Experiment 1: Introduction To Numpy And Introduction To Pandas

1.1 Introduction To Numpy

NUMPY : NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

```
import numpy as np  #importing numpy into notebook
```

```
a = np.arange(15).reshape(3, 5)  #arranging numbers from 0-14 into 3*5 matrix
a.shape  #displaying shape of array
```

```
(3, 5)
```

```
 a.ndim   #no of dimensions of array
```

```
2
```

```
a.dtype.name  #data types of array
```

```
'int64'
```

```
 a.itemsize   #size of element in an array
```

```
8
```

```
a.size     #no of elements in an array
```

```
15
```

```
type(a)   #type of matrix
```

```
numpy.ndarray
```

```
 b = np.array([6, 7, 8])  #creating new array b
```

```
b  #printing array b
```

```
     array([6, 7, 8])
```

```
type(b)  #type of array
```

```
     numpy.ndarray
```

```
b.dtype  #datatype of b
```

```
     dtype('int64')
```

```
c = np.array([[1, 2], [3, 4]], dtype=complex)  #creating a new array and converting dataty
```

```
c   #printing array c
```

```
     array([[1.+0.j, 2.+0.j],
            [3.+0.j, 4.+0.j]])
```

```
c.dtype.name #datatype of c
```

```
     'complex128'
```

```
np.zeros((3, 4)) #creating a matrix of 3 rows and 4 columns with all elements 0
```

```
     array([[0., 0., 0., 0.],
            [0., 0., 0., 0.],
            [0., 0., 0., 0.]])
```

```
np.ones((2, 3, 4), dtype=np.int16)  #creating two arrays of 3*4 matrix containing all elem
```

```
     array([[[1, 1, 1, 1],
             [1, 1, 1, 1],
             [1, 1, 1, 1]],

            [[1, 1, 1, 1],
             [1, 1, 1, 1],
             [1, 1, 1, 1]]], dtype=int16)
```

```
np.empty((2, 3)) #creating an array of 2*3 with random numbers
```

```
     array([[4.6356993e-310, 0.0000000e+000, 0.0000000e+000],
            [0.0000000e+000, 0.0000000e+000, 0.0000000e+000]])
```

```
np.eye(7)  #creating an identity matrix of 7*7
```

```
     array([[1., 0., 0., 0., 0., 0., 0.],
            [0., 1., 0., 0., 0., 0., 0.],
            [0., 0., 1., 0., 0., 0., 0.],
            [0., 0., 0., 1., 0., 0., 0.],
            [0., 0., 0., 0., 1., 0., 0.],
            [0., 0., 0., 0., 0., 1., 0.],
            [0., 0., 0., 0., 0., 0., 1.]])
```

```
np.arange(10, 30, 5)  #arranges the given numbers with step of 5
```

```
array([10, 15, 20, 25])
```

```
np.arange(0, 2, 0.3)  #arranges the given numbers with step of 0.3
```

```
array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8])
```

```
np.linspace(0, 2, 9)  #9 equidistant values between 0-2
```

```
array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  ])
```

```
np.arange(0, 11, 1)**2  #arranging the square of numbers 0-10 in an array
```

```
array([  0,   1,   4,   9,  16,  25,  36,  49,  64,  81, 100])
```

```
print(a) #printing a matrix
```

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

```
print(a.reshape(5, 3))  #printing matrix after reshaping
```

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

```
print(np.arange(10000))
```

```
[   0    1    2 ... 9997 9998 9999]
```

```
A = np.array([[1, 1], [0, 1]])  #arithmetic operation on matrix A & B
B = np.array([[2, 0], [3, 4]])
```

```
A+B
```

```
array([[3, 1],
              [3, 5]])
```

```
A-B
```

```
     array([[-1,  1],
            [-3, -3]])
```

```
A*B
```

```
     array([[2, 0],
            [0, 4]])
```

```
A @ B
```

```
     array([[5, 4],
            [3, 4]])
```

```
A/B
```

```
     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: RuntimeWarning: divi
       """Entry point for launching an IPython kernel.
     array([[0.5 ,  inf],
            [0.  , 0.25]])
```

```
 np.sin(np.arange(0, 2 * np.pi, np.pi / 6))  #sin of values in matrix
```

```
     array([ 0.00000000e+00,  5.00000000e-01,  8.66025404e-01,  1.00000000e+00,
             8.66025404e-01,  5.00000000e-01,  1.22464680e-16, -5.00000000e-01,
            -8.66025404e-01, -1.00000000e+00, -8.66025404e-01, -5.00000000e-01])
```

```
A.dot(B)  #dot product of matrix
```

```
     array([[5, 4],
            [3, 4]])
```

```
a[:6:2]
```

```
     array([[ 0,  1,  2,  3,  4],
            [10, 11, 12, 13, 14]])
```

```
a[::-1]
```

```
     array([[10, 11, 12, 13, 14],
            [ 5,  6,  7,  8,  9],
            [ 0,  1,  2,  3,  4]])
```

```
a.T #transpose of matrix
```

```
array([[ 0,  5, 10],
       [ 1,  6, 11],
       [ 2,  7, 12],
       [ 3,  8, 13],
       [ 4,  9, 14]])
```

```
 A.trace() #sum of diagonal elements
```

```
2
```

```
A[0][[False, True]]
```

```
array([1])
```

