Pandas. Data types and IO

```
pip install pandas
```

Pandas - це відкрита бібліотека для обробки та аналізу даних, яка надає потужні інструменти для роботи з даними у форматі таблиць. Вона є однією з найпопулярніших бібліотек для роботи з даними в середовищі Python.

Основними структурами даних, які надає Pandas, є два класи: DataFrame і Series.

DataFrame є основним об'єктом у Pandas і представляє собою двовимірну таблицю з даними, подібну до електронних таблиць. Вона складається з рядків та стовпців, кожен з яких може містити різні типи даних. DataFrame дозволяє виконувати різноманітні операції з даними, такі як фільтрація, сортування, обчислення статистики, об'єднання даних тощо.

Series - це одновимірна структура даних, яка представляє собою маркований масив даних. Вона може бути розглянута як один стовпець або рядок з DataFrame. Series дозволяє виконувати операції з індексами, а також забезпечує багато функцій для маніпулювання даними.

Основна функціональність Pandas включає:

1. Завантаження та збереження даних: Pandas може завантажувати дані з різних форматів, таких як CSV, Excel, SQL, JSON, HTML та інших. Також вона дозволяє зберігати дані у багатьох форматах.
2. Читання та запис даних: Pandas дозволяє зчитувати дані з файлів та баз даних, виконувати операції з ними та записувати результати назад.
3. Обробка та маніпулювання даними: Pandas надає широкі можливості для обробки даних, включаючи фільтрацію, сортування, групування, об'єднання, зведення, заповнення пропусків, обробку дат тощо.
4. Індексування та вибірка даних: Pandas дозволяє виконувати індексацію, сортування та вибірку даних з DataFrame та Series. Можна використовувати індекси на основі міток або позицій.

5. Обчислення статистики: Pandas надає функціональність для обчислення статистичних метрик, таких як середнє значення, медіана, сума, дисперсія тощо.
6. Візуалізація даних: Pandas має інтеграцію з іншими бібліотеками для візуалізації даних, такими як Matplotlib та Seaborn, що дозволяє створювати графіки, діаграми та інші візуалізації.

Pandas є потужним інструментом для обробки та аналізу даних, особливо для структурованих даних у форматі таблиць. Вона є важливим компонентом екосистеми Python для наукових обчислень та аналізу даних.

Series

Серії (Series) є одним з основних об'єктів в бібліотеці Pandas. Вони представляють собою одновимірну марковану структуру даних, яка може містити дані будь-якого типу.

Основні особливості серій в Pandas:

1. Маркування індексів: Кожен елемент в серії має унікальний індекс, який дозволяє доступатися до нього та маніпулювати даними. Індекс може бути заданий явно або створений автоматично.
2. Гнучкість у типах даних: Серії в Pandas можуть містити дані різних типів, таких як числа, рядки, булеві значення тощо. Вони також підтримують векторизовані операції, що дозволяють виконувати операції на всіх елементах серії швидко та ефективно.
3. Індексація та вибірка даних: Серії можна індексувати за допомогою числових позицій або маркованих індексів. Це дозволяє виконувати різні операції, такі як отримання підмножини даних, фільтрація, сортування та інше.
4. Векторизація операцій: Операції над серіями можуть бути виконані векторизовано, що дозволяє ефективно обробляти дані. Це означає, що ви можете застосовувати функції та операції до всіх елементів серії одночасно без необхідності використовувати цикли.
5. Підтримка відсутності даних: Серії в Pandas можуть містити пропущені значення, які позначаються як NaN (Not a Number). Pandas надає

функції для роботи з пропущеними значеннями, такі як виявлення, видалення або заповнення цих значень.

6. Операції агрегації та статистики: Pandas надає широкий набір функцій для обчислення різних статистичних метрик над серіями, таких як сума, середнє значення, медіана, мінімум, максимум тощо.

Серії використовуються для представлення одного стовпця або рядка даних в таблиці, а також для виконання операцій над окремими стовпцями або рядками. Вони є потужним інструментом для маніпулювання та аналізу даних у бібліотеці Pandas.

1. Створення серії зі списку:

```
import pandas as pd

data = [10, 20, 30, 40, 50]
series = pd.Series(data)
print(series)
```

Результат:

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

1. Створення серії зі словника:

```
import pandas as pd

data = {'A': 10, 'B': 20, 'C': 30}
series = pd.Series(data)
print(series)
```

Результат:

```
A    10
B    20
C    30
dtype: int64
```

1. Індексація та вибірка даних з серії:

```
import pandas as pd

data = [10, 20, 30, 40, 50]
series = pd.Series(data, index=['A', 'B', 'C', 'D', 'E'])
print(series['B'])
print(series[1:4])
```

Результат:

```
20
B    20
C    30
D    40
dtype: int64
```

1. Виконання операцій над серіями:

```
import pandas as pd

data1 = [10, 20, 30]
data2 = [1, 2, 3]
series1 = pd.Series(data1)
series2 = pd.Series(data2)

sum_series = series1 + series2
multiply_series = series1 * series2

print(sum_series)
print(multiply_series)
```

Результат:

```
0    11
1    22
2    33
dtype: int64
0    10
1    40
2    90
dtype: int64
```

Це лише кілька прикладів використання серій в Pandas. За допомогою серій ви можете виконувати багато інших операцій, таких як фільтрація, сортування, агрегація даних, обробка пропущених значень тощо. Pandas надає широкий набір функцій для роботи з серіями, які допомагають зручно та ефективно аналізувати дані.

