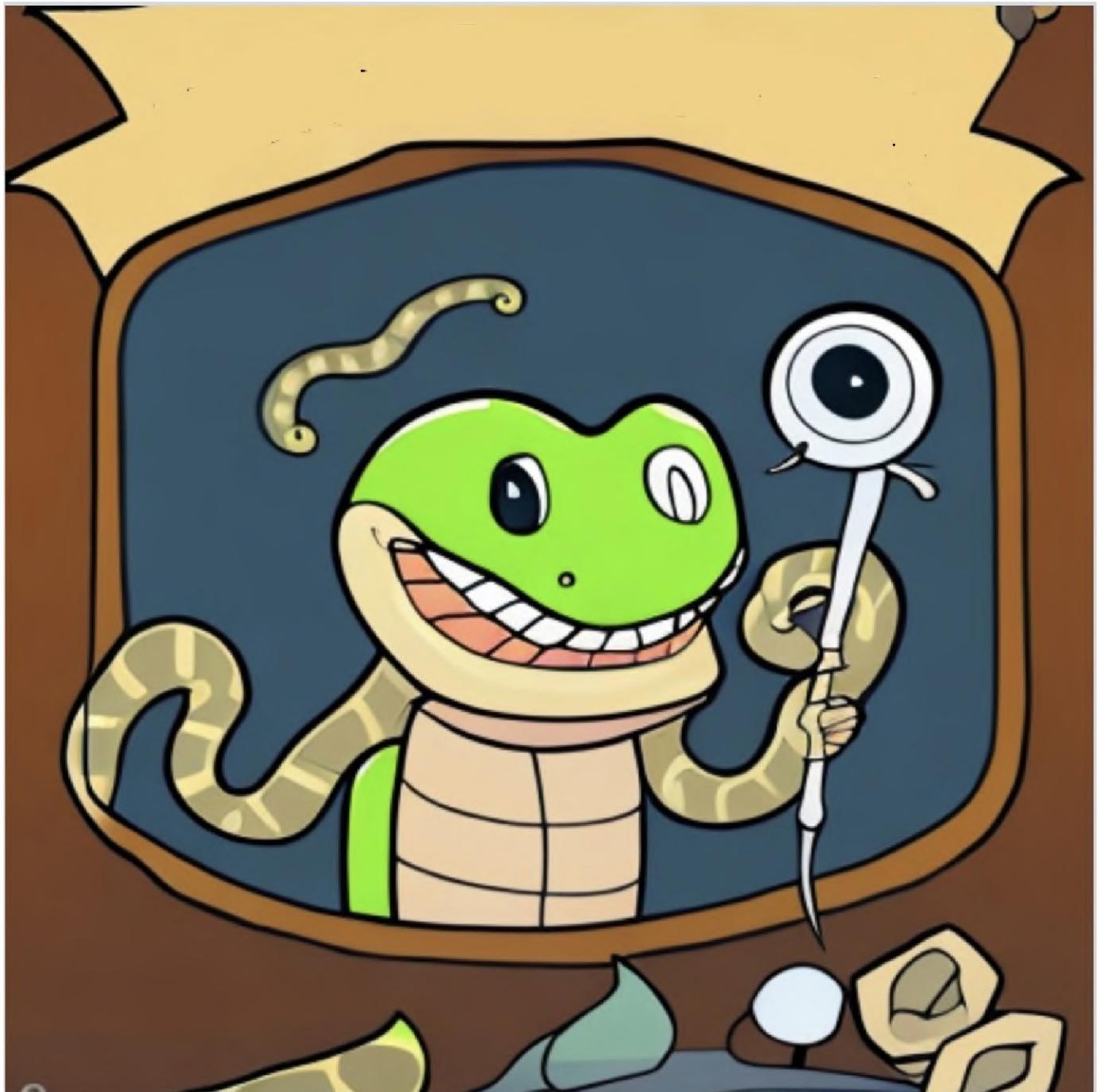


Python для OSINT. 21-дневный курс для
новичков



Для кого этот курс?

В первую очередь я провожу курс для подписчиков моего аккаунта в Твиттере. (https://twitter.com/cyb_detective), в котором я публикую твиты об OSINT (Разведка с открытым исходным кодом). Для тех, кто профессионально занимается или просто заинтересован в исследованиях и исследованиях открытых данных.

Если вы используете (или планируете использовать) инструменты OSINT, написанные на Python, но нестандартный функционал устраивает и хотелось бы его немного доработать, Этот курс поможет вам научиться делать это как можно быстрее.

