

Class #14: Introduction to Pandas - Part 1

1. Brief overview of Pandas and installation guide.
2. Introduction to Pandas library.
3. Pandas data structures: Series.

4. Introduction to Pandas DataFrames (excel/SQL).

Series: একমাত্রিক, একটি কলাম। DataFrame: দ্বিমাত্রিক, একাধিক কলাম। একাধিক Series নিয়ে DataFrame তৈরি হয়।

Python has four types of built-in array types: list, tuple, set, and dictionary. The third-party numeric array is provided by NumPy. Both NumPy and Pandas are non-built-in modules.

একটি 2D অ্যারে তৈরি এবং তার কিছু তথ্য প্রিন্ট

```
import pandas as pd
import numpy as np

# Create two string lists
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Print the index and column lists using f-string
print(f"Index List: {index1}")
print(f"Column List: {column1}")

# Create 2nd array
array_2d = np.arange(0, 100).reshape(10, 10)



# Print the 2D array using f-string
print(f"\n2D Array:\n{array_2d}")
```

Index List: ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']

Column List: ['c1', 'c2', 'c3', 'c4', 'c5', 'c6', 'c7', 'c8', 'c9', 'c10']

2D Array:

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```

1. **index1** এবং **column1** দুটি লিস্ট তৈরি করা হয়েছে, যেখানে প্রথমটি রো ইনডেক্স (r1 থেকে r10) এবং দ্বিতীয়টি কলাম নাম (c1 থেকে c10) ধারণ করে।
 **index1** নামের লিস্টে রো ইনডেক্সের মান (r1, r2, r3, ...) রাখা হয়েছে।
 **column1** নামের লিস্টে কলামের নাম (c1, c2, c3, ...) রাখা হয়েছে। এখানে `split()` ব্যবহার করে একক স্ট্রিংকে স্পেস দিয়ে আলাদা করা হয়েছে।
2. f-string ব্যবহার করে **index1** এবং **column1** লিস্টগুলো প্রিন্ট করা হয়েছে যাতে এটি আরও পরিষ্কারভাবে প্রদর্শিত হয়।
3. **array_2d** নামের একটি 2D অ্যারে তৈরি করা হয়েছে, যা 0 থেকে 99 পর্যন্ত সংখ্যার একটি 10x10 গ্রিড হিসেবে রূপান্তরিত হয়েছে।
4. f-string ব্যবহার করে এই 2D অ্যারের মানও প্রিন্ট করা হয়েছে।

index1 এবং **column1** হলো সাধারণ Python লিস্ট, কারণ এগুলো শুধুমাত্র স্ট্যান্ডার্ড Python ফিচার ব্যবহার করে তৈরি করা হয়েছে। অন্যদিকে, **array_2d** হলো একটি NumPy অ্যারে, কারণ এতে `np.arange()` এবং `.reshape()` মেথড ব্যবহার করা হয়েছে, যা NumPy লাইব্রেরির ফিচার। সুতরাং, যেকোনো ভেরিয়েবল যা `np` বা `pd` ধারণ করে, তা NumPy বা Pandas এর ডেটা স্ট্রাকচার।

```
import pandas as pd
import numpy as np

# Create two string lists
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create 2nd array
array_2d = np.arange(0, 100).reshape(10, 10)

# Let's create our first DataFrame using array_2d
df = pd.DataFrame(data=array_2d)
print(f"First DataFrame:\n{df}")

# Let's create our first DataFrame using index, columns, and array_2d
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)
print(f"\nSecond DataFrame with custom index and columns:\n{df}")
```

First DataFrame:

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	10	11	12	13	14	15	16	17	18	19
2	20	21	22	23	24	25	26	27	28	29
3	30	31	32	33	34	35	36	37	38	39
4	40	41	42	43	44	45	46	47	48	49
5	50	51	52	53	54	55	56	57	58	59
6	60	61	62	63	64	65	66	67	68	69

```

7  70  71  72  73  74  75  76  77  78  79
8  80  81  82  83  84  85  86  87  88  89
9  90  91  92  93  94  95  96  97  98  99

```

Second DataFrame with custom index and columns:

```

      c1  c2  c3  c4  c5  c6  c7  c8  c9  c10
r1      0   1   2   3   4   5   6   7   8   9
r2     10  11  12  13  14  15  16  17  18  19
r3     20  21  22  23  24  25  26  27  28  29
r4     30  31  32  33  34  35  36  37  38  39
r5     40  41  42  43  44  45  46  47  48  49
r6     50  51  52  53  54  55  56  57  58  59
r7     60  61  62  63  64  65  66  67  68  69
r8     70  71  72  73  74  75  76  77  78  79
r9     80  81  82  83  84  85  86  87  88  89
r10    90  91  92  93  94  95  96  97  98  99

```

দুটি DataFrame তৈরি করছে:

1. প্রথম DataFrame:

- array_2d ব্যবহার করে একটি DataFrame তৈরি করা হয়েছে, যেখানে ইনডেক্স এবং কলাম নির্ধারণ করা হয়নি।

2. দ্বিতীয় DataFrame:

- এই DataFrame-এ array_2d সহ কাস্টম ইনডেক্স (index1) এবং কাস্টম কলাম নাম (column1) নির্ধারণ করা হয়েছে।

প্রথম DataFrame শুধুমাত্র ডেটা দেখাবে, আর দ্বিতীয় DataFrame কাস্টম ইনডেক্স এবং কলাম নাম সহ দেখাবে। f-string ব্যবহার করে DataFrame গুলো প্রিন্ট করা হয়েছে।




নিচে macOS, Linux OS, এবং Windows OS এর মধ্যে পার্থক্য ছকে(DataFrame) দেওয়া হলো:

বিষয়	macOS (columns)	Linux OS (columns)	Windows OS (columns)
উৎপাদক (index)	Apple Inc.	ওপেন সোর্স, বিভিন্ন ডেভেলপার দ্বারা তৈরি	Microsoft
লাইসেন্স (index)	বাণিজ্যিক লাইসেন্স	ওপেন সোর্স (GNU GPL)	বাণিজ্যিক লাইসেন্স
সিকিউরিটি (index)	উচ্চ সিকিউরিটি, Apple এর নিজস্ব নিরাপত্তা	উচ্চ সিকিউরিটি, কমিউনিটি সাপোর্টে সুরক্ষিত	সাধারণত সিকিউরিটি কম, তবে নিয়মিত আপডেট
কাস্টমাইজেশন (index)	সীমিত কাস্টমাইজেশন	খুব বেশি কাস্টমাইজেশন সুবিধা	কিছুটা কাস্টমাইজেশন, তবে থার্ড- পার্টি সফটওয়্যার দিয়ে পরিবর্তন
হার্ডওয়্যার সাপোর্ট (index)	শুধুমাত্র Apple ডিভাইস (MacBook, iMac)	অনেক ধরনের হার্ডওয়্যার সাপোর্ট করে	ব্যাপক হার্ডওয়্যার সাপোর্ট

dataframe হতে হলে, data, row (row কে index বলে seriesএ), column থাকতে হয়। Data frame: row, column, data থাকবে।



