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. Assignment #14:

Pandas একটি জনপ্রিয় Python লাইব্রেরি যা ডেটা সায়েন্স, বিশ্লেষণ এবং ম্যানিপুলেশনের জন্য ব্যবহৃত হয়। এটি বিশেষত **DataFrame** (যে ধরনের ডেটা টেবিল আকারে থাকে) এবং **Series** (একটি একক কলাম বা ১ডি ডেটা) এর মত ডেটা স্ট্রাকচার প্রদান করে, যা বিশাল ডেটা সেট নিয়ে কাজ করতে সহজ করে তোলে।

Pandas এর প্রধান ফিচার:

1. **DataFrame:** টেবিল আকারে ডেটা, যেখানে রো এবং কলাম থাকে। DataFrame হলো excel/sql
2. **Series:** একক কলাম বা ১ডি ডেটা। Series হলো array with index
3. **Data Alignment:**
4. **GroupBy:**
5. **Time Series:**
6. **I/O Tools:** বিভিন্ন ফাইল ফরম্যাট যেমন CSV, Excel, SQL ইত্যাদি থেকে ডেটা লোড এবং সেভ করার সক্ষমতা।

1. Brief overview of Pandas and installation guide.

Pandas ইনস্টলেশন গাইড:

পান্ডাস ইনস্টল করা খুব সহজ। এটি **pip** ব্যবহার করে ইনস্টল করা যেতে পারে। নিচে ধাপ গুলি দেওয়া হল:

1. প্রথমে আপনার সিস্টেমে Python ইনস্টল থাকতে হবে।
2. তারপর, টার্মিনাল বা কমান্ড প্রম্পট খুলুন।
3. নিচের কমান্ডটি লিখুন:

```
pip3 install pandas
```

এটি Pandas ইনস্টল করে দেবে। যদি Jupyter Lab বা Anaconda তে আগে থেকে pandas install করা থাকে।

2. Introduction to Pandas library.

Pandas লাইব্রেরির পরিচিতি বলতে বুঝায়, এটি কী এবং কীভাবে কাজ করে, এবং কেন এটি ডেটা সায়েন্স ও ডেটা বিশ্লেষণের জন্য অত্যন্ত গুরুত্বপূর্ণ।

Pandas হল একটি **Python লাইব্রেরি** যা ডেটা ম্যানিপুলেশন এবং বিশ্লেষণ করতে ব্যবহৃত হয়। এটি মূলত দুইটি প্রধান ডেটা স্ট্রাকচার প্রদান করে:

1. **Series:** এটি একটি একক ডেটা কলাম (একটি ১ডি অ্যারে) এবং এটি ইনডেক্সের সাহায্যে ডেটার অ্যাক্সেস সুবিধা দেয়।
2. **DataFrame:** এটি একটি টেবিল আকারে ডেটা, যেখানে রো এবং কলাম থাকে, যেমন একটি স্প্রেডশিট বা SQL টেবিল।

Pandas লাইব্রেরি তৈরি করা হয়েছিল ডেটা আকারে কাজ করার জন্য, যেমন ডেটার মধ্যে ফিল্টার, সোর্ট, গ্রুপিং, ফিল্টারিং, এবং হিসাব (যেমন গড়, যোগফল) বের করার সুবিধা দেওয়া।

Pandas এর প্রধান বৈশিষ্ট্যসমূহ:

- **DataFrame ও Series:** টেবিল বা কলাম আকারে ডেটা ধারণের জন্য সহজে ব্যবহারযোগ্য স্ট্রাকচার।
- **ডেটা ক্লিনিং:** নষ্ট বা মিসিং ডেটা নিয়ে কাজ করা, যেমন isnull(), dropna(), ইত্যাদি ব্যবহার করা।
- **ডেটার সিলেকশন ও ফিল্টারিং:** ইনডেক্স বা কলাম দিয়ে ডেটা নির্বাচন।
- **গ্রুপিং:** ডেটাকে গ্রুপ করে বিভিন্ন অপারেশন করা (যেমন groupby() ব্যবহার করে)।
- **আই/ও অপারেশন:** CSV, Excel, SQL, JSON ইত্যাদি ফরম্যাট থেকে ডেটা পড়া এবং লিখা।

Pandas লাইব্রেরি আপনার ডেটা সেট নিয়ে দ্রুত কাজ করতে সহায়তা করে, যেমন বিশ্লেষণ, ডেটা ক্লিনিং, ভিজ্যুয়ালাইজেশন, ইত্যাদি।