DataFrame

Pandas DataFrames є одним з основних об'єктів в бібліотеці Pandas і представляють собою двовимірну структуру даних у форматі таблиці. DataFrame складається з рядків та стовпців, і кожен стовпець може містити дані різних типів. Він надає потужні функції для маніпулювання та аналізу даних.

Основні особливості Pandas DataFrame:

1. Створення DataFrame:

Можна створити DataFrame зі списку, словника, NumPy-масиву або іншого DataFrame. Ось приклади:

```
import pandas as pd

# Зі списку
data = [['John', 25], ['Alice', 30], ['Bob', 35]]
df = pd.DataFrame(data, columns=['Name', 'Age'])

# Зі словника
```

```
data = {'Name': ['John', 'Alice', 'Bob'], 'Age': [25, 30, 3
5]}
df = pd.DataFrame(data)

# З NumPy-масиву
import numpy as np
data = np.array([[1, 2, 3], [4, 5, 6]])
df = pd.DataFrame(data, columns=['A', 'B', 'C'])
```

1. Індексація та вибірка даних:

DataFrame надає різні способи індексації та вибірки даних. Можна використовувати індексацію на основі міток або позицій. Ось приклади:

```
# За допомогою міток
df['Name']  # Вибірка стовпця по назві
df.loc[0]  # Вибірка рядка за індексом

# За допомогою позицій
df.iloc[0]  # Вибірка рядка за позицією
df.iloc[0:3, 1:3]  # Вибірка підмножини даних
```

1. Операції з даними:

DataFrame дозволяє виконувати різні операції з даними, такі як сортування, фільтрація, групування, об'єднання, заповнення пропусків тощо. Ось декілька прикладів:

```
df.sort_values('Age')  # Сортування за стовпцем
df[df['Age'] > 30]  # Фільтрація за умовою
df.groupby('City').mean()

  # Групування та обчислення середнього значення
df.merge(other_df, on='ID')  # Об'єднання з іншим DataFrame
df.fillna(0)  # Заповнення пропусків значенням 0
```

1. Візуалізація даних:

DataFrame можна візуалізувати за допомогою бібліотек, таких як Matplotlib або Seaborn, для створення графіків, діаграм, теплових карт тощо. Ось

приклад:

```
import matplotlib.pyplot as plt

df.plot(kind='bar', x='Name', y='Age')
plt.show()
```

Це лише кілька основних операцій з Pandas DataFrame. Бібліотека Pandas надає багато інших функцій для роботи з DataFrame, таких як обрізка даних, перетворення типів, виконання операцій з датами та багато іншого. Вона є потужним інструментом для обробки, аналізу та маніпулювання табличними даними.

Read CSV

У бібліотеці Pandas є функція `read_csv()`, яка дозволяє читати дані з файлу CSV і створювати DataFrame. Файл CSV (Comma-Separated Values) є текстовим файлом, в якому дані розділені комами.

Ось приклад використання функції `read_csv()`:

```
import pandas as pd

df = pd.read_csv('data.csv')
```

У цьому прикладі ми передаємо шлях до файлу CSV `"data.csv"` у функцію `read_csv()`. Функція автоматично прочитає дані з файлу та створить DataFrame `df`.

Функція `read_csv()` має багато параметрів, які можна налаштувати в залежності від потреб. Ось кілька корисних параметрів:

- `sep`: Вказує роздільник, за замовчуванням використовується кома. Наприклад, `sep=';'` для файлів, де дані розділені крапкою з комою.
- `header`: Вказує, який рядок у файлі CSV слід вважати рядком заголовка. За замовчуванням вважається перший рядок.
- `index_col`: Вказує, який стовпець слід використовувати як індекс. Наприклад, `index_col=0` для використання першого стовпця як індексу.
- `usecols`: Вказує, які стовпці мають бути включені у DataFrame. Наприклад, `usecols=['Name', 'Age']` для включення лише стовпців 'Name' і

'Age'.

Це лише кілька параметрів, які можна використовувати. Докладну інформацію про параметри `read_csv()` можна знайти в документації Pandas.

Після читання CSV-файлу в DataFrame `df`, ви можете використовувати різні функції та операції Pandas для обробки та аналізу даних у таблиці.