Columns:

একটি **DataFrame** তৈরি এবং বিভিন্ন উপায়ে কলাম **ACCESS/PRINT** করে তা প্রদর্শন

code	output
<pre>import pandas as pd import numpy as np # Create two string lists index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10'] column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split() # Create 2nd array array_2d = np.arange(0, 100).reshape(10, 10) # Create DataFrame using index, columns, and array_2d df = pd.DataFrame(data=array_2d, index=index1, columns=column1) # Print selected columns with f-strings print(f'Accessing 'c1' column:\n{df['c1']}') print(f'\nAccessing 'c1' column as DataFrame:\n{df[['c1']]}) print(f'Type of df['c1']: {type(df['c1'])}') # Output: pandas Series # Selecting multiple columns ('c1' and 'c10') print(f'\nDataFrame of columns 'c1' and 'c10':\n{df[['c1', 'c10']]})</pre>	<pre>Accessing 'c1' column: r1 0 r2 10 r3 20 r4 30 r5 40 r6 50 r7 60 r8 70 r9 80 r10 90 Name: c1, dtype: int32 Accessing 'c1' column as DataFrame: c1 r1 0 r2 10 r3 20 r4 30 r5 40 r6 50 r7 60 r8 70 r9 80 r10 90 Type of df['c1']: <class 'pandas.core.series.Series'> DataFrame of columns 'c1' and 'c10': c1 c10 r1 0 9 r2 10 19 r3 20 29 r4 30 39 r5 40 49 r6 50 59 r7 60 69 r8 70 79 r9 80 89 r10 90 99</pre>
<ol style="list-style-type: none"> DataFrame তৈরি: index1 এবং column1 লিস্ট এবং array_2d দিয়ে একটি DataFrame তৈরি করা হয়েছে। এখানে রো ইনডেক্স এবং কলাম নাম কাস্টমাইজ করা হয়েছে। কলাম এক্সেস: <ul style="list-style-type: none">  df['c1']: এটি c1 কলামটি Series হিসেবে প্রিন্ট করবে।  df[['c1']]: এটি c1 কলামটি DataFrame আকারে প্রিন্ট করবে।  type(df['c1']): এটি c1 কলামের ডেটা টাইপ দেখাবে, যা একটি pandas Series। একাধিক কলাম নির্বাচন: df[['c1', 'c10']] দিয়ে c1 এবং c10 কলামগুলোকে DataFrame আকারে নির্বাচন করে প্রিন্ট করা হয়েছে। 	

Adding new column & Deleting the column -- drop()

একটি **DataFrame** তৈরি করছে এবং তার উপর কিছু কলাম মুছে ফেলার কার্যক্রম

Code	Output
<pre>import pandas as pd import numpy as np # Create two string lists for the index and columns index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10'] column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split() # Create a 10x10 array of numbers from 0 to 99 array_2d = np.arange(0, 100).reshape(10, 10) # Create DataFrame using the generated 2D array and the index and columns df = pd.DataFrame(data=array_2d, index=index1, columns=column1) # Add a new column 'c11' to the DataFrame df['c11'] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0] # Print the original DataFrame with the new 'c11' column print(f"Original DataFrame with 'c11' column:\n{df}") # Drop 'c11' column without affecting the original DataFrame df_without_c11 = df.drop('c11', axis=1) print(f"\nDataFrame after dropping 'c11' column (not in- place):\n{df_without_c11}") # Print the original DataFrame to confirm it is unchanged print(f"\nOriginal DataFrame (unchanged):\n{df}") # Drop 'c11' column in-place, modifying the original DataFrame df.drop('c11', axis=1, inplace=True) print(f"\nDataFrame after dropping 'c11' column (in- place):\n{df}")</pre>	<p>Original DataFrame with 'c11' column:</p> <pre>c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 r1 0 1 2 3 4 5 6 7 8 9 1 r2 10 11 12 13 14 15 16 17 18 19 2 r3 20 21 22 23 24 25 26 27 28 29 3 r4 30 31 32 33 34 35 36 37 38 39 4 r5 40 41 42 43 44 45 46 47 48 49 5 r6 50 51 52 53 54 55 56 57 58 59 6 r7 60 61 62 63 64 65 66 67 68 69 7 r8 70 71 72 73 74 75 76 77 78 79 8 r9 80 81 82 83 84 85 86 87 88 89 9 r10 90 91 92 93 94 95 96 97 98 99 0</pre> <p>DataFrame after dropping 'c11' column (not in- place):</p> <pre>c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 r1 0 1 2 3 4 5 6 7 8 9 r2 10 11 12 13 14 15 16 17 18 19 r3 20 21 22 23 24 25 26 27 28 29 r4 30 31 32 33 34 35 36 37 38 39 r5 40 41 42 43 44 45 46 47 48 49 r6 50 51 52 53 54 55 56 57 58 59 r7 60 61 62 63 64 65 66 67 68 69 r8 70 71 72 73 74 75 76 77 78 79 r9 80 81 82 83 84 85 86 87 88 89 r10 90 91 92 93 94 95 96 97 98 99</pre> <p>Original DataFrame (unchanged):</p> <pre>c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 r1 0 1 2 3 4 5 6 7 8 9 1 r2 10 11 12 13 14 15 16 17 18 19 2 r3 20 21 22 23 24 25 26 27 28 29 3 r4 30 31 32 33 34 35 36 37 38 39 4 r5 40 41 42 43 44 45 46 47 48 49 5 r6 50 51 52 53 54 55 56 57 58 59 6 r7 60 61 62 63 64 65 66 67 68 69 7 r8 70 71 72 73 74 75 76 77 78 79 8 r9 80 81 82 83 84 85 86 87 88 89 9 r10 90 91 92 93 94 95 96 97 98 99 0</pre> <p>DataFrame after dropping 'c11' column (in- place):</p> <pre>c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 r1 0 1 2 3 4 5 6 7 8 9 r2 10 11 12 13 14 15 16 17 18 19 r3 20 21 22 23 24 25 26 27 28 29 r4 30 31 32 33 34 35 36 37 38 39 r5 40 41 42 43 44 45 46 47 48 49 r6 50 51 52 53 54 55 56 57 58 59</pre>
<ol style="list-style-type: none"> DataFrame তৈরি:  index1 এবং column1 দিয়ে ১০x১০ এর একটি DataFrame তৈরি করা হয়েছে, যার মধ্যে ০ থেকে ৯৯ পর্যন্ত সংখ্যাগুলি রাখা হয়েছে। নতুন কলাম যোগ করা:  c11 নামক একটি নতুন কলাম df এ যোগ করা হয়েছে, যার মান ১ থেকে ৯ এবং ০ পর্যন্ত দেওয়া হয়েছে। List value insert করে panda তে dataframe বানাতে হয়। কলাম মুছে ফেলা: 	

