In [63]:
```
img1.save(r"C:\Users\CTTC\Downloads\modified_img.jpg")
```

PANDAS:- Panel Data Analysis

It is a python library that helps to deal with analyzing and manipulating the datas.
We can also read datasets using pandas.
Pandas is 2 types:-

- Series (single column)
- DataFrame (multiple columns)

In [64]:
```
# importing the library:
import pandas as pd
```

In [65]:
```
# Creating a Series:-
s = pd.Series(['a','b','c','d'])
s
```

Out[65]:
```
0    a
1    b
2    c
3    d
dtype: object
```

In [66]:
```
# assigning index values to the series.
s1 = pd.Series(['a','b','c'],index=[10,20,30])
s1
```

Out[66]:
```
10    a
20    b
30    c
dtype: object
```

In [67]:
```
# Indexing in Series:-
s1[20]
```

Out[67]:
```
'b'
```

In [69]:
```
# Slicing:
s1[::2]
```

Out[69]:
```
10    a
30    c
dtype: object
```

In [70]:
```
# Creating a Series from tuple:
s2 = ('apple','bat','cow','dog')
```

```
type(s2)
```

Out[70]: tuple

In [71]:
```
pd.Series(s2)
```

Out[71]:
```
0    apple
1      bat
2      cow
3      dog
dtype: object
```

In [73]:
```
# Creating a Series from set
s = {100,200,300,400,500}
pd.Series(s)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[73], line 3
      1 # Creating a Series from set
      2 s = {100,200,300,400,500}
----> 3 pd.Series(s)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\series.py:512, in Series.__init__(self, data, index, dtype, name, copy, fastpath)
    510         data = data.copy()
    511 else:
--> 512     data = sanitize_array(data, index, dtype, copy)
    514     manager = get_option("mode.data_manager")
    515     if manager == "block":

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\construction.py:641, in sanitize_array(data, index, dtype, copy, allow_2d)
    632     return sanitize_array(
    633         data,
    634         index=index,
   (...)
    637         allow_2d=allow_2d,
    638     )
    640 else:
--> 641     _sanitize_non_ordered(data)
    642     # materialize e.g. generators, convert e.g. tuples, abc.ValueView
    643     data = list(data)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\construction.py:692, in _sanitize_non_ordered(data)
    688 """
    689 Raise only for unordered sets, e.g., not for dict_keys
    690 """
    691 if isinstance(data, (set, frozenset)):
--> 692     raise TypeError(f"'{type(data).__name__}' type is unordered")

TypeError: 'set' type is unordered
```

In [74]:
```
# Creating a Series using dictionary:
d = pd.Series({1:'apple',2:'kiwi',3:'orange'})
```

In [75]:
```
d
```

Out[75]:
```
1     apple
2      kiwi
3    orange
dtype: object
```

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

In [2]:
```
d = pd.Series({'a':100,'b':200,'c':300,'d':400})
d
```

Out[2]:
```
a    100
b    200
c    300
d    400
dtype: int64
```

In [3]:
```
d1 =  pd.Series({'a':100,'b':200,'c':300,'d':400},
                index=['a','e','f','d'])
d1
```

Out[3]:
```
a    100.0
e      NaN
f      NaN
d    400.0
dtype: float64
```

In [4]:
```
d
```

Out[4]:
```
a    100
b    200
c    300
d    400
dtype: int64
```

In [5]:
```
# DataFrame:- multiple columns.
dt = pd.DataFrame({'Name':['Lily','Jasmin','Yuvraj','Tani'],
                   'Roll_No':[101,102,103,104],
                   'Courses':['AIML','Python','C','Java']})
```

In [6]:
```
dt
```

Out[6]:

	Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [7]:
```
type(dt)
```

Out[7]:
```
pandas.core.frame.DataFrame
```

In [8]:
```
# Accessing a specific columns value.
dt.Courses[3]
```

Out[8]:
```
'Java'
```

In [9]:
```
dt.Name[0]
```

Out[9]:
```
'Lily'
```

In [10]:
```
# head(): returns the first top 5 values of the dataframe
dt.head()
```

