

Яндекс

Data Science. Pandas

Московская Школа Программистов

Pandas



Pandas — высокоуровневая Python библиотека для анализа данных, использующая низкоуровневое API языка C.

Pandas является стандартом индустрии анализа данных

Типы данных



Данные

Одномерные

Series

Двумерные

DataFrame

Создание Series

```
sr = pd.Series([1,2,3,4,5]) #Создание Series из списка
```

```
# Изменение индекса
```

```
sr = pd.Series([5.0,4.0,3.0], index = ['Vasya', 'Petya', 'Kolya'])
```

```
# Создание Series из словаря
```

```
sr = pd.Series(  
    { 'Vasya ': 5.0,  
      'Petya': 4.0  
    })
```

Доступ к элементам Series

`t = sr[0]` # Доступ по нумерованному индексу

`t = sr['Vasya']` # Доступ по своему индексу

`t = sr.Vasya` # Так тоже можно

`t = sr.loc['Vasya']` # Эквивалент строкам сверху

`t = sr.iloc[0]` # Эквивалент первой строке сверху



| у Series есть индекс.
Индекс может быть
многомерным

Основные функции

`sr.apply(len)` # Применение функции к каждому эл-ту

`sr.sum()` # Сумма элементов

`sr.any(func)` # Первый эл-т, подходящий

`sr = sr + 5` # Добавить к каждому элементу 5

`sr = sr * 5` # Умножить каждый элемент на 5

Основные функции

`sr.isin(list)` # Есть ли элемент в списке

`sr.mean()` # Среднее значение

`sr.astype(dtype)` # Каждый элемент привести к типу

`sr.value_counts()` # Посчитать количество каждого элемента

DataFrame

DataFrame — это набор Series. Можно сказать, что это таблица, где каждая строка и каждый столбец — Series.

Доступ можно производить как к строкам, так и к столбцам.

Создание DataFrame

```
df = pd.read_csv('data.csv') # Чтение из CSV файла
```

```
df = pd.DataFrame([{"price": 3, "count": 8}, {"price": 4, "count": 11}])  
# Создание из списка словарей
```

```
df = pd.DataFrame({"price": [3,4], «count»:[8,11]}) # Создание из словаря
```

```
df = pd.DataFrame([[3,4],[8,11]]) # Создание из двумерного списка
```

Основные методы

`df.shape` # Вывод количества строк и столбцов(кортеж)

`df.columns` # Возвращает Index(количество столбцов)

`df.info()` # Возвращает общую информацию о таблице

`df.head(n)` # Возвращает первые n строчек(default — 5)

`df.tail(n)` # Возвращает последние n строчек(default — 5)

Доступ к элементам

К столбцам	К строкам
<code>df['count']</code>	<code>df.iloc[0]</code>
<code>df.count</code>	<code>df.loc['Moscow']</code>

Выборка нескольких строк

Код	Описание
<code>df[0:5]</code> или <code>df.iloc[0:5]</code>	Выбрать первые 5 строк
<code>df['name','city']</code>	Выбрать только столбцы name и city
<code>df['name':'city']</code>	Выбрать столбцы с name по city
<code>df.loc[0:3,['name','city']]</code> <code>df[['name','city']][0:3]</code>	Выбрать первые 3 строки, поля name и city

Фильтрация данных

`df[df['students_num'] > 200]` или `df[df.students_num > 200]`

Python	Pandas
or	
and	&
not	~

| Фильтрация в Pandas
происходит по `boolean`
`vector`

Пример — найти строки, у которых значения столбца «city» начинается с буквы М

```
df[df.city.str.startswith('М')] или  
cond = df.city.apply(lambda x:x.startswith('М'))  
df[cond]
```

Сортировка и группировка



Сортировка

sort_values

Синтаксис — `df.sort_values(by=<название столбца или список>, ascending = <True/False или список из True/False>)`

Отсортировать по столбцу **rating** по *убыванию*

`df.sort_values(by='rating',ascending=False)`

Отсортировать по столбцу **rating** и **age**, рейтинг — по убыванию, возраст — по возрастанию.

`df.sort_values(by=['rating','age'], ascending = [False,True])`

Группировка

groupby

Сгруппировать данные по значению столбца и применить функцию

Синтаксис — `df.groupby(<столбец_группировки>)`
`[<столбцы_показа>].функция`