1.2 Introduction To Pandas

```
import pandas as pd  #importing pandas in notebook
```

```
s = pd.Series([1,3, 5, np.nan, 6, 8])  #creates a new dataframe
```

```
s
```

```
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

```
dates = pd.date_range('20130101', periods=6)  #creates database of dates
```

```
dates
```

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')
```

```
df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))  #creating new
```

```
df
```

	A	B	C	D
2013-01-01	-0.897966	-0.085389	0.519369	-0.728660
2013-01-02	-0.897422	1.015018	-0.458587	-0.250734
2013-01-03	1.532031	-0.805968	-0.249865	1.543362
2013-01-04	-0.434921	-1.338483	-1.292154	-0.217500
2013-01-05	-0.375160	-0.183049	-0.422201	1.439585
2013-01-06	1.041061	1.153680	1.318065	1.411305

```
#creating a new dataframe
df2 = pd.DataFrame({
'A' : 1.0,
'B' : pd.Timestamp('20130102'),
'C' : pd.Series(1, index=list(range(4)), dtype='float32'),
'D' : np.array([3] * 4, dtype='int32'),
'E' : pd.Categorical(['test', 'train', 'test', 'train']),
'F' : 'foo'
})
```

```
df2
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

```
df2.dtypes  #datatypes of elements in database
```

```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

```
df.head(5)   #displaying first 5 rows of database
```

	A	B	C	D

```
df.tail(3)  #displaying last 3 rows of database
```

	A	B	C	D
2013-01-04	-0.434921	-1.338483	-1.292154	-0.217500
2013-01-05	-0.375160	-0.183049	-0.422201	1.439585
2013-01-06	1.041061	-1.152620	-1.318065	1.411205

```
df.index  #displaying index
```

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')
```

```
df.columns  #displaying column labels
```

```
Index(['A', 'B', 'C', 'D'], dtype='object')
```

```
df.to_numpy()  #arranges database as an array
```

```
array([[-0.89796604, -0.08538899,  0.51936876, -0.72865973],
       [-0.89742225,  1.01501784, -0.45858721, -0.25073413],
       [ 1.53203072, -0.80596786, -0.24986477,  1.54336186],
       [-0.43492127, -1.33848265, -1.29215393, -0.2175    ],
       [-0.37516031, -0.18304885, -0.42220108,  1.43958544],
       [ 1.0410614 , -1.1526203 , -1.31806496,  1.411205  ]])
```

```
df2.to_numpy()
```

```
array([[1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'train', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'train', 'foo']],
      dtype=object)
```

```
df.describe()  #displaying detailed statistics
```

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000

```
df.T  #transposes the database
```

	2013-01-01	2013-01-02	2013-01-03	2013-01-04	2013-01-05	2013-01-06
A	-0.897966	-0.897422	1.532031	-0.434921	-0.375160	1.041061
B	-0.085389	1.015018	-0.805968	-1.338483	-0.183049	-1.152620
C	0.519369	-0.458587	-0.249865	-1.292154	-0.422201	-1.318065
D	-0.728660	-0.250734	1.543362	-0.217500	1.439585	1.411205

```
df.sort_index(axis=1, ascending=False)
```

	D	C	B	A
2013-01-01	-0.728660	0.519369	-0.085389	-0.897966
2013-01-02	-0.250734	-0.458587	1.015018	-0.897422
2013-01-03	1.543362	-0.249865	-0.805968	1.532031
2013-01-04	-0.217500	-1.292154	-1.338483	-0.434921
2013-01-05	1.439585	-0.422201	-0.183049	-0.375160
2013-01-06	1.411205	-1.318065	-1.152620	1.041061

