Day 12 of Training at Ansh Info Tech

Topics Covered

- **Numpy Library**
  - Indexing, Sorting, Slicing, Conditional Statements, Copying, Aggregate Functions
- **Pandas Library**
  - Series
    - Creation using Lists or Dictionary, Series Labels and Accessing Series Items
  - DataFrames
    - Creation, Accessing, Adding and Deleting Rows/Columns of DataFrame, Conditional Selection
- **Pandas Worksheet 1 Solved**

Summary

Numpy Library

Numpy provides efficient support for arrays and matrices in Python, enabling operations like indexing (selecting specific elements), sorting, and slicing (extracting subarrays). Conditional statements allow filtering based on certain criteria, while copying ensures data integrity. Aggregate functions compute summary statistics across arrays, vital for numerical analysis and data processing.

Pandas Library

Pandas simplifies data manipulation with powerful data structures:

- **Series**: One-dimensional labeled arrays can be created from lists or dictionaries. Accessing items by labels facilitates data retrieval and manipulation.
- **DataFrames**: Two-dimensional data structures akin to tables in a database. They support operations such as creating, accessing, and modifying rows and columns. Conditional selection allows filtering rows based on conditions, essential for data analysis and reporting.

Pandas Worksheet 1 Solved

The solved worksheet likely reinforced understanding of Pandas concepts covered, providing practical experience in using Series and DataFrames effectively for data manipulation tasks.

Double-click (or enter) to edit

Numpy Library

Indexing

```
random_arr = np.random.randint(1, 100, size=(10, 10))
random_arr
```

```
array([[64, 55, 86, 83, 43, 61, 10, 77, 71, 46],
       [21, 84, 71, 43, 64,  3,  2, 63,  8, 78],
       [21, 43, 11, 78, 58, 44, 38, 80, 93, 20],
       [48, 88, 63, 59, 73, 48, 84, 91,  5, 34],
       [59, 88, 63, 24, 62, 65, 15, 12, 52, 32],
       [30, 90, 79, 42, 95, 90, 90, 91, 68, 95],
       [ 2, 95, 12, 53, 71, 37, 70, 33, 12,  5],
       [74, 32, 54, 71, 75, 94, 23, 94, 27, 86],
       [61, 46, 98, 70, 13, 91, 58, 24, 69, 53],
       [80, 28, 92, 74, 25, 85, 33,  5,  5, 93]])
```

```
arr1
```

```
array([1, 2, 3, 4])
```

```
arr1[2]
```

```
3
```

```
random_arr[5]
```

```
array([30, 90, 79, 42, 95, 90, 90, 91, 68, 95])
```

```
random_arr[5][2]
```

```
79
```

```
random_arr[3:8]
```

```
array([[48, 88, 63, 59, 73, 48, 84, 91,  5, 34],
       [59, 88, 63, 24, 62, 65, 15, 12, 52, 32],
       [30, 90, 79, 42, 95, 90, 90, 91, 68, 95],
       [ 2, 95, 12, 53, 71, 37, 70, 33, 12,  5],
       [74, 32, 54, 71, 75, 94, 23, 94, 27, 86]])
```

Sorting Arrays

```
unsorted_arr = np.array([23, 10, 25, 45, 500, 200, 50])
unsorted_arr
```

```
array([ 23,  10,  25,  45, 500, 200,  50])
```

```
sorted(unsorted_arr)
```

```
[10, 23, 25, 45, 50, 200, 500]
```

```
unsorted_arr
```

```
array([ 23,  10,  25,  45, 500, 200,  50])
```

```
unsorted_arr.sort()
```

```
unsorted_arr
```

```
array([ 10,  23,  25,  45,  50, 200, 500])
```

