 non-null   int64 
 8   Duration          1000 non-null   int64 
 9   Purpose           1000 non-null   object
dtypes: int64(5), object(5)
memory usage: 78.2+ KB
```

```
df.describe()
```

	Unnamed: 0	Age	Job	Credit amount	Duration
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	499.500000	35.546000	1.904000	3271.258000	20.903000
std	288.819436	11.375469	0.653614	2822.736876	12.058814
min	0.000000	19.000000	0.000000	250.000000	4.000000
25%	249.750000	27.000000	2.000000	1365.500000	12.000000
50%	499.500000	33.000000	2.000000	2319.500000	18.000000
75%	749.250000	42.000000	2.000000	3972.250000	24.000000
max	999.000000	75.000000	3.000000	18424.000000	72.000000