УДК 004.423 **Информационные технологии**

Дрянкова Дарья Александровна, *студент факультета информатики и вычислительной техники, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

Замулин Иван Сергеевич, *заведующий кафедрой ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

СОЗДАНИЕ ТАБЛИЦ И ИХ ФИЛЬТРАЦИЯ В БИБЛИОТЕКЕ PANDAS ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Аннотация: В статье рассматриваются способы создания таблиц библиотеки Pandas языка программирования Python и их выборка и фильтрация.

Ключевые слова: Python, Pandas, таблицы данных, строки, DataFrame, фильтрация.

Annotation: The article discusses ways to create tables of the Pandas library of the Python programming language and their selection and filtering.

Keywords: Python, Pandas, data tables, strings, DataFrame, filtering.

Библиотека Pandas - это библиотека для работы с данными в формате таблиц на языке программирования Python. Она позволяет считывать, записывать и обрабатывать данные из различных источников, таких как файлы CSV, Excel, базы данных SQL и другие.

В возможности библиотеки Pandas включены:

1) Создание и работа с таблицами данных: Pandas позволяет создавать таблицы данных, называемые DataFrame, которые представляют собой двумерные структуры данных с именованными столбцами и строками. DataFrame позволяет удобно и эффективно хранить, и обрабатывать большие объемы данных.

2) Выборка и фильтрация данных: Pandas предоставляет широкий набор инструментов для выборки и фильтрации данных из таблиц. Это включает в себя выбор столбцов и строк по определенным критериям, а также фильтрацию данных по определенным условиям [1].

Способы создания таблиц в Pandas:

a) Создание таблицы из файла: Pandas может загружать данные из различных источников, включая CSV, Excel, JSON, SQL, HTML и другие форматы файлов. Например, для загрузки таблицы из файла CSV можно использовать метод read_csv().

b) Создание таблицы вручную: Pandas также позволяет создавать таблицы вручную с помощью функций, таких как DataFrame() или Series(). Эти функции позволяют задавать значения столбцов и строк таблицы и определять типы данных.

Рассмотрим некоторые из этих способов:

На рисунке 1 изображен пример загрузки данных из файла в формате CSV:

```
import pandas as pd

df = pd.read_csv("file.csv")
```

Рисунок 1 – загрузка данных из файла в формате CSV

На рисунке 2 изображен пример загрузки данных из базы данных MySQL:

```
import pandas as pd
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="username",
  password="password",
  database="database"
)

query = "SELECT * FROM table"
df = pd.read_sql(query, mydb)
```

Рисунок 2 – загрузка данных из базы данных MySQL

На рисунке 3 представлен пример создания таблицы вручную с помощью функций Pandas:

```
>>> import pandas as pd
>>> df = pd.DataFrame({
...     'Name': ['Alice', 'Bob', 'Charlie'],
...     'Age': [25, 30, 35],
...     'City': ['New York', 'London', 'Paris']
... })
>>> df
      Name  Age      City
0    Alice   25  New York
1      Bob   30    London
2  Charlie   35     Paris
```

Рисунок 3 – Создание таблицы вручную с помощью функций Pandas

На рисунке 4 изображен пример создания таблицы данных с помощью массива NumPy:

```
>>> import pandas as pd
>>> import numpy as np
>>> data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> df = pd.DataFrame(data, columns=['A', 'B', 'C'])
>>> df
   A  B  C
0  1  2  3
1  4  5  6
2  7  8  9
```

Рисунок 4 – Создание таблицы данных с помощью массива NumPy

На рисунке 5 представлен пример создания таблицы из списка словарей:

```
>>> import pandas as pd
>>> data = [{'Name': 'Alice', 'Age': 25, 'City': 'New York'},
...         {'Name': 'Bob', 'Age': 30, 'City': 'London'},
...         {'Name': 'Charlie', 'Age': 35, 'City': 'Paris'}]
>>> df = pd.DataFrame(data)
>>> df
      Name  Age      City
0    Alice   25  New York
1      Bob   30    London
2  Charlie   35     Paris
```

Рисунок 5 – Создание таблицы из списка словарей

Как было указано ранее, библиотека Pandas имеет широкий набор

инструментов для выборки и фильтрации данных из таблиц, рассмотрим примеры некоторых из них:

На рисунке 6 изображен пример выбора строк по значению в столбце, где выбрали все значения со значением city == 'LA'.

```
>>> import pandas as pd
>>>
>>> data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
...         'age': [25, 32, 18, 47],
...         'city': ['NY', 'LA', 'SF', 'LA']}
>>> df = pd.DataFrame(data)
>>>
>>> df_la = df[df['city'] == 'LA']
>>> print(df_la)
    name  age city
1    Bob   32   LA
3  David   47   LA
```

Рисунок 6 – Выбор строк по значению в столбце

На рисунке 7 изображен пример выбора строк по нескольким значениям, в качестве значений для фильтрации выбираются строки со значениями, где где age <= 30 и city == 'NY'.

```
>>> df = pd.DataFrame(data)
>>> df_la_30 = df[(df['age'] <= 30) & (df['city'] == 'NY')]
>>> print(df_la_30)
    name  age city
0  Alice   25   NY
```

Рисунок 7 – Выбор строк по нескольким значениям

На рисунке 8 изображен выбор строк по частичному совпадению в столбце, в данном примере выбираются строки, где name содержит букву 'a'.

```
>>> import pandas as pd
>>> data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
...         'age': [25, 32, 18, 47],
...         'city': ['NY', 'LA', 'SF', 'LA']}
>>> df = pd.DataFrame(data)
>>>
>>> df_a = df[df['name'].str.contains('a')]
>>> print(df_a)
      name  age city
2  Charlie   18   SF
3    David   47   LA
```

Рисунок 8 – Выбор строк по частичному совпадению

Также мы можем выбирать строки по индексу, а столбцы по названию, как представлено на рисунке 9.

```
>>> import pandas as pd
>>> data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
...         'age': [25, 32, 18, 47],
...         'city': ['NY', 'LA', 'SF', 'LA']}
>>> df = pd.DataFrame(data)
>>>
>>> #выбор строк по индексу
>>> df_1 = df.loc[1]
>>> print(df_1)
name    Bob
age      32
city     LA
Name: 1, dtype: object
>>> #выбор столбцов по названию
>>> df_name = df['name']
>>> print(df_name)
0      Alice
1        Bob
2    Charlie
3      David
Name: name, dtype: object
```

Рисунок 9 – Выбор строк по индексу и выбор столбцов по названию

Преимущества библиотеки Pandas:

1) Интуитивный интерфейс: Pandas обладает очень удобным и интуитивным интерфейсом для работы с таблицами, благодаря чему программистам гораздо проще понимать, как она работает, и использовать ее для своих задач.

2) Широкий функционал: Pandas предоставляет множество функций для работы с данными, таких как выборка и фильтрация, группировка, объединение таблиц и преобразование данных, что делает ее очень мощной и универсальной библиотекой.

3) Простота использования: Благодаря интуитивному интерфейсу и широкому функционалу, Pandas позволяет программистам легко и быстро решать задачи работы с данными.

4) Широкое сообщество: Pandas имеет широкое сообщество

пользователей и разработчиков, которые создают множество дополнительных инструментов и библиотек для улучшения ее функционала и расширения возможностей.

Недостатки библиотеки Pandas:

1) Производительность: При работе с большими объемами данных Pandas может быть не слишком эффективной и занимать много времени на выполнение операций.

2) Потребление памяти: Pandas может потреблять много памяти для хранения таблиц и данных, особенно при работе с большими объемами данных.

3) Сложность: Pandas может быть сложна для понимания и использования для начинающих программистов, особенно при работе с более сложными операциями.

Заключение

В сравнении с другими библиотеками для работы с данными, Pandas обладает уникальным комбинацией преимуществ и недостатков, которые делают ее хорошим выбором для многих задач работы с данными, но не всегда оптимальным решением для конкретных задач. Например, для работы с большими объемами данных может быть более эффективным использование специализированных библиотек, таких как Apache Spark. В то же время, для более простых задач работы с данными Pandas является лучшим выбором благодаря своей простоте и удобству использования [2].

Библиографический список:

1. Гэддис Т. Начинаем программировать на Python [Текст] / Т. Гэддис. – СПб.: БХВ-Петербург, 2021. – 768 с. ISBN: 978-5-9775-4002-5.

2. Лусиану Рамальо, Слинкин А. А., Python. К вершинам мастерства. Лаконичное и эффективное программирование [Текст] / Лусиану Рамальо. – М.: ДМК-Пресс, 2022 г. – 898 с. ISBN: 978-5-97060-885-2.