Slicing vs Copying

```
unsorted_arr
```

```
array([ 10,  23,  25,  45,  50, 200, 500])
```

```
sliced_arr = unsorted_arr[2:6]
sliced_arr
```

```
array([ 25,  45,  50, 200])
```

```
sliced_arr[0] = 1000
```

```
sliced_arr
```

```
array([1000,   45,   50,  200])
```

```
unsorted_arr
```

```
array([  10,   23, 1000,   45,   50,  200,  500])
```

```
new_arr = unsorted_arr.copy()
new_arr
```

```
array([  10,   23, 1000,   45,   50,  200,  500])
```

```
new_arr[0] = -100
new_arr
```

```
array([-100,   23, 1000,   45,   50,  200,  500])
```

```
unsorted_arr
```

```
array([  10,   23, 1000,   45,   50,  200,  500])
```

Broadcasting

```
arr = np.arange(10)
arr
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
arr[4:8] = 100
```

```
arr
```

```
array([  0,   1,   2,   3, 100, 100, 100, 100,   8,   9])
```

```
arr * 10
```

```
array([   0,   10,   20,   30, 1000, 1000, 1000, 1000,   80,   90])
```

Conditional Statements

```
arr
```

```
array([  0,   1,   2,   3, 100, 100, 100, 100,   8,   9])
```

```
arr % 2 == 0
```

```
array([ True, False,  True, False,  True,  True,  True,  True,  True,
       False])
```

```
arr[arr % 2 == 0]
```

```
array([  0,   2, 100, 100, 100, 100,   8])
```

```
(arr % 2 == 0) & (arr > 10)
```

```
array([False, False, False, False,  True,  True,  True,  True, False,
       False])
```

```
arr[(arr % 2 == 0) & (arr > 10)]
```

```
array([100, 100, 100, 100])
```

Aggregate Functions

```
arr = np.random.randint(1, 100, size=10)
arr
```

```
array([ 5, 71,  6, 99, 12, 69, 44, 37, 93,  8])
```

```
arr.min()
```

```
5
```

```
arr.max()
```

```
99
```

```
arr.argmin()
```

```
0
```

```
arr.argmax()
```

```
3
```

```
arr.sum()
```

```
444
```

```
np.sqrt(arr)
```

```
array([2.23606798, 8.42614977, 2.44948974, 9.94987437, 3.46410162,
       8.30662386, 6.63324958, 6.08276253, 9.64365076, 2.82842712])
```

```
np.sin(arr)
```

```
array([-0.95892427,  0.95105465, -0.2794155 , -0.99920683, -0.53657292,
       -0.11478481,  0.01770193, -0.64353813, -0.94828214,  0.98935825])
```

```
import numpy as np
id1 = np.identity(4, dtype = int)
id1
```

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

```
mat2 = np.linspace(10,20, 100)
```

```
mat2
```

```
array([10.        , 10.1010101 , 10.2020202 , 10.3030303 , 10.4040404 ,
       10.50505051, 10.60606061, 10.70707071, 10.80808081, 10.90909091,
       11.01010101, 11.11111111, 11.21212121, 11.31313131, 11.41414141,
       11.51515152, 11.61616162, 11.71717172, 11.81818182, 11.91919192,
       12.02020202, 12.12121212, 12.22222222, 12.32323232, 12.42424242,
       12.52525253, 12.62626263, 12.72727273, 12.82828283, 12.92929293,
       13.03030303, 13.13131313, 13.23232323, 13.33333333, 13.43434343,
       13.53535354, 13.63636364, 13.73737374, 13.83838384, 13.93939394,
       14.04040404, 14.14141414, 14.24242424, 14.34343434, 14.44444444,
       14.54545455, 14.64646465, 14.74747475, 14.84848485, 14.94949495,
       15.05050505, 15.15151515, 15.25252525, 15.35353535, 15.45454545,
       15.55555556, 15.65656566, 15.75757576, 15.85858586, 15.95959596,
       16.06060606, 16.16161616, 16.26262626, 16.36363636, 16.46464646,
       16.56565657, 16.66666667, 16.76767677, 16.86868687, 16.96969697,
       17.07070707, 17.17171717, 17.27272727, 17.37373737, 17.47474747,
       17.57575758, 17.67676768, 17.77777778, 17.87878788, 17.97979798,
       18.08080808, 18.18181818, 18.28282828, 18.38383838, 18.48484848,
       18.58585859, 18.68686869, 18.78787879, 18.88888889, 18.98989899,
       19.09090909, 19.19191919, 19.29292929, 19.39393939, 19.49494949,
       19.5959596 , 19.6969697 , 19.7979798 , 19.8989899 , 20.        ])
```

Start coding or generate with AI.