Также этот курс поможет вам автоматизировать различные рутинные задачи, связанные с расследованиями: обработка данных API, сбор данных с веб-сайтов, сбор результатов поиска, работа с интернет-архивами, создание отчетов и визуализация данных.

Основная цель курса – не научить вас писать код на Python, а научить вас тратить меньше времени на рутинные OSINT-задачи. Итак, кроме того для примеров кода, я также дам вам ссылки на различные сервисы, которые помогут вы решаете разные проблемы.

Этот курс также будет полезен тем, кто далек от информатики и хотят немного повысить свой технический уровень, попробовать использовать Linux, научиться работать с командной строкой и понимать различные популярные ИТ-термины, такие как «JSON», «API», «WHOIS» и т. д.

Кому следует **избегать** этого курса?

Для тех, кто никогда не занимался OSINT и собирается заниматься OSINT. Этот курс состоит по большей части из специализированных тем, связанных с расследованием и сбор данных.

Для тех, кто хочет изучить Python, чтобы:

- стать действительно хорошим разработчиком;
- сдать экзамен для поступления в университет;
- пройти собеседование при приеме на работу.

В этом курсе опущено ОЧЕНЬ много важных вещей, а иногда даже рекомендует то, что можно было бы назвать плохой практикой.

Есть вещи это не имеет значения при написании небольших средств автоматизации для повседневных задач OSINT, но чрезвычайно важны при создании серьезных командных проектов.

Как пройти этот курс

Первое, что советую вам сделать, это посмотреть оглавление, полистать страницы книги, и четко решите, будет ли вам полезен этот курс.

Если вы приняли четкое решение, каждый день вдумчиво читайте один урок и старайтесь каждый день думать о том, как вы могли бы применить то, чему научились, в ваши расследования. Если вы пропустите день или даже неделю, пожалуйста, не ругайтесь себя за это, но просто продолжайте курс изо дня в день.

Я также рекомендую вам попробовать запустить весь пример кода и попробовать изменить что-то в этом.

Все примеры кода в книге доступны в этом репозитории —
<https://github.com/cipher387/python-for-OSINT-21-days>

Этот курс распространяется совершенно бесплатно. В начале я думал о продаже, но так как мои подписчики разбросаны по всей планете и имеют очень разные уровни доходов, я решил распространить это без ограничения.



Но чтобы укрепить вашу дисциплину и мотивировать доводить дело до конца, я рекомендую вам сделать небольшое пожертвование.

Бесплатные курсы люди часто не доходят до конца, и оплата поможет вам относиться к обучению серьезно. Кроме того, каждое пожертвование будет мотивировать меня создавать новые Курсы OSINT и сделать их доступными для людей во всем мире.

Сумму пожертвования вы определяете сами.

Например, если вы курите, то для вас цена курса может быть равна к цене пачки любимых сигарет.

Если вы пьете алкоголь, то стоимость банки пива в ближайшем супермаркете или небольшой бокал вина в ресторане на соседней улице. Если вам нравится фаст-фуд, выбирайте цену небольшого гамбургера или упаковки картошки фри.

Вы можете отправить пожертвование через банковскую карту или PayPal:
https://boosty.to/cyb_detective

Если по какой-то причине вы не хотите отправлять пожертвование, я все равно буду очень рад. рад, если ты прошел этот курс

День 0. Подготовка к работе

Для полноценного участия в этом курсе вам необходим доступ в Интернет и компьютер или смартфон с установленными Python и Git или последней версией популярного браузера для запуска [Gitpod](#) (веб-приложение, предоставляющее среду разработки в вашем браузере) или его аналоги (Repl.it, CodePen, CodeAnywhere и т. д.).

Я также рекомендую вам установить приложение Notion на свой компьютер или телефон чтобы вы могли отмечать там каждый из 21 дней для завершения «полного Курсовая задача». [Notion](#) бесплатен для личного использования.

Если по каким-то причинам вам не нравится приложение, вы можете просто прочитать одну главу PDF-книги в день, но я все равно советую вам присмотреться к Notion. Как установить Питон?

Я не буду вдаваться в подробности этого курса, поскольку все читатели работают на разных платформах. Я просто дам ссылки на инструкции для разных платформ.

Установочные файлы:

Windows:

<https://www.python.org/downloads/windows/>

MacOS :

<https://www.python.org/downloads/macos/>

Linux:

<https://www.python.org/downloads/source/>

Инструкция по установке (для разных платформ):

<https://wiki.python.org/moin/BeginnersGuide/Download>

Приложения для запуска скриптов Python с телефона:

Android-приложение Termux

<https://play.google.com/store/apps/details?id=com.termux&hl=en&pli=1>

(для установки используйте инструкции Linux)

Приложение Pythonista для iOS <https://apps.apple.com/us/app/pythonista-3/id1085978097?ls=1>

Как установить Гит?

Git — это система контроля версий. Это помогает вам отлаживать ошибки в вашем коде, позволяющий вернуться в состояние «когда все еще работало» и организовать работу в больших командах программистов (наглядно видеть «кто все сломал»).

В рамках этого курса вы будете использовать Git для копирования примеров кода из Github и для установки различных OSINT-инструментов.

Инструкция по установке для Windows, Linux и MacOS:

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Я сам буду использовать Gitpod во всех примерах. Это онлайн-разработка среда на базе Ubuntu (дистрибутив Linux).

Все, что вам нужно для его использования, — это браузер и учетная запись Github/Gitlab/Bitbucket для авторизация.

Если чтение инструкций по ссылкам выше помогло вам неудобно (например, командную строку не найти), я рекомендую вам также использовать Gitpod (до 50 часов в месяц для бесплатно).

Нужно ли проходить этот курс ровно 21 день?

Я бы настоятельно не советовал делать это быстрее. Если только вы не школьник (или девочка), у которого каникулы и у которого много свободного времени. В таком случае можно делать 2-3 урока в день (но не более).

Если вы работаете 8 и более часов в день, вы можете делать 1 урок каждые два-три дня. Вы также можете сделать перерыв на несколько дней во время курса, чтобы дать себе времени отдохнуть и поразмышлять над тем, что вы узнали.

Однако я бы не рекомендовал вам проходить этот курс более двух раз или три месяца.

Я бы также рекомендовал вам проходить уроки строго по порядку.

Но если вы решите иначе, в этом нет ничего плохого. Если только ты не можешь возникнуть ошибка «Нет имени модуля», если сценарий использует настроенный пакет на одном из предыдущих уроков.

```
PROBLEMS    OUTPUT    PORTS    DEBUG CONSOLE    TERMINAL
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_18
gitpod /workspace/python-for-OSINT-21-days/Day_18 (main) $ python map.py
Traceback (most recent call last):
  File "/workspace/python-for-OSINT-21-days/Day_18/map.py", line 1, in <module>
    import numpy as np
ModuleNotFoundError: No module named 'numpy'
gitpod /workspace/python-for-OSINT-21-days/Day_18 (main) $ █
```

В этой ситуации вам просто нужно установить нужный модуль (пакет) с помощью pip.

Например:

```
pip install numpy
```

```
gitpod /workspace/python-for-OSINT-21-days/Day_18 (main) $ pip install numpy
Collecting numpy
  Downloading numpy-1.24.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
    17.3/17.3 MB 21.1 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.24.3
```

Теперь приступим к обучению!

День 1. Запускаем первый скрипт

Давайте начнем с копирования репозитория Github с примерами файлов кода для этого конечно на свой компьютер.

Введите в командной строке:

```
git clone https://github.com/cipher387/python-for-OSINT-21-days  
cd python-for-OSINT-21-days
```

Если вы увидите сообщение с запросом имени пользователя и пароля, введите свой Имя пользователя и пароль учетной записи Github.

Репозиторий Github — это, по сути, хранилище файлов. с кодом, файлами Данных, и документация. От обычного каталога он отличается некоторыми Дополнительными функционал: история версий, возможность создавать проблемы (заметки с ошибками отчеты и вопросы), вилки (копии, дорабатываемые независимо от каждого другое) и некоторые другие Особенности.

Затем введите в командной строке:

```
python Day_1/start.py
```

Результат должен быть примерно таким:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files in a folder named 'PYTHON-FOR-OSINT-21-DAYS'. The 'Day_1' folder contains a file named 'start.py'. The main editor area displays the contents of 'start.py':

```
1 print ("Welcome to 21 day Python course!")
```

. Below the editor are tabs for 'PROBLEMS', 'OUTPUT', 'PORTS', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, showing the command `gitpod /workspace/python-for-OSINT-21-days (main) $ python Day_1/start.py` and its output: `Welcome to 21 day Python course!`

Попробуйте изменить текст в кавычках и запустите скрипт еще раз.

Если вы не используете Gitpod, теперь вы можете столкнуться с вопросом:
«Что такое лучшее приложение для редактирования файлов кода Python?»