Out[10]:

	Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [11]:
```
# returns the first specified datas from dataframe.
dt.head(2)
```

Out[11]:

	Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python

In [12]:
```
# tail(): returns the last 5 values from dataframe.
dt.tail()
```

Out[12]:

	Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [13]:
```
dt.tail(3)
```

Out[13]:

	Name	Roll_No	Courses
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [14]:
```
# Slicing of Dataframe:
dt[::,::]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3791, in
Index.get_loc(self, key)
   3790 try:
-> 3791     return self._engine.get_loc(casted_key)
   3792 except KeyError as err:

File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:158, in pandas._libs.index.IndexEngine.get_loc()

TypeError: '(slice(None, None, None), slice(None, None, None))' is an invalid key

During handling of the above exception, another exception occurred:

InvalidIndexError                         Traceback (most recent call last)
Cell In[14], line 2
      1 # Slicing of Dataframe:
----> 2 dt[::,::]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3893, in DataFr
ame.__getitem__(self, key)
   3891 if self.columns.nlevels > 1:
   3892     return self._getitem_multilevel(key)
-> 3893 indexer = self.columns.get_loc(key)
   3894 if is_integer(indexer):
   3895     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3803, in
Index.get_loc(self, key)
   3798     raise KeyError(key) from err
   3799 except TypeError:
   3800     # If we have a listlike key, _check_indexing_error will raise
   3801     #  InvalidIndexError. Otherwise we fall through and re-raise
   3802     #  the TypeError.
-> 3803     self._check_indexing_error(key)
   3804     raise

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:5975, in
Index._check_indexing_error(self, key)
   5971 def _check_indexing_error(self, key):
   5972     if not is_scalar(key):
   5973         # if key is not a scalar, directly raise an error (the code below
   5974         # would convert to numpy arrays and raise later any way) - GH29926
-> 5975         raise InvalidIndexError(key)

InvalidIndexError: (slice(None, None, None), slice(None, None, None))
```

In [ ]:
```
# iloc: this helps to perform slicing on Dataframes.
dt
```

In [ ]:
```
dt.iloc[::,::]
```

In [ ]:
```
dt.iloc[::2,::]
```

In [15]:
```
dt.iloc[::,::2]
```

Out[15]:

	Name	Courses
0	Lily	AIML
1	Jasmin	Python
2	Yuvraj	C
3	Tani	Java

In [16]:
```
# returns all rows and only 2nd indexed column.
dt.iloc[::,2]
```

Out[16]:
```
0      AIML
1    Python
2         C
3      Java
Name: Courses, dtype: object
```

In [17]:
```
# Renaming a columns name:-
dt
```

Out[17]:

	Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [18]:
```
dt = dt.rename(columns={'Name':'Students_Name'})
dt
```

Out[18]:

	Students_Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [19]:
```
# Set Index:-
dt.set_index('Roll_No')
```

Out[19]:

Roll_No	Students_Name	Courses
101	Lily	AIML
102	Jasmin	Python
103	Yuvraj	C
104	Tani	Java

In [20]:
```
dt
```

Out[20]:

	Students_Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [21]:
```
dt1 = dt.set_index('Roll_No')
dt1
```

Out[21]:

Roll_No	Students_Name	Courses
101	Lily	AIML
102	Jasmin	Python
103	Yuvraj	C
104	Tani	Java

In [22]:
```
dt
```

Out[22]:

	Students_Name	Roll_No	Courses
0	Lily	101	AIML
1	Jasmin	102	Python
2	Yuvraj	103	C
3	Tani	104	Java

In [23]:
```
dt.set_index('Roll_No',inplace=True)
```

In [24]:
```
dt
```

Out[24]:

Roll_No	Students_Name	Courses
101	Lily	AIML
102	Jasmin	Python
103	Yuvraj	C
104	Tani	Java