- df.drop('c11', axis=1) কলামটি নতুন DataFrame তে মুছে ফেলে (এটি মূল DataFrame পরিবর্তন করবে না) এবং df_without_c11 তে সংরক্ষণ করা হয়েছে।
- তারপর, মূল DataFrame df মুদ্রণ করা হয়েছে যাতে দেখা যায়, এটি অপরিবর্তিত রয়েছে।
- পরিশেষে, df.drop('c11', axis=1, inplace=True) ব্যবহার করে মূল DataFrame থেকে c11 কলামটি সরানো হয়েছে (এটি সরাসরি পরিবর্তন করে)। inplace দিয়ে মূল dataframe এর column permanent delete করে।

```
r7 60 61 62 63 64 65 66 67 68 69
r8 70 71 72 73 74 75 76 77 78 79
r9 80 81 82 83 84 85 86 87 88 89
r10 90 91 92 93 94 95 96 97 98 99
```

df.drop('c11', axis=1, inplace=True)
axis=1: axis আর্গুমেন্টটি নির্ধারণ করে যে আপনি রো (row) না কলাম (column) মুছে ফেলতে চান।

axis=0 দিলে ROW মুছে যাবে (ডিফল্ট মান), কিন্তু এখানে axis=1 দেওয়া হয়েছে, অর্থাৎ COLUMN মুছে ফেলা হবে।

Rows:

একটি DataFrame তৈরি এবং বিভিন্ন উপায়ে রো ACCESS/PRINT করে তা প্রদর্শন. LOCation (loc) আর Index LOCation (iloc). loc iloc শুধুমাত্র dataframe এর row কাজ করে। series এ কাজ করে না।

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)
print(f"Original DataFrame:\n{df}")

# Access row 'r4' by label using loc
print(f"\nRow 'r4' using loc:\n{df.loc['r4']}")

# Access row 'r4' as DataFrame using loc
print(f"\nRow 'r4' as DataFrame using loc:\n{df.loc[['r4']]}")

# Access row 4 by position using iloc
print(f"\nRow 4 (index 4) using iloc:\n{df.iloc[4]}")

# Access row 4 as DataFrame using iloc
```

```
print(f"\nRow 4 (index 4) as DataFrame using
iloc:\n{df.iloc[[4]]}")

# Access multiple rows using loc
print(f"\nRows 'r1', 'r2', and 'r3' using loc:\n{df.loc[['r1',
'r2', 'r3']]}")
```

Original DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29
r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

Row 'r4' using loc:

c1	30
c2	31
c3	32
c4	33
c5	34
c6	35
c7	36
c8	37
c9	38
c10	39

Name: r4, dtype: int32

Row 'r4' as DataFrame using loc:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r4	30	31	32	33	34	35	36	37	38	39

Row 4 (index 4) using iloc:

c1	40
c2	41
c3	42
c4	43
c5	44
c6	45
c7	46
c8	47
c9	48
c10	49

Name: r5, dtype: int32

Row 4 (index 4) as DataFrame using iloc:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r5	40	41	42	43	44	45	46	47	48	49

Rows 'r1', 'r2', and 'r3' using loc:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29

1. DataFrame তৈরি:

- index1 এবং column1 দিয়ে একটি 10x10 এর DataFrame তৈরি করা হয়েছে। এতে 0 থেকে 99 পর্যন্ত সংখ্যা রয়েছে।

2. রো অ্যাক্সেস করা:

- loc: এটি লেবেল অনুযায়ী রো অ্যাক্সেস করতে ব্যবহৃত হয়। Location (LOC)
 - df.loc['r4'] → রো 'r4' এর ডাটা দেখায়।
 - df.loc[['r4']] → রো 'r4' কে একটি DataFrame হিসেবে দেখায়।
 - df.loc[['r1', 'r2', 'r3']] → একাধিক রো (r1, r2, r3) একসাথে দেখায়।
- iloc: এটি পজিশন অনুযায়ী রো অ্যাক্সেস করতে ব্যবহৃত হয়। Index Location (ILOC)
 - df.iloc[4] → 8 নম্বর পজিশনের (অর্থাৎ, 5ম রো) ডাটা দেখায়।
 - df.iloc[[4]] → 8 নম্বর পজিশনের রো একটি DataFrame হিসেবে দেখায়।

Grabbing an element or a sub-set of the dataframe:

একটি DataFrame তৈরি এবং সেটিতে বিভিন্ন রো এবং কলাম অ্যাক্সেস করার পদ্ধতি:

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)
print(f"Original DataFrame:\n{df}")

# Access a specific element using loc (row 'r1', column 'c1')
print(f"\nElement at row 'r1' and column 'c1': {df.loc['r1', 'c1']}")

# Access a subset of rows and columns using loc
print(f"\nSubset of rows 'r1' and 'r2' with columns 'c1' and 'c2':\n{df.loc[['r1', 'r2'], ['c1', 'c2']]}")

# Access another subset with different rows and columns
```



```
print(f"\nSubset of rows 'r2' and 'r5' with columns 'c3' and 'c4':\n{df.loc[['r2', 'r5'], ['c3', 'c4']]}")
```

Original DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29
r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

Element at row 'r1' and column 'c1': 0

Subset of rows 'r1' and 'r2' with columns 'c1' and 'c2':

	c1	c2
r1	0	1
r2	10	11

Subset of rows 'r2' and 'r5' with columns 'c3' and 'c4':

	c3	c4
r2	12	13
r5	42	43

1. DataFrame তৈরি:

- index1 এবং column1 নামের লিস্ট ব্যবহার করে একটি ১০x১০ DataFrame তৈরি হয়েছে, যেখানে 0 থেকে 99 পর্যন্ত মান রয়েছে।

