Введение в Pandas

Установка библиотеки

Чтобы установить библиотеку pandas достаточно прописать в консоли:

```
pip install pandas
```

Основы pandas

Библиотека pandas - это мощный инструмент для анализа данных и манипуляций с ними в языке программирования Python. Она предоставляет высокопроизводительные и простые в использовании структуры данных, такие как DataFrame, для обработки и анализа табличных данных.

В pandas существует две основные структуры данных:

- `Series` - 1D однородно-типизированный массив. Можно выполнять все те же операции, что и с векторами.
- `DataFrame` - 2D, изменяемая по размеру табличная структура с потенциально разнородно типизированным столбцами.

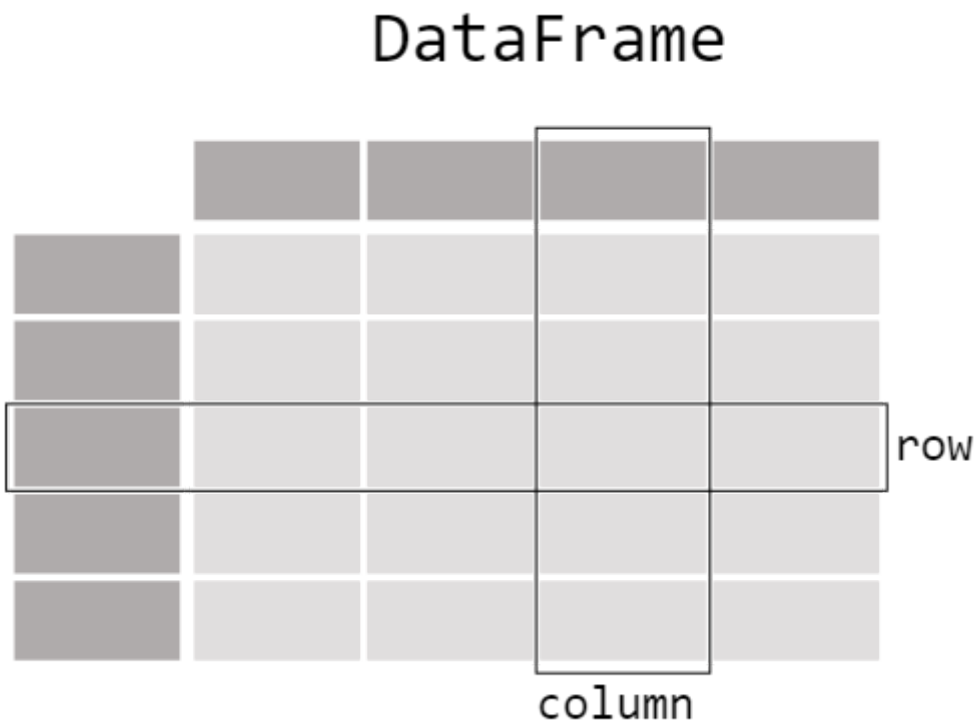
Чтобы использовать pandas достаточно импортировать:

```
import pandas as pd
```

Pandas является надстройкой на Numpy, поэтому у них похожий синтаксис и большинство методов и функций numpy можно применять к `Series` и `DataFrame`.

`DataFrame`

Как упоминалось выше `DataFrame` это таблица, в которой столбцы могут иметь различные типы данных. Пройдёмся по основам взаимодействия с ней.



Схематичное изображение `DataFrame`

Пример создания и вывода таблицы в Jupyter notebook:

```
In [1]: df = pd.DataFrame(
   ...:     {
   ...:         "Name": [
   ...:             "Braund, Mr. Owen Harris",
   ...:             "Allen, Mr. William Henry",
   ...:             "Bonnell, Miss. Elizabeth",
   ...:         ],
   ...:         "Age": [22, 35, 58],
   ...:         "Sex": ["male", "male", "female"],
   ...:     }
   ...: )
   ...:

In [2]: df
Out[2]:
                        Name  Age     Sex
0   Braund, Mr. Owen Harris   22    male
1  Allen, Mr. William Henry   35    male
2  Bonnell, Miss. Elizabeth   58  female
```

Каждый столбец таблицы - это `Series` и получить столбец можно различными способами:

```
In [3]: # Возьмём столбец Age для примера.
   ...: col = df["Age"]
```

```
   ...: # Также его можно взять с помощью loc
   ...: col = df.loc[:, "Age"]
   ...: # Или по индекса с помощью iloc
   ...: col = df.iloc[:, 1]
   ...: col
Out[3]:
0    22
1    35
2    58
```