In [27]:
```
dt.Students_Name[101]
```

Out[27]:
```
'Lily'
```

In [28]:
```
dt.Students_Name[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3791, in Index.get_loc(self, key)
   3790 try:
-> 3791     return self._engine.get_loc(casted_key)
   3792 except KeyError as err:

File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:2606, in pandas._libs.hashtable.Int64HashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:2630, in pandas._libs.hashtable.Int64HashTable.get_item()

KeyError: 0

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[28], line 1
----> 1 dt.Students_Name[0]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\series.py:1040, in Series.__getitem__(self, key)
   1037     return self._values[key]
   1039 elif key_is_scalar:
-> 1040     return self._get_value(key)
   1042 # Convert generator to list before going through hashable part
   1043 # (We will iterate through the generator there to check for slices)
   1044 if is_iterator(key):

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\series.py:1156, in Series._get_value(self, label, takeable)
   1153     return self._values[label]
   1155 # Similar to Index.get_value, but we do not fall back to positional
-> 1156 loc = self.index.get_loc(label)
   1158 if is_integer(loc):
   1159     return self._values[loc]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3798, in Index.get_loc(self, key)
   3793     if isinstance(casted_key, slice) or (
   3794         isinstance(casted_key, abc.Iterable)
   3795         and any(isinstance(x, slice) for x in casted_key)
   3796     ):
   3797         raise InvalidIndexError(key)
-> 3798     raise KeyError(key) from err
   3799 except TypeError:
   3800     # If we have a listlike key, _check_indexing_error will raise
   3801     #  InvalidIndexError. Otherwise we fall through and re-raise
   3802     #  the TypeError.
   3803     self._check_indexing_error(key)
```

```
KeyError: 0
```

In [32]:
```
# Working with Null Values:-
d = pd.DataFrame({'Days':[1,2,3,4,5,6],
                  'Places':['Odisha',np.nan,'Manali','Puri','Goa','Kashmir'],
                  'Visitors':[6000,2500,4600,4592,2350,1000]})
```

In [33]:
```
d
```

Out[33]:

	Days	Places	Visitors
0	1	Odisha	6000
1	2	NaN	2500
2	3	Manali	4600
3	4	Puri	4592
4	5	Goa	2350
5	6	Kashmir	1000

In [34]:
```
# to check missing/null values.
d.isnull()
```

Out[34]:

	Days	Places	Visitors
0	False	False	False
1	False	True	False
2	False	False	False
3	False	False	False
4	False	False	False
5	False	False	False

In [35]:
```
d.isnull().sum()
```

Out[35]:
```
Days        0
Places      1
Visitors    0
dtype: int64
```

In [36]:
```
# to replace the null value.
d.fillna('Hyderabad')
```

Out[36]:

	Days	Places	Visitors
0	1	Odisha	6000
1	2	Hyderabad	2500
2	3	Manali	4600
3	4	Puri	4592
4	5	Goa	2350
5	6	Kashmir	1000

In [37]:
```
d
```

Out[37]:

	Days	Places	Visitors
0	1	Odisha	6000
1	2	NaN	2500
2	3	Manali	4600
3	4	Puri	4592
4	5	Goa	2350
5	6	Kashmir	1000

In [39]:
```
# to replace the null values permanently.
d.fillna('Hyderabad',inplace=True)
d
```

Out[39]:

	Days	Places	Visitors
0	1	Odisha	6000
1	2	Hyderabad	2500
2	3	Manali	4600
3	4	Puri	4592
4	5	Goa	2350
5	6	Kashmir	1000

In [40]:
```
d
```

Out[40]:

	Days	Places	Visitors
0	1	Odisha	6000
1	2	Hyderabad	2500
2	3	Manali	4600
3	4	Puri	4592
4	5	Goa	2350
5	6	Kashmir	1000