Pandas

```
import numpy as np
import pandas as pd


l1 = [1,2,3,4]
l2 = ['a', 'b', 'c', 'd']
d1 = {'India':'Delhi', 'China': 'Beijing', 'France': 'Paris'}
s1 = pd.Series(l1)
s2 = pd.Series(l1, index = ['p', 'q', 'r', 's'])
s3 = pd.Series(d1)
s3.index.name = 'Country'
s3.name = 'Capital'
print(s1, '\n')
print(s2, '\n')
print(s3, '\n')\

print(s1[1])
print(s2['r'])
print(s2[2])
print(s3['China'])
```

```
0    1
1    2
2    3
3    4
dtype: int64

p    1
q    2
r    3
s    4
dtype: int64

Country
India       Delhi
China     Beijing
France      Paris
Name: Capital, dtype: object

2
3
3
Beijing
```

DataFrames

```
df1 = np.random.randint(low = 1, high = 100, size = (5, 4))
df2 = pd.DataFrame(df1)
print(type(df1), '\n')
print(type(df2))
```

```
<class 'numpy.ndarray'>

<class 'pandas.core.frame.DataFrame'>
    0   1   2   3
0  23  32   3  84
1  11   6  48  40
2  24  22  21  19
3  39  80  68  58
4  63  85  24  29
```

```
df2
```

	0	**1**	**2**	**3**
0	23	32	3	84
1	11	6	48	40
2	24	22	21	19
3	39	80	68	58
4	63	85	24	29

```
# Grab Rows
df2.loc[1:]
```

	0	**1**	**2**	**3**
1	11	6	48	40
2	24	22	21	19
3	39	80	68	58
4	63	85	24	29

```
#Grab Columns
df2[1]
```

```
0    32
1     6
2    22
3    80
4    85
Name: 1, dtype: int64
```

```
df2[[1,3]]
```

	1	3
0	32	84
1	6	40
2	22	19
3	80	58
4	85	29

```
df2.columns = ['A', 'B', 'C', 'D']
df2.index = ['a', 'b', 'c', 'd', 'e']
df2
```

	A	B	C	D
a	23	32	3	84
b	11	6	48	40
c	24	22	21	19
d	39	80	68	58
e	63	85	24	29

```
df2['B']
```

```
a    32
b     6
c    22
d    80
e    85
Name: B, dtype: int64
```

```
df2['Z'] = [1,2,3,4,5]
df2
```

	A	B	C	D	Z
a	23	32	3	84	1
b	11	6	48	40	2
c	24	22	21	19	3
d	39	80	68	58	4
e	63	85	24	29	5

```
df2.at['a', 'B']
```

```
32
```

```
df2.drop('Z', axis = 1)
```

	A	B	C	D
a	23	32	3	84
b	11	6	48	40
c	24	22	21	19
d	39	80	68	58
e	63	85	24	29

```
df2.drop('e', axis = 0)
```

	A	B	C	D	Z
a	23	32	3	84	1
b	11	6	48	40	2
c	24	22	21	19	3
d	39	80	68	58	4

```
df2.insert(2, 'E', [1,2,3,4,5])
df2
```

	A	B	E	C	D	Z
a	23	32	1	3	84	1
b	11	6	2	48	40	2

```
df2.loc['f'] = [15, 89, 93, 51, 162,24]
df2
```

	A	B	E	C	D	Z
a	23	32	1	3	84	1
b	11	6	2	48	40	2
c	24	22	3	21	19	3
d	39	80	4	68	58	4
e	63	85	5	24	29	5
5	15	89	93	51	162	24
f	15	89	93	51	162	24

```
df2
```

	A	B	E	C	D	Z