এখন, যখন আপনি pandas ব্যবহার করবেন, আপনি সহজেই ডেটা নিয়ে বিভিন্ন ধরনের বিশ্লেষণ এবং কাজ করতে পারবেন, যেটা এক সময় হয়তো Excel বা অন্যান্য টুলে অনেক সময়সাপেক্ষ ছিল।

Pandas লাইব্রেরি সহজে ব্যবহারযোগ্য হলেও এটি অত্যন্ত শক্তিশালী এবং এর অনেক সুবিধা রয়েছে।

3. Pandas data structures: Series.

Pandas Essentials Pandas পরিচিতি

Pandas একটি ওপেন সোর্স লাইব্রেরি যা Python প্রোগ্রামিং ভাষার জন্য হাই-পারফরম্যান্স, সহজে ব্যবহারযোগ্য ডেটা স্ট্রাকচার এবং ডেটা বিশ্লেষণ টুলস প্রদান করে। আজকাল, pandas একটি সক্রিয় কমিউনিটি দ্বারা সমর্থিত যা সারা পৃথিবী থেকে একে সহায়তা করে, যাতে ওপেন সোর্স pandas সম্ভব হয়।

"Python for Data Analysis" বইটি লেখা হয়েছে Wes McKinney দ্বারা, যিনি Pandas লাইব্রেরির স্রষ্টা।

এই কোর্সের এই অংশে আমরা pandas ব্যবহার করে ডেটা বিশ্লেষণ শিখবো। যদি আপনি কখনো pandas ব্যবহার না করে থাকেন, তবে আপনি pandas-কে এক ধরনের অত্যন্ত শক্তিশালী Excel হিসেবে ভাবতে পারেন, তবে এতে আরও অনেক বেশি বৈশিষ্ট্য রয়েছে।

এই কোর্সের **Pandas Essentials** অংশে আমরা নিচের গুরুত্বপূর্ণ বিষয়গুলো শিখবো:

1. **Series**
2. **Series থেকে ডেটা নেওয়া:**
   সিরিজের উপর বিভিন্ন মৌলিক অপারেশন সাধারণত ইনডেক্সের উপর ভিত্তি করে করা হয়।
   isnull(), notnull(), axes, values, head(), tail(), size, empty ইত্যাদি।

আরো অনেক **গুরুত্বপূর্ণ পদ্ধতি এবং অপারেশন** থাকবে এবং শেষে দুইটি পূর্ণ ডেটা বিশ্লেষণ অভ্যাস থাকবে, যেখানে আপনি আপনার শিখে নেয়া স্কিল ব্যবহার করতে পারবেন।

তাহলে, প্রথমেই হলো ইনস্টলেশন, আশা করছি আপনি ইতিমধ্যেই এই শক্তিশালী লাইব্রেরি ইনস্টল করে ফেলেছেন, যদি না করে থাকেন, তবে...

Series:

Series হলো একটি একমাত্রিক (one-dimensional) অ্যারে-সদৃশ অবজেক্ট, যার মধ্যে মান এবং মানগুলির সাথে সম্পর্কিত একটি লেবেল অ্যারে থাকে। Series-কে লেবেল ব্যবহার করে ইনডেক্স করা যেতে পারে।
(Series হল NumPy অ্যারের মতো, আসলে এটি NumPy অ্যারে অবজেক্টের ওপর নির্মিত।)
Series যেকোনো ধরনের পছন্দসই Python অবজেক্ট ধারণ করতে পারে।

We can create a Series using list, numpy array, or dictionary: Let's create these objects and convert them into panda's Series!

Series using lists: Lets create a Python list containing labels and another with data.

```
import numpy as np
import pandas as pd

my_data = [100, 200, 300]
# Converting my_data (Python list) to Series (pandas series)
minhaz = pd.Series(data=my_data)

# Printing important content in the print statement
print(f"This is the Pandas Series created from the Python
list:\n{minhaz}")
```

```
This is the Pandas Series created from the Python list:
0    100
1    200
2    300
dtype: int64
```

এই কোডটি সংক্ষেপে ব্যাখ্যা করা হলো:

1. **import numpy as np** এবং **import pandas as pd**: numpy এবং pandas লাইব্রেরি ইম্পোর্ট করা হচ্ছে। pandas ডেটা ম্যানিপুলেশনের জন্য এবং numpy গণনা করার জন্য ব্যবহৃত হয়।