In [41]:
```
d.isnull().sum()
```

Out[41]:
```
Days        0
Places      0
Visitors    0
dtype: int64
```

In [42]:
```
# Working with Null Values:-
d = pd.DataFrame({'Days':[1,2,3,4,5,6],
                  'Places':['Odisha',np.nan,'Manali','Puri',np.nan,'Kashmir'],
                  'Visitors':[6000,2500,4600,4592,2350,1000]})
```

In [43]:
```
d
```

Out[43]:

	Days	Places	Visitors
0	1	Odisha	6000
1	2	NaN	2500
2	3	Manali	4600
3	4	Puri	4592
4	5	NaN	2350
5	6	Kashmir	1000

In [44]:
```
d.isnull().sum()
```

Out[44]:
```
Days        0
Places      2
Visitors    0
dtype: int64
```

In [45]:
```
d.fillna('Bangalore',inplace=True)
d
```

Out[45]:

	Days	Places	Visitors
0	1	Odisha	6000
1	2	Bangalore	2500
2	3	Manali	4600
3	4	Puri	4592
4	5	Bangalore	2350
5	6	Kashmir	1000

In [47]:
```
# returns the datatype of every column in dataframe.
d.dtypes
```

Out[47]:
```
Days         int64
Places      object
Visitors     int64
dtype: object
```

In [49]:
```
# info(): returns all the information of our dataset.
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Days      6 non-null      int64 
 1   Places    6 non-null      object
 2   Visitors  6 non-null      int64 
dtypes: int64(2), object(1)
memory usage: 276.0+ bytes
```

In [50]:
```
# to find the unique values.
d.Places.unique()
```

Out[50]:
```
array(['Odisha', 'Bangalore', 'Manali', 'Puri', 'Kashmir'], dtype=object)
```

In [51]:
```
d.Visitors.unique()
```

Out[51]:
```
array([6000, 2500, 4600, 4592, 2350, 1000], dtype=int64)
```

In [52]:
```
# to find the average.
d.Visitors.mean()
```

Out[52]:
```
3507.0
```

In [54]:
```
d.Visitors.max()
```

Out[54]:
```
6000
```

In [56]:
```
# to find all the statistical values of our dataframe.
# describe(): returns statistical values of only integer cols.
d.describe()
```

Out[56]:

	Days	Visitors
count	6.000000	6.00000
mean	3.500000	3507.00000
std	1.870829	1856.05334
min	1.000000	1000.00000
25%	2.250000	2387.50000
50%	3.500000	3546.00000
75%	4.750000	4598.00000
max	6.000000	6000.00000

In [59]:
```
# returns the statistical values of only object col.
d.describe(include='O')
```

Out[59]:

	Places
count	6
unique	5
top	Bangalore
freq	2

In [61]:
```
# returns the statistical values of all the columns in dataset.
d.describe(include='all')
```

Out[61]:

	Days	Places	Visitors
count	6.000000	6	6.00000
unique	NaN	5	NaN
top	NaN	Bangalore	NaN
freq	NaN	2	NaN
mean	3.500000	NaN	3507.00000
std	1.870829	NaN	1856.05334
min	1.000000	NaN	1000.00000
25%	2.250000	NaN	2387.50000
50%	3.500000	NaN	3546.00000
75%	4.750000	NaN	4598.00000
max	6.000000	NaN	6000.00000

Working with Datasets:-

In [62]:
```
import numpy as np
import pandas as pd
```

In [63]:
```
# to read the dataset.
df = pd.read_csv(r"C:\Users\CTTC\Downloads\iris\iris.data")
df
```

Out[63]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

In the above code 👆 as our dataset doesnot contain the columns name, the 1st row data is being interpreted as column's name.

Therefore, inorder to overcome this we are using "header = None" as below 👇.

In [64]:
```
# to read the dataset.
df = pd.read_csv(r"C:\Users\CTTC\Downloads\iris\iris.data",header=None)
df
```

Out[64]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [ ]: