



МИНОБРНАУКИ
РОССИИ



УНИВЕРСИТЕТ
ИННОПОЛИС



онлайн
университет

Программная инженерия. Разработка ПО (Python для продвинутых специалистов. Машинное обучение)

Модуль: Введение в Python для машинного обучения

Лекция 3: Библиотеки Numpy, Pandas

Дата: 31.03.2024

Q&A



Содержание лекции

- Работа с файлами в Python
- Модули
- Введение в Numpy
- Введение в Pandas
- Практическая часть



Работа с текстовым файлом. Методы

Метод	Значение
<code>open(file, [mode])</code>	Открывает файл <code>file</code> в режиме <code>mode</code>
<code>read(n)</code>	Считывает <code>n</code> символов из файла
<code>readline()</code>	Считывает очередную строку из файла
<code>readlines()</code>	Считывает все строки из файла и создает из них список
<code>write(s)</code>	Записывает строку <code>s</code> в файл на то место, где стоит каретка
<code>seek(n)</code>	Переставляет каретку <code>n</code> -ный байт файла
<code>close()</code>	Закрывает файл

```
f = open( "<путь к файлу>", "<режим открытия BSD - like>" ) # открываем файл
# делаем какую-то работу с файлом
f.close() # закрываем файл
# ИЛИ
with open( "<путь к файлу>", "<режим открытия BSD - like>" ) as f:
    # делаем какую-то работу с файлом
# f будет доступен внутри конструкции with
# после выхода из конструкции файл будет закрыт
```

Режимы открытия текстового файла

- **"r"** Открыть текстовый файл для чтения. Курсор будет в начале файла.
- **"r+"** Открыть для чтения и записи. Курсор в начале файла.
- **"w"** Открыть для записи. Файл обрезается до начала или создается, если ранее не существовал.

Курсор в начале файла.

- **"w+"** Открыть для чтения и записи. Файл обрезается до начала или создается, если ранее не существовал. Курсор в начале файла.
- **"a"** Открыть для записи. Файл создается, если ранее не существовал. Курсор в конце файла (добавление новой информации в конец файла).
- **"a+"** Открыть для чтения и записи. Файл создается если ранее не существовал. Курсор в конце файла (добавление новой информации в конец файла).



Передовые
инженерные
школы



МИНОБРНАУКИ
РОССИИ



УНИВЕРСИТЕТ
ИННОПОЛИС



онлайн
университет

Модули

Глобально, модули можно разделить на 2 простые категории:

- предустановленные (идут вместе с языком);
- сторонние (можно установить при необходимости) - <https://pypi.org/>

Примеры предустановленных модулей:

- `random` – модуль для генерации случайных чисел и последовательностей;
- `math` – математические функции;
- `json` – модуль для работы с .json файлами;
- `csv` – модуль для чтения .csv файлов (и в целом .csv like файлов).

Список тут: <https://docs.python.org/3/py-modindex.html>

Установка сторонних модулей возможна с помощью утилиты `pip`.

Команды:

- `pip list` – покажет все установленные модули и их версии (`pip freeze` тоже)
- `pip install --upgrade pip` – обновляет сам пакетный менеджер
- `pip install <package>` – устанавливает пакет `<package>` в текущее окружение
- `pip uninstall <package>` – удаляет пакет `<package>` из текущего окружения



Передовые
инженерные
школы



МИНОБРНАУКИ
РОССИИ



УНИВЕРСИТЕТ
ИННОПОЛИС



онлайн
университет

NumPy



NumPy: вводная информация

NumPy - библиотека для работы с многомерными массивами и матрицами.

Зачем нужен NumPy:

- Эффективная работа с массивами и матрицами.
- Быстрое выполнение математических операций благодаря реализации на языке C.
- Основа для многих других библиотек, таких как Pandas и SciPy.



NumPy

```
import numpy as np

# Создание массива
arr = np.array([1, 2, 3, 4, 5])
print("Массив:", arr)
```

- **Одномерные массивы:** Создание массива из списка
- **Двумерные массивы:** Создание массива из вложенных списков
- **Многомерные массивы:** Создание массива с более чем двумя измерениями

```
import numpy as np

# Одномерный массив
one_d_array = np.array([1, 2, 3, 4, 5])
print("Одномерный массив:", one_d_array)

# Двумерный массив
two_d_array = np.array([[1, 2, 3], [4, 5, 6]])
print("Двумерный массив:", two_d_array, sep="\n")

# Многомерный массив
multi_d_array = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("Многомерный массив:", multi_d_array, sep="\n")
```

Основные операции с массивами

- **Арифметические операции:** Сложение, вычитание, умножение и деление массивов
- **Универсальные функции:** Синус, косинус, экспонента и т.д.
- **Агрегационные функции:** Сумма, среднее, минимум, максимум по измерениями

```
import numpy as np

# Создание массивов
a = np.array([1, 2, 3, 4, 5])
b = np.array([5, 4, 3, 2, 1])

# Арифметические операции
print("Сложение:", a + b)
print("Вычитание:", a - b)
print("Умножение:", a * b)
print("Деление:", a / b)

# Универсальные функции
print("Синус:", np.sin(a))
print("Экспонента:", np.exp(a))

# Агрегационные функции
print("Сумма:", np.sum(a))
print("Среднее:", np.mean(a))
print("Минимум:", np.min(a))
print("Максимум:", np.max(a))
```

Индексирование и срезы



```
import numpy as np

# Одномерный массив
a = np.array([1, 2, 3, 4, 5])
print("Элемент с индексом 2:", a[2])
print("Срез [1:4]:", a[1:4])

# Двумерный массив
b = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Элемент [1,2]:", b[1, 2])
print("Строка с индексом 1:", b[1, :])
print("Столбец с индексом 2:", b[:, 2])

# Многомерный массив
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("Элемент [1,1,1]:", c[1, 1, 1])
print("Срез[:, 1, :]:", c[:, 1, :], sep="\n")
```

Изменение формы массивов

- **Reshape:** Изменение формы массива без изменения данных.
- **Ravel:** Преобразование многомерного массива в одномерный.
- **Transpose:** Транспонирование массива (перестановка осей).

```
import numpy as np

# Создание двумерного массива
a = np.array([[1, 2, 3], [4, 5, 6]])

# Изменение формы массива
b = a.reshape((3, 2))
print("Измененная форма:", b, sep="\n")

# Преобразование в одномерный массив
c = a.ravel()
print("Одномерный массив:", c)

# Транспонирование массива
d = a.transpose()
print("Транспонированный массив:", d, sep="\n")
```

Сравнение производительности NumPy и списков Python

```
import numpy as np
import time

# Сравнение производительности сложения элементов в списках и массивах NumPy
size = 1000000

# Создание списков и массивов
list1 = list(range(size))
list2 = list(range(size))
array1 = np.array(list1)
array2 = np.array(list2)

# Сложение элементов в списках
start_time = time.time()
result_list = [x + y for x, y in zip(list1, list2)]
print("Сложение списков:", time.time() - start_time, "секунд")

# Сложение элементов в массивах NumPy
start_time = time.time()
result_array = array1 + array2
print("Сложение массивов NumPy:", time.time() - start_time, "секунд")
```



Передовые
инженерные
школы



МИНОБРНАУКИ
РОССИИ



УНИВЕРСИТЕТ
ИННОПОЛИС



онлайн
университет

Pandas

Pandas: вводная информация

Pandas - библиотека для работы с данными и их анализа.

Зачем нужен Pandas:

- Упрощает работу с табличными данными.
- Интеграция с другими библиотеками, такими как NumPy и Matplotlib.
- Множество встроенных функций для анализа данных.

```
import pandas as pd

# Создание серии данных
series = pd.Series([1, 2, 3, 4, 5])
print(series)

# Создание DataFrame
data = {
    'Имя': ['Анна', 'Борис', 'Виктор'],
    'Возраст': [25, 30, 35],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань']
}
dataframe = pd.DataFrame(data)
print(dataframe)
```

Pandas



Определение Series

- Одномерная структура данных, похожая на список или массив NumPy
- Имеет метки (индексы) для каждого элемента, что делает ее похожей на словарь.
- Может содержать любые типы данных: числа, строки, объекты Python.

```
import pandas as pd

# Создание Series из списка
list_series = pd.Series([10, 20, 30, 40, 50])
print("Series из списка:", list_series, sep="\n")

# Создание Series из массива NumPy
import numpy as np
array_series = pd.Series(np.array([10, 20, 30, 40, 50]))
print("Series из массива NumPy:", array_series, sep="\n")

# Создание Series из словаря
dict_series = pd.Series({'a': 10, 'b': 20, 'c': 30})
print("Series из словаря:", dict_series, sep="\n")
```

Основные операции с Series

```
import pandas as pd

# Создание Series
series = pd.Series([10, 20, 30, 40, 50])

# Доступ к элементам
print("Элемент с индексом 1:", series[1])
print("Элемент с меткой 2:", series.iloc[2])

# Фильтрация данных
filtered_series = series[series > 25]
print("Фильтрация элементов больше 25:", filtered_series, sep="\n")

# Арифметические операции
print("Сумма всех элементов:", series.sum())
print("Среднее значение:", series.mean())
print("Умножение всех элементов на 2:", series * 2, sep="\n")
```

Определение DataFrame

DataFrame – это:

- Двумерная структура данных, аналогичная таблице или электронной таблице.
- Содержит упорядоченные коллекции столбцов, каждый из которых может иметь разный тип данных.
- Каждый столбец в DataFrame — это Series.

Создание DataFrame

```
import pandas as pd

# Создание DataFrame из словаря
data = {
    'Имя': ['Анна', 'Борис', 'Виктор'],
    'Возраст': [25, 30, 35],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань']
}
df = pd.DataFrame(data)
print("DataFrame из словаря:", df, sep="\n")

# Создание DataFrame из списка словарей
data = [
    {'Имя': 'Анна', 'Возраст': 25, 'Город': 'Москва'},
    {'Имя': 'Борис', 'Возраст': 30, 'Город': 'Санкт-Петербург'},
    {'Имя': 'Виктор', 'Возраст': 35, 'Город': 'Казань'}
]
df = pd.DataFrame(data)
print("DataFrame из списка словарей:", df, sep="\n")

# Создание DataFrame из массива NumPy
import numpy as np
data = np.array([
    ['Анна', 25, 'Москва'],
    ['Борис', 30, 'Санкт-Петербург'],
    ['Виктор', 35, 'Казань']
])
df = pd.DataFrame(data, columns=['Имя', 'Возраст', 'Город'])
print("DataFrame из массива NumPy:", df, sep="\n")
```

Основные операции с DataFrame



```
import pandas as pd

# Создание DataFrame
data = {
    'Имя': ['Анна', 'Борис', 'Виктор'],
    'Возраст': [25, 30, 35],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань']
}
df = pd.DataFrame(data)

# Доступ к данным
print("Столбец 'Возраст':", df['Возраст'], sep="\n")
print("Строка с индексом 1:", df.iloc[1], sep="\n")
print("Строки, где возраст больше 25:", df[df['Возраст'] > 25], sep="\n")

# Фильтрация данных
filtered_df = df[df['Город'] == 'Москва']
print("Фильтрация по городу Москва:", filtered_df, sep="\n")

# Агрегация данных
grouped_df = df.groupby('Город').mean()
print("Средний возраст по городам:", grouped_df, sep="\n")
```

Загрузка и сохранение данных



```
import pandas as pd

# Загрузка данных из CSV
df = pd.read_csv('sales_data.csv')
print("Загрузка данных из CSV:", df.head(), sep="\n")

# Сохранение данных в CSV
df.to_csv('output_data.csv', index=False)
print("Данные сохранены в output_data.csv")

# Загрузка данных из Excel
df = pd.read_excel('sales_data.xlsx')
print("Загрузка данных из Excel:", df.head(), sep="\n")

# Сохранение данных в Excel
df.to_excel('output_data.xlsx', index=False)
print("Данные сохранены в output_data.xlsx")
```



Передовые
инженерные
школы



МИНОБРНАУКИ
РОССИИ



УНИВЕРСИТЕТ
ИННОПОЛИС



онлайн
университет

Практическая часть

Задание 1: Работа с Excel

Прочитайте файл с данными о продажах sales.xlsx.

Вычислите общую сумму продаж и день, в который сумма продаж была максимальной.

Задание 2: Анализ спортивных данных

Используйте библиотеку NumPy для анализа данных о результатах спортивных соревнований. У вас есть массив данных с результатами бегунов на дистанцию 100 метров. Ваша задача — вычислить среднее время, медианное время, стандартное отклонение, а также найти лучшие и худшие результаты.

Задание 3: Анализ данных о недвижимости

Используйте библиотеку Pandas для анализа данных о недвижимости. У вас есть данные о домах, включая цену, количество комнат, площадь и местоположение. Ваша задача — определить среднюю цену домов в каждом районе, найти дома с самой высокой и самой низкой ценой, а также посчитать среднюю площадь домов.



Передовые
инженерные
школы



МИНОБРНАУКИ
РОССИИ



УНИВЕРСИТЕТ
ИННОПОЛИС



онлайн
университет

Спасибо за внимание