```
df.sort_values(by='B')  #sorting the databases according to ascending of column B
```

	A	B	C	D
2013-01-04	-0.434921	-1.338483	-1.292154	-0.217500
2013-01-06	1.041061	-1.152620	-1.318065	1.411205
2013-01-03	1.532031	-0.805968	-0.249865	1.543362
2013-01-05	-0.375160	-0.183049	-0.422201	1.439585
2013-01-01	-0.897966	-0.085389	0.519369	-0.728660
2013-01-02	-0.897422	1.015018	-0.458587	-0.250734

```
df['A']  #displaying column values of A
```

```
2013-01-01   -0.897966
2013-01-02   -0.897422
2013-01-03    1.532031
2013-01-04   -0.434921
2013-01-05   -0.375160
```

```
2013-01-06    1.041061
Freq: D, Name: A, dtype: float64
```

```
df[0:3]  #displaying columns in given range
```

	A	B	C	D
2013-01-01	-0.897966	-0.085389	0.519369	-0.728660
2013-01-02	-0.897422	1.015018	-0.458587	-0.250734
2013-01-03	1.532031	-0.805968	-0.249865	1.543362

```
df['20130102':'20130104']  #displaying data in given date range
```

	A	B	C	D
2013-01-02	-0.897422	1.015018	-0.458587	-0.250734
2013-01-03	1.532031	-0.805968	-0.249865	1.543362
2013-01-04	-0.434921	-1.338483	-1.292154	-0.217500

```
df.loc[dates[0]]  #values at index 0 in date columns
```

```
A   -0.897966
B   -0.085389
C    0.519369
D   -0.728660
Name: 2013-01-01 00:00:00, dtype: float64
```

```
df.loc[:, ["A", "B"]]  #displaying values of column A and B
```

	A	B
2013-01-01	-0.897966	-0.085389
2013-01-02	-0.897422	1.015018
2013-01-03	1.532031	-0.805968
2013-01-04	-0.434921	-1.338483
2013-01-05	-0.375160	-0.183049
2013-01-06	1.041061	-1.152620

```
df.at[dates[0], 'A']  #displaying values of column A at index 0
```

```
-0.8979660412249537
```

```
 df.iloc[3]
```

```
A   -0.434921
B   -1.338483
C   -1.292154
D   -0.217500
Name: 2013-01-04 00:00:00, dtype: float64
```

```
df[df['A'] > 0]  #display rows where A>0
```

	A	B	C	D
2013-01-03	1.532031	-0.805968	-0.249865	1.543362
2013-01-06	1.041061	-1.152620	-1.318065	1.411205

```
s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date_range("20130102", periods=6))  #create ne
```

```
s1
```

```
2013-01-02    1
2013-01-03    2
2013-01-04    3
2013-01-05    4
2013-01-06    5
2013-01-07    6
Freq: D, dtype: int64
```

```
df['F'] = s1
```

```
df
```

	A	B	C	D	F
2013-01-01	-0.897966	-0.085389	0.519369	-0.728660	NaN
2013-01-02	-0.897422	1.015018	-0.458587	-0.250734	1.0
2013-01-03	1.532031	-0.805968	-0.249865	1.543362	2.0
2013-01-04	-0.434921	-1.338483	-1.292154	-0.217500	3.0
2013-01-05	-0.375160	-0.183049	-0.422201	1.439585	4.0
2013-01-06	1.041061	-1.152620	-1.318065	1.411205	5.0

```
df.fillna(value=5)
```

	A	B	C	D	F
2013-01-01	-0.897966	-0.085389	0.519369	-0.728660	5.0
2013-01-02	-0.897422	1.015018	-0.458587	-0.250734	1.0
2013-01-03	1.532031	-0.805968	-0.249865	1.543362	2.0
2013-01-04	-0.434921	-1.338483	-1.292154	-0.217500	3.0

```
pd.isna(df)
```

	A	B	C	D	F
2013-01-01	False	False	False	False	True
2013-01-02	False	False	False	False	False
2013-01-03	False	False	False	False	False
2013-01-04	False	False	False	False	False
2013-01-05	False	False	False	False	False
2013-01-06	False	False	False	False	False

```
df.mean()  #displays mean value of columns
```

```
A   -0.005396
B   -0.425082
C   -0.536917
D    0.532876
F    3.000000
dtype: float64
```

```
df.mean(axis=1)
```

```
2013-01-01   -0.298161
2013-01-02    0.081655
2013-01-03    0.803912
2013-01-04   -0.056612
2013-01-05    0.891835
2013-01-06    0.996316
Freq: D, dtype: float64
```

```
df.apply(np.cumsum)
```

	A	B	C	D	F
2013-01-01	-0.897966	-0.085389	0.519369	-0.728660	NaN

```
df.apply(lambda x: x.max() - x.min())  #displaying the difference of max and min values in
```

```
A    2.429997
B    2.353500
C    1.837434
D    2.272022
F    4.000000
dtype: float64
```

```
s = pd.Series(["A", "B", "C", "Aaba", "Baca", np.nan, "CABA", "dog", "cat"])  #create new
```

```
s.str.lower()  #converts all strings to lower case
```

```
0       a
1       b
2       c
3    aaba
4    baca
5     NaN
6    caba
7     dog
8     cat
dtype: object
```

```
df = pd.DataFrame(np.random.randn(10, 4))
```

```
df
```

	0	1	2	3
0	0.588694	1.331366	1.207421	-0.270676
1	-0.036991	-0.367180	-0.098889	-0.391912
2	0.666960	1.131689	1.703514	-0.895154
3	0.566886	-0.103435	-0.062773	-0.304860
4	1.748775	-2.244922	1.071522	0.333791
5	0.911029	1.198571	2.054860	-0.791518
6	0.804183	-0.443459	0.196154	1.236847
7	-0.846512	0.804016	0.731116	-1.592932
8	0.203853	-0.326364	0.834151	0.381960
9	-0.335691	0.649013	-0.425492	0.560395

```
pieces = [df[:3], df[3:7], df[7:]]
```

```
pieces
```

```
[          0         1         2         3
 0  0.588694  1.331366  1.207421 -0.270676
 1 -0.036991 -0.367180 -0.098889 -0.391912
 2  0.666960  1.131689  1.703514 -0.895154,
           0         1         2         3
 3  0.566886 -0.103435 -0.062773 -0.304860
 4  1.748775 -2.244922  1.071522  0.333791
 5  0.911029  1.198571  2.054860 -0.791518
 6  0.804183 -0.443459  0.196154  1.236847,
           0         1         2         3
 7 -0.846512  0.804016  0.731116 -1.592932
 8  0.203853 -0.326364  0.834151  0.381960
 9 -0.335691  0.649013 -0.425492  0.560395]
```

```
pd.concat(pieces)
```

	0	1	2	3
0	0.588694	1.331366	1.207421	-0.270676
1	-0.036991	-0.367180	-0.098889	-0.391912
2	0.666960	1.131689	1.703514	-0.895154
3	0.566886	-0.103435	-0.062773	-0.304860
4	1.748775	-2.244922	1.071522	0.333791
5	0.911029	1.198571	2.054860	-0.791518
6	0.804183	-0.443459	0.196154	1.236847
7	-0.846512	0.804016	0.731116	-1.592932
8	0.203853	-0.326364	0.834151	0.381960
9	-0.335691	0.649013	-0.425492	0.560395

```
left = pd.DataFrame({"key": ["foo", "foo"], "lval": [1, 2]})
```

```
right = pd.DataFrame({"key": ["foo", "foo"], "rval": [4, 5]})
```

```
 pd.merge(left, right, on="key")
```

	key	lval	rval
0	foo	1	4

```
df.groupby(1).sum()
```

	0	2	3
1			
-2.244922	1.748775	1.071522	0.333791
-0.443459	0.804183	0.196154	1.236847
-0.367180	-0.036991	-0.098889	-0.391912
-0.326364	0.203853	0.834151	0.381960
-0.103435	0.566886	-0.062773	-0.304860
0.649013	-0.335691	-0.425492	0.560395
0.804016	-0.846512	0.731116	-1.592932
1.131689	0.666960	1.703514	-0.895154
1.198571	0.911029	2.054860	-0.791518
1.331366	0.588694	1.207421	-0.270676

```
df.sort_values(by=1)
```

	0	1	2	3
4	1.748775	-2.244922	1.071522	0.333791
6	0.804183	-0.443459	0.196154	1.236847
1	-0.036991	-0.367180	-0.098889	-0.391912
8	0.203853	-0.326364	0.834151	0.381960
3	0.566886	-0.103435	-0.062773	-0.304860
9	-0.335691	0.649013	-0.425492	0.560395
7	-0.846512	0.804016	0.731116	-1.592932
2	0.666960	1.131689	1.703514	-0.895154
5	0.911029	1.198571	2.054860	-0.791518
0	0.588694	1.331366	1.207421	-0.270676

✓ 0s completed at 3:50 PM ● ×