2. **my_data = [100, 200, 300]**: এটি একটি সাধারণ Python লিস্ট, যার মধ্যে তিনটি মান রয়েছে (100, 200, 300)।
3. **minhaz = pd.Series(data=my_data)**: এখানে my_data লিস্টটিকে pandas সিরিজে রূপান্তর করা হচ্ছে। pd.Series একটি pandas ফাংশন, যা একমাত্রিক ডেটা (একটি অ্যারে) গ্রহণ করে এবং এটি সিরিজ হিসেবে ফেরত দেয়।
4. **print(f"...")**: ফরম্যাটেড স্ট্রিং ব্যবহার করে minhaz সিরিজটি প্রিন্ট করা হচ্ছে। এখানে \n দিয়ে একটি নতুন লাইন শুরু হচ্ছে, যাতে সিরিজটি সঠিকভাবে প্রদর্শিত হয়।

এই কোডের ফলস্বরূপ, my_data লিস্টটিকে pandas সিরিজে রূপান্তর করে এবং সেটি প্রিন্ট করা হবে।

numpy বা pandas এর কোনো fanction call দিতে হলে, np/pd দিয়ে কল দিতে হবে।

কোলাম "0 1 2" হল সিরিজের উপাদানগুলোর জন্য স্বয়ংক্রিয়ভাবে তৈরি করা ইনডেক্স, যার ডেটা হচ্ছে 100, 200 এবং 300। আমরা ইনডেক্স মান নির্দিষ্ট করতে পারি এবং এই ইনডেক্স ব্যবহার করে ডেটা পয়েন্টগুলি কল করতে পারি। এখন "my_labels" কে ইনডেক্স হিসেবে সিরিজে পাস করি।

Series হচ্ছে one dimensional array এবং তাতে value এবং তার index (ক্রম) থাকে। কিছু না লিখলে default হিসেবে 0,1,2,3, . . .থাকে। আর, নিজের ইচ্ছামতন লিখা যায় ।

```
import numpy as np
import pandas as pd

my_data = [100, 200, 300]
my_labels = ['x', 'y', 'z']

# Creating the pandas Series with specified index labels
minhaz = pd.Series(data=my_data, index=my_labels)

# Printing the Series with f-string
print(f"The Pandas Series with data {my_data} and index labels
{my_labels} is:\n{minhaz}")
```

```
The Pandas Series with data [100, 200, 300] and index labels ['x',
'y', 'z'] is:
x    100
y    200
z    300
dtype: int64
```

এই কোডটির ব্যাখ্যা হলো:

1. **import numpy as np** এবং **import pandas as pd**: এখানে numpy এবং pandas লাইব্রেরি ইম্পোর্ট করা হচ্ছে। numpy গণনা সংক্রান্ত কাজের জন্য এবং pandas ডেটা ম্যানিপুলেশনের জন্য ব্যবহৃত হয়।
2. **my_data = [100, 200, 300]**: একটি ডেটা লিস্ট, যার মান 100, 200, 300।

3. **my_labels = ['x', 'y', 'z']**: ইনডেক্স লেবেলগুলো, যা সিরিজের জন্য 'x', 'y', 'z' হবে।
4. **minhaz = pd.Series(data=my_data, index=my_labels)**: pandas সিরিজ তৈরি করা হচ্ছে, যেখানে my_data লিস্টের মানগুলো এবং my_labels লেবেলগুলো ইনডেক্স হিসেবে ব্যবহৃত হচ্ছে।

ফলস্বরূপ, একটি pandas সিরিজ তৈরি হবে এবং প্রিন্ট করা হবে, যেখানে 'x', 'y', 'z' ইনডেক্সের বিপরীতে 100, 200, 300 মান থাকবে।

Series হচ্ছে one dimensional array তাতে শুধুমাত্র একটি row এর মান লম্বালম্বিভাবে আসে। সব value একই ধরণের হয়।

List and array এর মধ্যে সিরিজে পার্থক্য নাই Series এ Data Load করার সময়ে ।

my_data = [100, 200, 300]	লিস্ট বানালাম।
pd.Series(data = my_data)	লিস্টকে pandas এর Series এ লোড করলাম।
my_array = np.array(my_data)	লিস্টকে numpy এর মাধ্যমে array বানালাম ।
pd.Series(data = my_array)	array pandas এর Series এ লোড করলাম। এবং উপরের মতন একই পাবো।

Series using NumPy arrays

```
import numpy as np
import pandas as pd

my_data = [100, 200, 300]
my_labels = ['x', 'y', 'z']

# Lets create NumPy array from my_data and then Series from that
array
numpy_arr1 = np.array(my_data)

# Creating the Series from the NumPy array
minhaz = pd.Series(data=numpy_arr1)
print(f"The Pandas Series created from the NumPy array
is:\n{minhaz}")

# Creating the Series from the NumPy array with specified index
labels
kabir = pd.Series(data=numpy_arr1, index=my_labels)
print(f"The Pandas Series with custom index labels is:\n{kabir}")
```

```
The Pandas Series created from the NumPy array is:
0    100
1    200
2    300
dtype: int32
The Pandas Series with custom index labels is:
x    100
y    200
z    300
```

```
dtype: int32
```

1. **my_data = [100, 200, 300]:** এটি একটি সাধারণ Python লিস্ট, যার মধ্যে তিনটি মান রয়েছে (100, 200, 300)।
2. **my_labels = ['x', 'y', 'z']:** এটি একটি লিস্ট, যার মধ্যে ইনডেক্স লেবেলগুলি (x, y, z) রয়েছে, যেগুলো সিরিজে ব্যবহৃত হবে।
3. **numpy_arr1 = np.array(my_data):** এখানে my_data লিস্টটিকে একটি NumPy অ্যারে (array) এ রূপান্তরিত করা হয়েছে।
4. **minhaz = pd.Series(data=numpy_arr1):** এখানে NumPy অ্যারে numpy_arr1 থেকে একটি pandas সিরিজ তৈরি করা হয়েছে, যেখানে স্বয়ংক্রিয়ভাবে ইনডেক্স তৈরি হবে (যেমন 0, 1, 2...)।
5. **kabir = pd.Series(data=numpy_arr1, index=my_labels):** এখানে একই NumPy অ্যারে ব্যবহার করে একটি pandas সিরিজ তৈরি করা হয়েছে, তবে এইবার ইনডেক্স হিসেবে my_labels লেবেলগুলো দেওয়া হয়েছে (x, y, z)।
6. **print(f"..."):** ফরম্যাটেড স্ট্রিং ব্যবহার করে সিরিজের তথ্য প্রিন্ট করা হয়েছে। প্রথমে সিরিজটি স্বাভাবিক ইনডেক্সসহ এবং তারপর কাস্টম ইনডেক্সসহ প্রিন্ট করা হচ্ছে।

ফলস্বরূপ: প্রথম সিরিজটি স্বাভাবিক ইনডেক্স (0, 1, 2) সহ হবে, আর দ্বিতীয় সিরিজটি কাস্টম ইনডেক্স (x, y, z) সহ হবে।

Series using dictionary:

Notice the difference here,

if we pass a dictionary to Series, pandas will take the keys as index/labels and values as data.

আমরা জানি, dictionary এর KEY এবং VALUE থাকে।

Series can hold a wide variety of objects(string) types, lets see with examples:

```
import numpy as np
import pandas as pd

# Let's create a dictionary my_dic
my_dic = {'x': 100, 'y': 200, 'z': 300}

# Creating a pandas Series from the dictionary
minhaz = pd.Series(data=my_dic)
print(f"The Pandas Series created from the dictionary
is:\n{minhaz}")

my_list = ['x', 'y', 'z']
# Creating a pandas Series by passing my_list (a list of strings)
as data
kabir = pd.Series(data=my_list)
print(f"The Pandas Series created from the list of labels
is:\n{kabir}")
```

```
The Pandas Series created from the dictionary is:
x    100
y    200
z    300
```

```
dtype: int64
The Pandas Series created from the list of labels is:
0    x
1    y
2    z
dtype: object
```

1. **my_dic = {'x': 100, 'y': 200, 'z': 300}**: একটি ডিকশনারি তৈরি করা হয়েছে, যেখানে কী (key) হলো 'x', 'y', 'z' এবং মান (value) হলো 100, 200, 300।
2. **minhaz = pd.Series(data=my_dic)**: my_dic ডিকশনারি থেকে একটি pandas সিরিজ তৈরি করা হয়েছে। সিরিজে কীগুলো হবে ইনডেক্স এবং মানগুলো হবে ডেটা।
3. **my_list = ['x', 'y', 'z']**: একটি সাধারণ লিস্ট তৈরি করা হয়েছে, যার মধ্যে স্ট্রিং (x, y, z) রয়েছে।
4. **kabir = pd.Series(data=my_list)**: এখানে my_list লিস্ট থেকে একটি pandas সিরিজ তৈরি করা হয়েছে। এখানে লিস্টের উপাদানগুলো সিরিজের ডেটা হিসেবে ব্যবহার হবে, এবং স্বয়ংক্রিয়ভাবে ইনডেক্স 0, 1, 2 হবে।
5. **print(f"...")**: ফরম্যাটেড স্ট্রিং ব্যবহার করে সিরিজের বিষয়বস্তু প্রিন্ট করা হয়েছে। প্রথমে ডিকশনারি থেকে তৈরি সিরিজ এবং তারপর লিস্ট থেকে তৈরি সিরিজ প্রিন্ট করা হচ্ছে।

ফলস্বরূপ: প্রথম সিরিজটি ডিকশনারি থেকে তৈরি হয়েছে, যেখানে 'x', 'y', 'z' ইনডেক্স হিসেবে এবং 100, 200, 300 মান হিসেবে রয়েছে। দ্বিতীয় সিরিজটি একটি সাধারণ লিস্ট থেকে তৈরি হয়েছে, যেখানে স্ট্রিং 'x', 'y', 'z' ডেটা হিসেবে রয়েছে।

Grabbing data from Series:

Indexes are the key thing to understand in Series. Pandas use these indexes (numbers or names) for fast information retrieval. (Index works just like a hash table or a dictionary).
To understand the concepts, Let's create three Series, ser1, ser2, ser3 from dictionaries with some random data:

```
import numpy as np
import pandas as pd

# Creating dictionaries
dic_1 = {'Toronto': 500, 'Calgary': 200, 'Vancouver': 300,
'Montreal': 700}
dic_2 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700}
dic_3 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700,
'Jasper': 1000}

# Creating pandas series from the dictionaries
ser1 = pd.Series(dic_1)
print(f"The Pandas Series created from dic_1 is:\n{ser1}")

ser2 = pd.Series(dic_2)
print(f"\nThe Pandas Series created from dic_2 is:\n{ser2}")

ser3 = pd.Series(dic_3)
print(f"\nThe Pandas Series created from dic_3 is:\n{ser3}")
```

```
The Pandas Series created from dic_1 is:
Toronto      500
Calgary      200
Vancouver    300
Montreal     700
dtype: int64

The Pandas Series created from dic_2 is:
Calgary      200
Vancouver    300
Montreal     700
dtype: int64

The Pandas Series created from dic_3 is:
Calgary       200
Vancouver     300
Montreal      700
Jasper       1000
dtype: int64
```

1. **dic_1 = {'Toronto': 500, 'Calgary': 200, 'Vancouver': 300, 'Montreal': 700}**: একটি ডিকশনারি তৈরি করা হয়েছে, যেখানে শহরের নাম (যেমন 'Toronto', 'Calgary', 'Vancouver', 'Montreal') কী (key) এবং সংশ্লিষ্ট মান (value) হলো তাদের সংখ্যা।
2. **dic_2 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700}** এবং **dic_3 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700, 'Jasper': 1000}**: আরও দুটি ডিকশনারি তৈরি করা হয়েছে, যা কিছু শহরের নাম এবং মান ধারণ করে।
3. **ser1 = pd.Series(dic_1), ser2 = pd.Series(dic_2), ser3 = pd.Series(dic_3)**: তিনটি pandas সিরিজ তৈরি করা হয়েছে, প্রতিটি সিরিজে ডিকশনারি থেকে ডেটা নেওয়া হয়েছে। সিরিজে শহরগুলোর নাম ইনডেক্স হিসেবে এবং সংখ্যাগুলো ডেটা হিসেবে থাকবে।
4. **print(f"...")**: ফরম্যাটেড স্ট্রিং ব্যবহার করে প্রতিটি সিরিজের বিষয়বস্তু প্রিন্ট করা হয়েছে। প্রতি সিরিজের আগে একটি বর্ণনা দেওয়া হয়েছে, যেমন "dic_1 থেকে তৈরি সিরিজ", "dic_2 থেকে তৈরি সিরিজ", ইত্যাদি।

ফলস্বরূপ: তিনটি pandas সিরিজ তৈরি হবে, যেগুলোর মধ্যে শহরের নাম ইনডেক্স হিসেবে এবং তাদের মান (সংখ্যা) ডেটা হিসেবে থাকবে।

Basic operations on series are usually based on the index.

উদাহরণস্বরূপ, যদি আমরা **ser1 + ser2** করি, তবে সিরিজ দুটি ইনডেক্সের মাধ্যমে মিলিয়ে অপারেশনটি সম্পন্ন হবে। মানে, যেসব ইনডেক্স উভয় সিরিজে মিলবে (যেমন **Calgary**, **Montreal**, এবং **Vancouver**), তাদের মান যোগ করা হবে। কিন্তু যেহেতু **Toronto** শুধুমাত্র ser1-এ রয়েছে, এবং ser2-এ নেই, তাই ser2-এ **Toronto** এর মান পাওয়া যাবে না এবং সেখানে **NaN** (Not a Number) রাখা হবে।

```
import numpy as np
import pandas as pd

# Creating dictionaries
```

```
dic_1 = {'Toronto': 500, 'Calgary': 200, 'Vancouver': 300,
'Montreal': 700}
dic_2 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700}
dic_3 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700,
'Jasper': 1000}

# Creating pandas Series from the dictionaries
ser1 = pd.Series(dic_1)
ser2 = pd.Series(dic_2)
ser3 = pd.Series(dic_3)

# Adding ser1 and ser2 and printing the result ক্রমিক হয়ে যাবে ।
ser4 = ser1 + ser2
print(f"The result of adding ser1 and ser2 is:\n{ser4}")

# Printing the content of ser3 ক্রম আগের মতন আসবে।
print(f"\nThe content of ser3 is:\n{ser3}")

# Adding ser1, ser2, and ser3 and printing the result ক্রমিক হয়ে
যাবে ।
ser5 = ser1 + ser2 + ser3
print(f"\nThe result of adding ser1, ser2, and ser3 is:\n{ser5}")
```

```
The result of adding ser1 and ser2 is:
Calgary       400.0
Montreal     1400.0
Toronto         NaN
Vancouver     600.0
dtype: float64

The content of ser3 is:
Calgary       200
Vancouver     300
Montreal      700
Jasper       1000
dtype: int64

The result of adding ser1, ser2, and ser3 is:
Calgary       600.0
Jasper          NaN
Montreal     2100.0
Toronto         NaN
Vancouver     900.0
dtype: float64
```

1. **ডিকশনারি তৈরি:**
    - dic_1, dic_2, এবং dic_3 নামে তিনটি ডিকশনারি তৈরি করা হয়েছে, যেখানে শহরের নাম এবং তাদের মান (সংখ্যা) রয়েছে।
2. **পান্ডাস সিরিজ তৈরি:**

- ser1, ser2, এবং ser3 সিরিজ তৈরি করা হয়েছে, যেখানে প্রতিটি সিরিজ একটি ডিকশনারি থেকে তৈরি হয়েছে। প্রতিটি সিরিজে শহরের নাম ইনডেক্স হিসেবে এবং তাদের মান ডেটা হিসেবে থাকবে।

3. **সিরিজ যোগফল:**
   - **ser4:** এখানে ser1 এবং ser2 সিরিজের মানগুলো যোগ করা হচ্ছে। **ser5:** এখানে ser1, ser2 এবং ser3 সিরিজের মানগুলো যোগ করা হচ্ছে। যেসব ইনডেক্স উভয় সিরিজে থাকবে (যেমন Calgary, Vancouver, Montreal), তাদের মান যোগ হবে। যেসব ইনডেক্স একটিতে রয়েছে আর অন্যটিতে নেই (যেমন: Toronto ও Jasper সবগুলোতে নেই), সেখানে **NaN** থাকবে।

4. **সিরিজ প্রিন্ট:**
   - প্রথমে **ser4** প্রিন্ট করা হয়েছে, যেটি ser1 এবং ser2 সিরিজের যোগফল।
   - এরপর **ser3** সিরিজের বিষয়বস্তু দেখানো হয়েছে।
   - শেষে **ser5** প্রিন্ট করা হয়েছে, যেটি ser1, ser2, এবং ser3 সিরিজগুলোর যোগফল।

দ্রষ্টব্য যে, সিরিজে পাওয়া মানগুলি তাদের উপযুক্ত ইনডেক্স অনুযায়ী যোগ করা হয়েছে, অন্যদিকে, যদি কোনো মিল না পাওয়া যায়, তবে মান হিসেবে **NaN** (Not a Number) প্রদর্শিত হবে, যা pandas-এ অনুপস্থিত বা NA (Not Available) মান চিহ্নিত করার জন্য ব্যবহৃত হয়।

isnull(), notnull()

- detect missing data

```
import numpy as np
import pandas as pd

# Creating dictionaries
dic_1 = {'Toronto': 500, 'Calgary': 200, 'Vancouver': 300,
'Montreal': 700}
dic_2 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700}
dic_3 = {'Calgary': 200, 'Vancouver': 300, 'Montreal': 700,
'Jasper': 1000}

# Creating pandas Series from the dictionaries
ser1 = pd.Series(dic_1)
ser2 = pd.Series(dic_2)
ser3 = pd.Series(dic_3)

# Adding ser1 and ser2
ser4 = ser1 + ser2
# Adding ser1, ser2, and ser3
ser5 = ser1 + ser2 + ser3

# Checking for NaN values in ser4 and printing the result
minhaz = pd.isnull(ser4)
print(f"The result of checking for NaN values in ser4
is:\n{minhaz}")
```

```
# Checking for non-NaN values in ser5 and printing the result
kabir = pd.notnull(ser5)
print(f"\nThe result of checking for non-NaN values in ser5
is:\n{kabir}")
```

```
The result of checking for NaN values in ser4 is:
Calgary      False
Montreal     False
Toronto       True
Vancouver    False
dtype: bool

The result of checking for non-NaN values in ser5 is:
Calgary       True
Jasper       False
Montreal      True
Toronto      False
Vancouver     True
dtype: bool
```

1. **ডিকশনারি তৈরি:**
   - dic_1, dic_2, এবং dic_3 ডিকশনারি তিনটি তৈরি করা হয়েছে, যেখানে বিভিন্ন শহরের নাম এবং তাদের সংশ্লিষ্ট মান রয়েছে।
2. **সিরিজ তৈরি:**
   - ser1, ser2, এবং ser3 সিরিজ তৈরি করা হয়েছে, যা প্রতিটি ডিকশনারি থেকে তৈরি করা সিরিজ।
3. **সিরিজ যোগফল:**
   - **ser4 = ser1 + ser2:** এখানে ser1 এবং ser2 সিরিজের মানগুলির যোগফল করা হচ্ছে।
   - **ser5 = ser1 + ser2 + ser3:** একইভাবে, ser1, ser2, এবং ser3 সিরিজের যোগফল করা হচ্ছে।
4. **NaN মান চেক করা:**
   - **minhaz = pd.isnull(ser4):** এখানে ser4 সিরিজের মধ্যে NaN মানগুলো চেক করা হচ্ছে। ফলস্বরূপ, যেখানে NaN থাকবে, সেখানে True এবং অন্যথায় False দেখানো হবে।
5. **Non-NaN মান চেক করা:**
   - **kabir = pd.notnull(ser5):** এখানে ser5 সিরিজের মধ্যে Non-NaN (অর্থাৎ, যে মানগুলো বৈধ) চেক করা হচ্ছে। যেখানে মান থাকবে, সেখানে True এবং যদি NaN থাকে, সেখানে False দেখানো হবে।

ফলস্বরূপ:

- প্রথম print স্টেটমেন্টে **NaN** চেক করে ফলাফল প্রদর্শিত হবে।
- দ্বিতীয় print স্টেটমেন্টে Non-NaN চেক করে ফলাফল দেখানো হবে।

axes, values

- axes: returns list of the row axis labels/index
- values: returns list of values/data

Let's try axes and values on our series!

```
import numpy as np
```

```
import pandas as pd

# Creating dictionaries
dic_1 = {'Toronto': 500, 'Calgary': 200, 'Vancouver': 300,
'Montreal': 700}

# Creating pandas Series from the dictionaries
ser1 = pd.Series(dic_1)

# Printing the row axis labels (index) of ser1
print(f"The row axis labels (index) of ser1 are:\n{ser1.axes}")

# Printing the values/data of ser1
print(f"\nThe values/data of ser1 are:\n{ser1.values}")
```

```
The row axis labels (index) of ser1 are:
[Index(['Toronto', 'Calgary', 'Vancouver', 'Montreal'],
dtype='object')]

The values/data of ser1 are:
[500 200 300 700]
```

1. **ডিকশনারি তৈরি**: dic_1 ডিকশনারি তৈরি করা হয়েছে, যেখানে শহরের নাম (যেমন Toronto, Calgary, Vancouver, Montreal) এবং তাদের সংশ্লিষ্ট মান (যেমন 500, 200, 300, 700) দেওয়া হয়েছে।
2. **পান্ডাস সিরিজ তৈরি**: ser1 সিরিজ তৈরি করা হয়েছে, যেটি dic_1 ডিকশনারি থেকে তৈরি।
3. **রো অ্যাক্সেস লেবেল (ইন্ডেক্স) মুদ্রণ: ser1.axes**: এটি ser1 সিরিজের ইনডেক্স বা KEY গুলি (শহরের নাম) প্রদর্শন করে।
4. **মান/ডেটা মুদ্রণ: ser1.values**: এটি ser1 সিরিজের ডেটা বা VALUE (সংখ্যা) প্রদর্শন করে।

সংক্ষেপে: কোডটি ser1 সিরিজ এর dict এর KEY এবং VALUE দেখায়। আমরা জানি, dictionary এর KEY ও VALUE থাকে।

head(), tail()

To view a small sample of a Series or DataFrame (we will learn DataFrame in the next lecture) object, use the **head()** and **tail()** methods.

The default number of elements to display is **five**, but you may pass a custom number.

আমরা একটি pandas সিরিজ তৈরি করেছি যেখানে শহরের নাম এবং তাদের সংশ্লিষ্ট মান রয়েছে। এরপর, আমরা head() এবং tail() ফাংশন ব্যবহার করে সিরিজের প্রথম এবং শেষ কিছু উপাদান প্রিন্ট করেছি।

```
import numpy as np
import pandas as pd

# Creating dictionaries with 7 values
dic_1 = {'Toronto': 500, 'Calgary': 200, 'Vancouver': 300,
'Montreal': 700, 'Ottawa': 400, 'Quebec': 600, 'Edmonton': 800}
# Creating pandas Series from the dictionary
ser1 = pd.Series(dic_1)
```

```
# Printing the entire series using head() (default shows first 5
elements)
print(f"The entire series (ser1) using head() is:\n{ser1.head()}")
# Printing the first 2 elements using head()
print(f"\nThe first 2 elements of the series (ser1) using head()
are:\n{ser1.head(2)}")

# Printing the entire series using tail() (default shows last 5
elements)
print(f"\nThe entire series (ser1) using tail()
is:\n{ser1.tail()}")
# Printing the last 2 elements using tail()
print(f"\nThe last 2 elements of the series (ser1) using tail()
are:\n{ser1.tail(2)}")
```

```
The entire series (ser1) using head() is:
Toronto      500
Calgary      200
Vancouver    300
Montreal     700
Ottawa       400
dtype: int64

The first 2 elements of the series (ser1) using head() are:
Toronto    500
Calgary    200
dtype: int64

The entire series (ser1) using tail() is:
Vancouver    300
Montreal     700
Ottawa       400
Quebec       600
Edmonton     800
dtype: int64

The last 2 elements of the series (ser1) using tail() are:
Quebec      600
Edmonton    800
dtype: int64
```

ব্যাখ্যা:

1. **ser1.head()**: এটি সিরিজের প্রথম ৫টি উপাদান (ডিফল্ট) দেখায়। এখানে, head() ফাংশনটি শহরের নাম এবং তাদের মান প্রথম ৫টি উপাদান হিসেবে দেখাবে।
2. **ser1.head(2)**: এটি প্রথম ২টি উপাদান দেখাবে।
3. **ser1.tail()**: এটি সিরিজের শেষ ৫টি উপাদান (ডিফল্ট) দেখাবে।
4. **ser1.tail(2)**: এটি সিরিজের শেষ ২টি উপাদান দেখাবে।

উদাহরণ:

যদি ser1 সিরিজে ৭টি শহরের তথ্য থাকে, তাহলে head() প্রথম ৫টি শহর এবং tail() শেষ ৫টি শহরের মান দেখাবে। তবে, যখন সংখ্যা দেয়া হয় (যেমন head(2) বা tail(2)), তখন শুধুমাত্র প্রথম বা শেষ ২টি উপাদানই দেখানো হবে।

size

- To check the number of elements in your data.

empty

- True if the series in empty

```
import numpy as np
import pandas as pd

# Creating dictionaries with 7 values
dic_1 = {'Toronto': 500, 'Calgary': 200, 'Vancouver': 300,
'Montreal': 700, 'Ottawa': 400, 'Quebec': 600, 'Edmonton': 800}
# Creating pandas Series from the dictionary
ser1 = pd.Series(dic_1)

# Printing the size of the Series (number of elements)
print(f"The size of the series (ser1) is: {ser1.size}")

# Checking if the Series is empty (returns True for empty series)
print(f"Is the series (ser1) empty? {ser1.empty}")
```

```
The size of the series (ser1) is: 7
Is the series (ser1) empty? False
```

1. **ser1.size:** এটি সিরিজে মোট উপাদান (এখানে, শহরের নাম এবং তাদের মান) এর সংখ্যা দেখায়। ser1.size এর মান হবে ৭, কারণ এখানে ৭টি শহরের তথ্য রয়েছে।
2. **ser1.empty:** এটি চেক করে যে সিরিজটি খালি কিনা। যদি সিরিজে কোনো উপাদান না থাকে, তবে এটি True রিটার্ন করবে, অন্যথায় False রিটার্ন করবে। এখানে, ser1.empty হবে False, কারণ সিরিজে ৭টি উপাদান রয়েছে।

এটি সিরিজের মোট উপাদান সংখ্যা এবং সিরিজটি খালি কিনা তা নির্দেশ করে।

Python contain 4 type built-in array types (list, tuple, set, dictionary) সাধারণত সাধারণ ডাটা সংরক্ষণের জন্য ব্যবহৃত হয়।

NumPy array, একটি 3rd Party লাইব্রেরি যা গাণিতিক গণনা, বড় ডাটা এবং মাল্টি-ডাইমেনশনাল array পরিচালনায় ব্যবহৃত হয়, এবং এটি অনেক দ্রুত কার্যকর। install দিতে হয়।