Конкретный элемент, как `DataFrame` можно взять по индексу или метке ("Age" например метка) используя свойство `.iloc[i_index, j_column]` или `.loc[index, column]`.

```
In [4]: df.loc[0, "Name"]
Out[4]: 'Braund, Mr. Owen Harris'

In [5]: df.iloc[0, 0]
Out[5]: 'Braund, Mr. Owen Harris'
```

Также в pandas можно делать срезы и накладывать булевые маски:

```
In [6]: df.loc[:1, ["Name", "Sex"]]
Out[6]:
                        Name  Sex
0  Braund, Mr. Owen Harris male
1 Allen, Mr. William Henry male

In [7]: df.iloc[:2, [0, 2]]
Out[7]:
                                     Name  Sex
0  Braund, Mr. Owen Harris male
1 Allen, Mr. William Henry male

In [8]: df[df["Age"] > 30]
Out[8]:
                                        Name  Sex
0  Braund, Mr. Owen Harris male
1 Allen, Mr. William Henry male
```

Series

`Series` как `ndarray` имеет различные методы, вроде `sum`, `min`, `max`, `mean`, `std` и т.д. (можно посчитать сразу все эти свойства пользуясь методом `describe`). Также `Series` доступны математические операции (`+`, `-`, `*`, `/`,...) или логические операции (`<`, `>`, `==`,...) работающие поэлементно.

```
In [9]: df["Age"] + df["Age"] + 10 == df["Age"] * 2 + 10
Out[9]:
0 True
1 True
2 True
Name: Age, dtype: bool

In [10]: df["Age"].mean()
Out[10]: 38.333333333333336
```

Чтобы привести `Series` к другому формату данных достаточно воспользоваться методом `astype`:

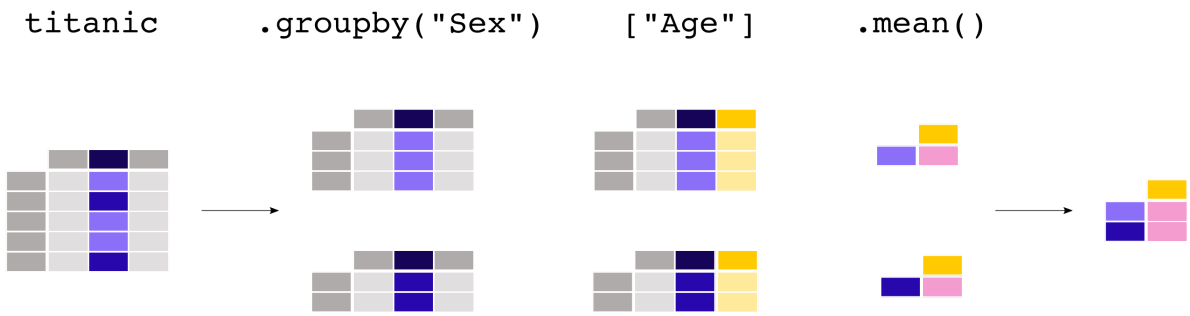
```
In [11]: pd.Series([1, 0, 1]).astype(bool)
Out[11]:
0 True
1 False
2 True
dtype: bool
```

Group by

Вычисление статистики (например, среднего возраста) для каждой категории столбца (например, мужчина/женщина в столбце Age) является обычной задачей. Метод `groupby` используется для поддержки этого типа операций. Это вписывается в более общий шаблон разделения-применения-комбинирования:

- **Разделить** данные на группы
- **Применить** функцию к каждой группе независимо
- **Объединить** результат в структуре данных
  Шаги "применить" и "объединить" обычно выполняются одновременно в pandas.

```
In [12]: df.groupby("Sex")["Age"].mean()
Out[12]:
Sex
female 58.0
male 28.5
Name: Age, dtype: float64
```



Поэтапная иллюстрация кода выше (titanic=df).

Вычисление количества элементов категории

Иногда требуется подсчитать количество каждого уникального элемента `Series`. Для этого применяется метод `value_counts`.

```
In [13]: df["Sex"].unique()
Out[13]: array(['male', 'female'], dtype=object)

In [14]: df["Sex"].value_counts()
Out[14]:
male 2
female 1
Name: Sex, dtype: int64
```

Дополнительно

Рассмотрим часто используемые методы и свойства для взаимодействия с таблицами:

- `sort_index()` - сортирует по индексам.
- `sort_values()` - сортирует по значениям.
- `shape` - содержит информацию о форме `DataFrame` / `Series`.
- `T` - содержит транспонированную таблицу.
- `index` - содержит индексы.
- `columns` - содержит названия столбцов.

Заключение

Мы познакомились с основными методами и классами pandas, знание которых необходимо для дальнейшего изучения курса. В случае, если представленной информации недостаточно, следует воспользоваться официальной документацией pandas.