2. loc ব্যবহার করে একক উপাদান অ্যাক্সেস:

- df.loc['r1', 'c1'] → r1 রো এবং c1 কলামের মান (যা 0) বের করা হয়েছে।

3. loc ব্যবহার করে রো ও কলামের সাবসেট অ্যাক্সেস:

- 🧩 df.loc[['r1', 'r2'], ['c1', 'c2']]: এটি r1 এবং r2 রো-এর জন্য c1 এবং c2 কলামগুলোর মান দেখায়।
→ সেটের চিহ্ন: $(r1 \cup r2) \cap (c1 \cup c2)$
- 🧩 df.loc[['r2', 'r5'], ['c3', 'c4']]: এটি r2 এবং r5 রো-এর জন্য c3 এবং c4 কলামগুলোর মান দেখায়।
→ সেটের চিহ্ন: $(r2 \cup r5) \cap (c3 \cup c4)$

This is similar to NumPy boolean mask, lets try this:

```
*bool_mask = df % 3 == 0
```

```
*df[bool_mask]
```

returns values where it is True and NaN where False. এই কোডটি একটি DataFrame তৈরি করে এবং বিভিন্ন শর্তে ডেটা নির্বাচন করে।

```
import pandas as pd
import numpy as np
```

```
# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)
print(f"Original DataFrame:\n{df}")

# Conditional selection: print values greater than 5
print(f"\nConditional Selection (values greater than 5):\n{df > 5}")

# Mask to find elements divisible by 3
bool_mask = (df % 3 == 0)
# Print elements divisible by 3
print(f"\nElements divisible by 3:\n{df[bool_mask]}")
```

Original DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29
r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

Conditional Selection (values greater than 5):

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	False	False	False	False	False	False	True	True	True	True
r2	True	True	True	True	True	True	True	True	True	True
r3	True	True	True	True	True	True	True	True	True	True
r4	True	True	True	True	True	True	True	True	True	True
r5	True	True	True	True	True	True	True	True	True	True
r6	True	True	True	True	True	True	True	True	True	True
r7	True	True	True	True	True	True	True	True	True	True

r8	True	True	True	True	True	True	True	True	True	True
True										
r9	True	True	True	True	True	True	True	True	True	True
True										
r10	True	True	True	True	True	True	True	True	True	True
True										

Elements divisible by 3:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0.0	NaN	NaN	3.0	NaN	NaN	6.0	NaN	NaN	9.0
r2	NaN	NaN	12.0	NaN	NaN	15.0	NaN	NaN	18.0	NaN
r3	NaN	21.0	NaN	NaN	24.0	NaN	NaN	27.0	NaN	NaN
r4	30.0	NaN	NaN	33.0	NaN	NaN	36.0	NaN	NaN	39.0
r5	NaN	NaN	42.0	NaN	NaN	45.0	NaN	NaN	48.0	NaN
r6	NaN	51.0	NaN	NaN	54.0	NaN	NaN	57.0	NaN	NaN
r7	60.0	NaN	NaN	63.0	NaN	NaN	66.0	NaN	NaN	69.0
r8	NaN	NaN	72.0	NaN	NaN	75.0	NaN	NaN	78.0	NaN
r9	NaN	81.0	NaN	NaN	84.0	NaN	NaN	87.0	NaN	NaN
r10	90.0	NaN	NaN	93.0	NaN	NaN	96.0	NaN	NaN	99.0

1. DataFrame তৈরি করা:

প্রথমে, একটি 10x10 আকারের array_2d তৈরি করা হয়েছে, যেখানে 0 থেকে 99 পর্যন্ত সংখ্যা রাখা হয়েছে। তারপর সেই ডেটা ব্যবহার করে df নামে একটি DataFrame তৈরি করা হয়েছে। এর ইনডেক্সে r1 থেকে r10 এবং কলামে c1 থেকে c10 নামক কলাম রাখা হয়েছে।

2. শর্তসাপেক্ষ নির্বাচন:

- df > 5 শর্তের মাধ্যমে DataFrame থেকে এমন সব মান নির্বাচন করা হয়েছে যা 5-এর বেশি।
- df % 3 == 0 শর্তের মাধ্যমে DataFrame থেকে এমন সব মান নির্বাচন করা হয়েছে যা 3 দ্বারা বিভাজ্য।

3. ফলাফল প্রদর্শন:

- প্রথমে df > 5 এর ফলাফল (একটি বুলিয়ান DataFrame) প্রিন্ট করা হয়েছে, যেখানে True মান দেখানো হবে যেগুলি 5-এর বেশি।
- এরপর, df[bool_mask] এর মাধ্যমে 3 দ্বারা বিভাজ্য উপাদানগুলো প্রিন্ট করা হয়েছে।

এভাবে, কোডটি DataFrame থেকে শর্তাবলীতে মান নির্বাচন এবং প্রদর্শন করার কাজটি করে।

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)
```

```
# Create DataFrame using the generated 2D array and the index and
columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)

# Print the original DataFrame
print(f"Original DataFrame:\n{df}")

boolean_mask = df['c1'] > 11
# Check which values in 'c1' column are greater than 11
print(f"\nBoolean mask (values in 'c1' > 11):\n{boolean_mask}")

# print all rows values where value of row is true above
print(f"\nTrue value of rows where only 'c1' >
11:\n{df[boolean_mask]}")

# Filter the same rows but only show the 'c1' column
boolean_mask = df[df['c1']>11]
print(f"\nFiltered 'c1' column where 'c1' >
11:\n{boolean_mask['c1']}")

# Alternatively, chaining conditions: 'c1'> 11 and then selecting
the 'c1' column
print(f"\nChained condition: Rows where 'c1' > 11 and showing 'c1'
values:\n{df[df['c1'] > 11]['c1']}")
```

Original DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29
r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

Boolean mask (values in 'c1' > 11):

r1	False
r2	False
r3	True
r4	True
r5	True
r6	True
r7	True
r8	True
r9	True
r10	True

Name: c1, dtype: bool

True value of rows where only 'c1' > 11:

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
----	----	----	----	----	----	----	----	----	-----

```

r3    20  21  22  23  24  25  26  27  28  29
r4    30  31  32  33  34  35  36  37  38  39
r5    40  41  42  43  44  45  46  47  48  49
r6    50  51  52  53  54  55  56  57  58  59
r7    60  61  62  63  64  65  66  67  68  69
r8    70  71  72  73  74  75  76  77  78  79
r9    80  81  82  83  84  85  86  87  88  89
r10   90  91  92  93  94  95  96  97  98  99

```

Filtered 'c1' column where 'c1' > 11:

```

r3      20
r4      30
r5      40
r6      50
r7      60
r8      70
r9      80
r10     90

```

Name: c1, dtype: int32

Chained condition: Rows where 'c1' > 11 and showing 'c1' values:

```

r3      20
r4      30
r5      40
r6      50
r7      60
r8      70
r9      80
r10     90

```

Name: c1, dtype: int32

1. **DataFrame তৈরি:** প্রথমে r1 থেকে r10 পর্যন্ত ১০টি রো এবং c1 থেকে c10 পর্যন্ত ১০টি কলাম দিয়ে একটি ১০x১০ আকারের DataFrame তৈরি করা হয়েছে। এখানে ০ থেকে ৯৯ পর্যন্ত সংখ্যা ব্যবহার করা হয়েছে।
2. **Boolean Mask তৈরি:**
 - `df['c1'] > 11` একটি শর্ত, যেখানে c1 কলামের যে মানগুলো ১১ এর চেয়ে বেশি, সেগুলোর জন্য True এবং বাকি গুলোর জন্য False হবে।
 - এই শর্তটি একটি boolean_mask হিসাবে সংরক্ষিত হয়েছে।
3. **যে রোতে 'c1' > 11 তা প্রিন্ট করা:**
 - `df[boolean_mask]` এর মাধ্যমে শুধুমাত্র সেই রো গুলো প্রিন্ট করা হচ্ছে, যেখানে c1 কলামের মান ১১ এর চেয়ে বেশি।
4. **'c1' কলামের মান যেখানে 'c1' > 11:**
 - `df[df['c1'] > 11]['c1']` ব্যবহার করে শুধুমাত্র c1 কলামটি দেখানো হচ্ছে, যেখানে c1 এর মান ১১ এর বেশি।
5. **Chained condition:**

- একই শর্তে আরও একটি কন্ডিশন চেইন করা হয়েছে `df[df['c1'] > 11]['c1']`, যাতে আবারও শুধু `c1` কলামের মান গুলো দেখা যায় যেখানে `'c1' > 11`।

```
boolean_mask = df['c1'] > 11
```

```
df[boolean_mask]. True যদি DataFrame এ দেয়া হয়, তাহলে সেগুলো মান দেখায় ।
```

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)

# Create a boolean mask for 'c1' values greater than 11
boolean_mask = df['c1'] > 11

# Print the boolean mask (True for values greater than 11 in 'c1')
print(f"\nBoolean mask (values in 'c1' > 11):\n{boolean_mask}")

# Print rows where the boolean mask is True
print(f"\nRows where 'c1' > 11:\n{df[boolean_mask]}")

# Print selected columns 'c1' and 'c9' where 'c1' > 11
print(f"\nRows where 'c1' > 11 with columns 'c1' and 'c9':\n{df[boolean_mask][['c1', 'c9']]}")
```

```
Boolean mask (values in 'c1' > 11):
```

```
r1      False
r2      False
r3       True
r4       True
r5       True
r6       True
r7       True
r8       True
r9       True
r10      True
```

```
Name: c1, dtype: bool
```

```
Rows where 'c1' > 11:
```

```
      c1  c2  c3  c4  c5  c6  c7  c8  c9  c10
r3    20  21  22  23  24  25  26  27  28  29
```

r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

Rows where 'c1' > 11 with columns 'c1' and 'c9':

	c1	c9
r3	20	28
r4	30	38
r5	40	48
r6	50	58
r7	60	68
r8	70	78
r9	80	88
r10	90	98

ব্যাখ্যা:

1. **DataFrame তৈরি:**

- index1 এবং column1 ব্যবহার করে ১০x১০ আকারের DataFrame তৈরি করা হয়েছে, যার মধ্যে ০ থেকে ৯৯ পর্যন্ত সংখ্যাগুলি রয়েছে।

2. **Boolean Mask তৈরি:**

- df['c1'] > 11 শর্তের মাধ্যমে, 'c1' কলামের যেসব মান ১১ এর বেশি, তাদের জন্য True এবং বাকি সবগুলোর জন্য False একটি Boolean mask তৈরি করা হয়েছে।

3. **Boolean Mask প্রিন্ট:**

- boolean_mask প্রিন্ট করে দেখানো হয়েছে, যাতে প্রত্যেক রো এর জন্য 'c1' কলামের মান ১১ এর চেয়ে বড় হলে তা True হবে।

4. **Shaping the Data:**

- df[boolean_mask] ব্যবহার করে সেই সব রো গুলি নির্বাচন করা হয়েছে যেগুলোর 'c1' এর মান ১১ এর চেয়ে বড়।

5. **নির্বাচিত কলাম সহ ফিল্টার করা:**

- df[boolean_mask][['c1', 'c9']] ব্যবহার করে শুধুমাত্র 'c1' এবং 'c9' কলামগুলো দেখানো হয়েছে, যেখানে 'c1' এর মান ১১ এর চেয়ে বড়।

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
```

```

array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and
columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)

# Print the original DataFrame
print(f"Original DataFrame:\n{df}")

# Check where 'c1' column is equal to 70
print(f"\nRows where 'c1' is equal to 70:\n{df['c1'] == 70}")

# Print rows where 'c1' is equal to 70
print(f"\nRows where 'c1' equals 70:\n{df[df['c1'] == 70]}")

# Boolean mask for 'c1' values greater than 11
boolean_mask = df['c1'] > 11

# Print rows where 'c1' > 11 and the index is 'r3' or 'r5'
print(f"\nRows where 'c1' > 11 and index is 'r3' or
'r5':\n{df[boolean_mask].loc[['r3', 'r5']]}")

```

Original DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29
r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

Rows where 'c1' is equal to 70:

r1	False
r2	False
r3	False
r4	False
r5	False
r6	False
r7	False
r8	True
r9	False
r10	False

Name: c1, dtype: bool

Rows where 'c1' equals 70:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r8	70	71	72	73	74	75	76	77	78	79

Rows where 'c1' > 11 and index is 'r3' or 'r5':

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r3	20	21	22	23	24	25	26	27	28	29
r5	40	41	42	43	44	45	46	47	48	49

1. ডেটাফ্রেম তৈরি:

- index1 এবং column1 এর মাধ্যমে একটি ১০x১০ ডেটাফ্রেম তৈরি করা হয়েছে, যেখানে index1 রো নামে এবং column1 কলাম নামে ব্যবহৃত হয়েছে।
- array_2d এর মাধ্যমে ০ থেকে ৯৯ পর্যন্ত সংখ্যা সাজিয়ে ১০x১০ আকারে রূপান্তরিত করা হয়েছে এবং সেটি ডেটাফ্রেমে ব্যবহার করা হয়েছে।

2. 'c1' কলামে ৭০ এর মান কোথায় রয়েছে তা চেক করা:

- df['c1'] == 70 দিয়ে দেখা হয়েছে কোথায় 'c1' কলামে ৭০ রয়েছে।
- তারপর df[df['c1'] == 70] দিয়ে সেই রো গুলিকে দেখানো হয়েছে যেখানে 'c1' কলামে ৭০ রয়েছে।

3. 'c1' কলামে ১১ এর বেশি মান গুলিকে চেক করা:

- df['c1'] > 11 দিয়ে 'c1' কলামের ১১ এর বেশি মান গুলির জন্য একটি বুলিয়ান মাস্ক তৈরি করা হয়েছে।
- এরপর, df[boolean_mask].loc[['r3', 'r5']] দিয়ে শুধুমাত্র রো 'r3' এবং 'r5' এর জন্য সেই মানগুলো দেখানো হয়েছে।

Combine 2 conditions

Let's try on c1 for a value > 60 and on c2 for a value > 80

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)

# Print the original DataFrame
print(f"Original DataFrame:\n{df}")

# Define a condition: 'c1' > 60 and 'c2' > 80
minhaz = (df['c1'] > 60) & (df['c2'] > 80)

# Print the boolean mask
print(f"\nBoolean mask where 'c1' > 60 and 'c2' > 80:\n{minhaz}")

# Print rows where the condition is True
```

```
print(f"\nRows where 'c1' > 60 and 'c2' > 80:\n{df[minhaz]}")
```

Original DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29
r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

Boolean mask where 'c1' > 60 and 'c2' > 80:

r1	False
r2	False
r3	False
r4	False
r5	False
r6	False
r7	False
r8	False
r9	True
r10	True

dtype: bool

Rows where 'c1' > 60 and 'c2' > 80:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r9	80	81	82	83	84	85	86	87	88	89
r10	90	91	92	93	94	95	96	97	98	99

তারপর:

1. **কন্ডিশন:** এটি যাচাই করছে কোন কোন রো-তে c1 কলামের মান ৬০-এর বেশি এবং c2 কলামের মান ৮০-এর বেশি।
2. **বুলিয়ান মাস্ক:** প্রথমে এই শর্তটি একটি বুলিয়ান মাস্কে (True বা False) রূপান্তরিত হচ্ছে, যেখানে True মানে শর্তটি পূর্ণ হয়েছে এবং False মানে শর্তটি পূর্ণ হয়নি।
3. **ফলাফল:** এরপর, এই বুলিয়ান মাস্ক ব্যবহার করে সেই রো গুলি প্রিন্ট করা হচ্ছে যেখানে উল্লিখিত শর্তটি পূর্ণ হয়েছে (c1 > 60 এবং c2 > 80).

ফলাফলস্বরূপ:

- প্রথমে প্রিন্ট হবে পুরো DataFrame।
- তারপর, কন্ডিশন অনুযায়ী শর্তযুক্ত বুলিয়ান মাস্কটি প্রিন্ট হবে (True বা False হবে)।
- শেষে, শুধু DataFrame এর সেই রো গুলি প্রিন্ট হবে যেখানে c1 > 60 এবং c2 > 80।

এভাবে, কোডটি শর্ত অনুযায়ী সিলেক্টেড রো গুলি দেখাবে।

Let's have a quick look on couple of useful methods: `reset_index()` and `set_index()`

pandas লাইব্রেরি ব্যবহার করে একটি DataFrame তৈরি এবং বিভিন্ন ইনডেক্স পরিবর্তন ফাংশন `reset_index()` এবং `set_index()` এর কার্যপদ্ধতি প্রদর্শন: (এই টপিক আমি বুঝি নাই -_-) কেউ বুঝে আমাকে বুঝাইয়ো

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)

# reset_index() without inplace: It will return a new DataFrame
with the original index converted into a column
print(f"Original DataFrame with
reset_index():\n{df.reset_index()}")

# Print the original DataFrame
print(f"\nOriginal DataFrame:\n{df}")

# reset_index() with inplace=True: It modifies the original
DataFrame by adding the index as a column and resetting the index
df.reset_index(inplace=True)
print(f"\nDataFrame after reset_index(inplace=True):\n{df}")

# Reset the index again without inplace, which returns a new
DataFrame (original DataFrame remains unchanged)
print(f"\nDataFrame after another reset_index()
call:\n{df.reset_index()}")
print(f"Original DataFrame after second reset_index():\n{df}")

# Adding a new column 'newind' with custom index values
newind = 'a b c d e f g h i j'.split() # Custom index values
df['newind'] = newind # Add the new column 'newind'

# set_index() to set the new column as the index
df.set_index('newind', inplace=True)
print(f"\nDataFrame after set_index('newind'):\n{df}")
```

```
Original DataFrame with reset_index():
   index  c1  c2  c3  c4  c5  c6  c7  c8  c9  c10
0    r1   0   1   2   3   4   5   6   7   8   9
1    r2  10  11  12  13  14  15  16  17  18  19
2    r3  20  21  22  23  24  25  26  27  28  29
3    r4  30  31  32  33  34  35  36  37  38  39
4    r5  40  41  42  43  44  45  46  47  48  49
5    r6  50  51  52  53  54  55  56  57  58  59
```

```

6    r7  60  61  62  63  64  65  66  67  68  69
7    r8  70  71  72  73  74  75  76  77  78  79
8    r9  80  81  82  83  84  85  86  87  88  89
9   r10  90  91  92  93  94  95  96  97  98  99

```

Original DataFrame:

```

   c1  c2  c3  c4  c5  c6  c7  c8  c9  c10
r1   0   1   2   3   4   5   6   7   8    9
r2  10  11  12  13  14  15  16  17  18   19
r3  20  21  22  23  24  25  26  27  28   29
r4  30  31  32  33  34  35  36  37  38   39
r5  40  41  42  43  44  45  46  47  48   49
r6  50  51  52  53  54  55  56  57  58   59
r7  60  61  62  63  64  65  66  67  68   69
r8  70  71  72  73  74  75  76  77  78   79
r9  80  81  82  83  84  85  86  87  88   89
r10 90  91  92  93  94  95  96  97  98   99

```

DataFrame after reset_index(inplace=True):

```

   index  c1  c2  c3  c4  c5  c6  c7  c8  c9  c10
0    r1   0   1   2   3   4   5   6   7   8    9
1    r2  10  11  12  13  14  15  16  17  18   19
2    r3  20  21  22  23  24  25  26  27  28   29
3    r4  30  31  32  33  34  35  36  37  38   39
4    r5  40  41  42  43  44  45  46  47  48   49
5    r6  50  51  52  53  54  55  56  57  58   59
6    r7  60  61  62  63  64  65  66  67  68   69
7    r8  70  71  72  73  74  75  76  77  78   79
8    r9  80  81  82  83  84  85  86  87  88   89
9   r10  90  91  92  93  94  95  96  97  98   99

```

DataFrame after another reset_index() call:

```

   level_0  index  c1  c2  c3  c4  c5  c6  c7  c8  c9  c10
0         0    r1   0   1   2   3   4   5   6   7   8    9
1         1    r2  10  11  12  13  14  15  16  17  18   19
2         2    r3  20  21  22  23  24  25  26  27  28   29
3         3    r4  30  31  32  33  34  35  36  37  38   39
4         4    r5  40  41  42  43  44  45  46  47  48   49
5         5    r6  50  51  52  53  54  55  56  57  58   59
6         6    r7  60  61  62  63  64  65  66  67  68   69
7         7    r8  70  71  72  73  74  75  76  77  78   79
8         8    r9  80  81  82  83  84  85  86  87  88   89
9         9   r10  90  91  92  93  94  95  96  97  98   99

```

Original DataFrame after second reset_index():

```

   index  c1  c2  c3  c4  c5  c6  c7  c8  c9  c10
0    r1   0   1   2   3   4   5   6   7   8    9
1    r2  10  11  12  13  14  15  16  17  18   19
2    r3  20  21  22  23  24  25  26  27  28   29
3    r4  30  31  32  33  34  35  36  37  38   39
4    r5  40  41  42  43  44  45  46  47  48   49
5    r6  50  51  52  53  54  55  56  57  58   59
6    r7  60  61  62  63  64  65  66  67  68   69

```

7	r8	70	71	72	73	74	75	76	77	78	79
8	r9	80	81	82	83	84	85	86	87	88	89
9	r10	90	91	92	93	94	95	96	97	98	99

DataFrame after set_index('newind'):

	index	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
newind											
a	r1	0	1	2	3	4	5	6	7	8	9
b	r2	10	11	12	13	14	15	16	17	18	19
c	r3	20	21	22	23	24	25	26	27	28	29
d	r4	30	31	32	33	34	35	36	37	38	39
e	r5	40	41	42	43	44	45	46	47	48	49
f	r6	50	51	52	53	54	55	56	57	58	59
g	r7	60	61	62	63	64	65	66	67	68	69
h	r8	70	71	72	73	74	75	76	77	78	79
i	r9	80	81	82	83	84	85	86	87	88	89
j	r10	90	91	92	93	94	95	96	97	98	99

ব্যাখ্যা:

1. reset_index() (বিনা inplace):

🔗 প্রথমে, reset_index() ফাংশনটি কল করা হয়েছে যাতে DataFrame এর ইনডেক্সটিকে একটি সাধারণ কলাম হিসেবে রূপান্তর করা হয়। এই ফাংশনটি একটি নতুন DataFrame রিটার্ন করে, যেখানে আগের ইনডেক্স কলাম হিসেবে যুক্ত থাকে। মূল DataFrame অপরিবর্তিত থাকে।

🔗 আউটপুটে দেখা যাবে, যে DataFrame-এ একটি নতুন কলাম "index" যোগ করা হয়েছে, যা আগের ইনডেক্সের মানগুলো ধারণ করে।

2. reset_index() (with inplace=True):

🔗 এখানে reset_index() ফাংশনটি inplace=True দিয়ে ব্যবহার করা হয়েছে। এর মানে হলো যে, মূল DataFrame-এই পরিবর্তন ঘটবে এবং এটি ইনডেক্সকে কলামে রূপান্তরিত করবে। নতুন একটি DataFrame রিটার্ন করা হবে না।

🔗 এর ফলে, df DataFrame-টি পরিবর্তিত হবে, এবং index নামক একটি কলাম যোগ হবে যা আগের ইনডেক্সের মান ধারণ করবে।

3. reset_index() পুনরায় (বিনা inplace):

🔗 reset_index() আবার ব্যবহার করা হয়েছে, তবে এবার inplace=False রাখলে এটি নতুন একটি DataFrame তৈরি করবে, যেখানে "index" কলাম দুটি থাকবে—একটি পুরোনো ইনডেক্স (যা প্রথমবারের reset-এ তৈরি হয়েছে) এবং নতুন "index" কলাম (যা নতুন ইনডেক্স রূপে পরিবর্তিত হয়েছে)।

🔗 তবে, মূল df DataFrame অপরিবর্তিত থাকবে, কারণ inplace=False ব্যবহার করা হয়েছে।

4. নতুন কলাম newind যোগ করা এবং set_index() ব্যবহার:

🔗 নতুন একটি newind নামক কলাম তৈরি করা হয়েছে, যেখানে 'a', 'b', 'c', ... 'j' মান রয়েছে।

🔗 পরে, set_index() ফাংশনটি ব্যবহার করে এই newind কলামটিকে নতুন ইনডেক্স হিসেবে সেট করা হয়েছে। এর ফলে, newind কলামটি DataFrame এর ইনডেক্স হিসেবে কাজ করবে এবং মূল DataFrame পরিবর্তিত হবে, কারণ inplace=True ব্যবহার করা হয়েছে।

সারসংক্ষেপ:

এই কোডের মাধ্যমে আপনি `reset_index()` এবং `set_index()` ফাংশনগুলো কিভাবে কাজ করে তা দেখতে পাচ্ছেন:

- `reset_index()` ইনডেক্সকে কলামে রূপান্তরিত করে এবং ইনডেক্স পুনঃস্থাপন করে।
- `set_index()` একটি নির্দিষ্ট কলামকে ইনডেক্স হিসেবে সেট করে।
- `inplace=True` ব্যবহার করলে মূল DataFrame পরিবর্তিত হয়, আর `inplace=False` থাকলে নতুন DataFrame তৈরি হয়।

head(), tail():

info(): Provides a concise summary of the DataFrame.

describe(): Generates descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.

```
import pandas as pd
import numpy as np

# Create two string lists for the index and columns
index1 = ['r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10']
column1 = 'c1 c2 c3 c4 c5 c6 c7 c8 c9 c10'.split()

# Create a 10x10 array of numbers from 0 to 99
array_2d = np.arange(0, 100).reshape(10, 10)

# Create DataFrame using the generated 2D array and the index and columns
df = pd.DataFrame(data=array_2d, index=index1, columns=column1)

# Print the first 5 rows of the DataFrame using f-string
print(f"First 5 rows of the DataFrame:\n{df.head()}")

# Print the last 5 rows of the DataFrame using f-string
print(f"\nLast 5 rows of the DataFrame:\n{df.tail()}")

# Print the summary statistics of the DataFrame using f-string
print(f"\nSummary Statistics of the DataFrame:\n{df.describe()}")
```

First 5 rows of the DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	0	1	2	3	4	5	6	7	8	9
r2	10	11	12	13	14	15	16	17	18	19
r3	20	21	22	23	24	25	26	27	28	29
r4	30	31	32	33	34	35	36	37	38	39
r5	40	41	42	43	44	45	46	47	48	49

Last 5 rows of the DataFrame:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r6	50	51	52	53	54	55	56	57	58	59
r7	60	61	62	63	64	65	66	67	68	69
r8	70	71	72	73	74	75	76	77	78	79

```

r9      80  81  82  83  84  85  86  87  88  89
r10     90  91  92  93  94  95  96  97  98  99

Summary Statistics of the DataFrame:

c6 \      c1      c2      c3      c4      c5
count  10.000000  10.000000  10.000000  10.000000  10.000000
mean    45.000000  46.000000  47.000000  48.000000  49.000000
std     30.276504  30.276504  30.276504  30.276504  30.276504
min      0.000000   1.000000   2.000000   3.000000   4.000000
25%     22.500000  23.500000  24.500000  25.500000  26.500000
50%     45.000000  46.000000  47.000000  48.000000  49.000000
75%     67.500000  68.500000  69.500000  70.500000  71.500000
max     90.000000  91.000000  92.000000  93.000000  94.000000

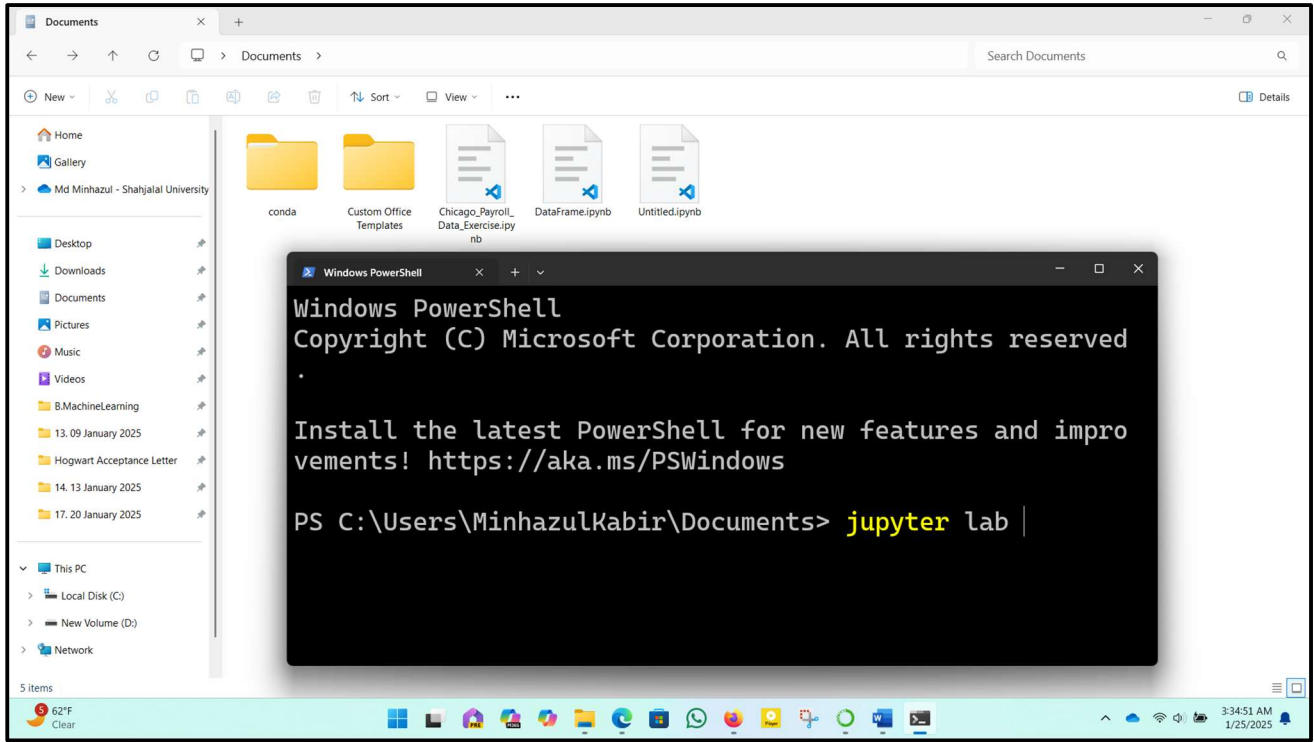
c6 \      c7      c8      c9      c10
count  10.000000  10.000000  10.000000  10.000000
mean    51.000000  52.000000  53.000000  54.000000
std     30.276504  30.276504  30.276504  30.276504
min      6.000000   7.000000   8.000000   9.000000
25%     28.500000  29.500000  30.500000  31.500000
50%     51.000000  52.000000  53.000000  54.000000
75%     73.500000  74.500000  75.500000  76.500000
max     96.000000  97.000000  98.000000  99.000000

```

মতামতঃ

- 1) Nested list হচ্ছে numpy এর 2d array
- 2) Model কে ডেটা দিয়ে ট্রেন করাবো machine learning এ। Picture = data, Video = data, Website = data, Excel = data, Csv = data, Database = data. আমরা আশেপাশে যা কিছু দেখতে পাই না কেন, তার সব কিছুই হচ্ছে data.

যেকোনো ফোল্ডার বা লোকাল ড্রাইভে প্রজেক্ট তৈরি করলে, প্রথমে সেই ফোল্ডারে গিয়ে Command Line বা Terminal খুলুন। এরপর, সেখানে jupyter lab টাইপ করুন এবং Enter চাপুন। এর ফলে, স্বয়ংক্রিয়ভাবে Jupyter Lab চালু হয়ে প্রজেক্টটি সেই ফোল্ডারে লোড হবে। এখন আপনি ব্রাউজারে প্রজেক্ট ফাইলগুলো এডিট করতে পারবেন।



এই Terminal কেটে দিলেই jupyter lab বন্ধ হয়ে যাবে।