Вы можете использовать любой текстовый редактор, который вам нравится.
Блокнот илиTextEdit тоже подойдут, но я по-прежнему рекомендую вам
попробовать и популярные редакторы кода: [Sublime Text](#), [Notepad++](#),
[Visual Studio Code](#) и т. д. Они могут автоматически выделять синтаксис.
и предложите имена функций/переменных (код автозаполнения).

Это все на сегодня.

Этот урок намного короче остальных, потому что я хочу оставить свободное
Время для тех, кто еще не успел установить Python, и для тех, кто уже
что-то идет не так, и вам нужно решить некоторые дополнительные проблемы.

Что делать, если что-то пошло не так?

Убедитесь, что вы правильно установили Python:

`python –version`

Убедитесь, что вы правильно установили Git:

`git –version`

Если папка «python-for-OSINT-21-days» не копируется, проверьте, ввели ли вы пароль правильный.

Если при запуске скрипта вы получаете сообщение об ошибке, попробуйте удалить папку и снова копирование. На всякий случай вы что-то изменили в коде, когда вы его просматривали.

А если ничего из этого не поможет, рекомендую подумать об использовании Gitpod.

Просто откройте его в браузере:

<https://gitpod.io#https://github.com/cipher387/python-for-OSINT-21-days>



И создайте New Workspace со стандартными настройками. При необходимости войдите в систему вашу учетную запись Github, Gitlab или Bitbucket.

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named 'Day_1' containing a file 'start.py'. The 'start.py' file is open in the editor, displaying the following code:

```
1 print ("Welcome to 21 day Python course!")
```

In the bottom right corner, the terminal window shows the command being run and its output:

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ python Day_1/start.py
  Welcome to 21 day Python course!
○ gitpod /workspace/python-for-OSINT-21-days (main) $
```

Когда все будет готово, введите в командной строке:

```
python Day_1/start.py
```

Пожалуйста, не переходите к следующему дню, пока не выполните Успешно завершил этот скрипт и появилось сообщение «Добро пожаловать в 21-дневный Python». Конечно!" появляется.

День 2. Минимальный базовый синтаксис

Сегодня мы познакомим вас с четырьмя основными понятиями синтаксиса Python которые также встречаются в большинстве популярных языков программирования.

Расскажу о них максимально кратко и просто.

Да, читатели, которые раньше изучали Python, могут подумать, что я что-то Упускаю очень важные вещи. Но еще раз повторюсь: этот курс не предназначен для того, чтобы вы хороший разработчик Python, он просто показывает вам возможные решения проблема автоматизации рутин в OSINT.

Переменная

Согласно классическому определению, это именованная область памяти, используется для доступа к определенным данным.

Переменные Python могут хранить:

текстовые значения (например, имя человека или глава в книге). Этот тип данных

объявляется с помощью `str()`;

Целые числа. Объявлен с помощью функции `int()`;

Плавающие числа. Объявлена с помощью функции `float()`;

Верно/неверно. Объявлен с помощью функции `bool()`.

Существует множество других типов данных, которые мы не будем рассматривать в этом курсе.

В некоторых языках необходимо объявить тип переменной. На Python без надобности этого делать не нужно (мы сделаем это всего пару раз во всем этом курсе). Например, если вы хотите добавить число к текстовая строка или каким-то образом объединить переменные, которые по умолчанию определены как данные разных типов.

В тексте можно использовать заглавные и строчные буквы, а также символ Подчеркивания. имена переменных. Вы также можете использовать цифры, но не в качестве первого символа.

Старайтесь давать переменным имена, которые имеют как можно больше смысла, чтобы они через некоторое время вам будет легче понять ваш код.

Давайте немного потренируемся. Запустите скрипт variable.py из папки Day_2:

```
cd Day_2  
python variable.py
```

```
variable.py M X  
Day_2 > variable.py  
1 first_name = "John"  
2 last_name = input('What is your last name?\n')  
3 print("You are" + " " + first_name + " " + last_name)  
4  
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL  
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_2  
● gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) $ python variable.py  
What is your last name?  
Smith  
You are John Smith  
○ gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) $
```

Обратите внимание, что скрипт Python мы запускаем немного иначе, чем в первом уроке. В прошлый раз мы сразу указывали путь к файлу, а теперь Делаем Сначала активируйте папку Day_2, а затем запустите скрипт. Оба Варианта приемлемо, делайте это так, как вам нравится.

В результате у вас должно получиться что-то похожее на картинку.

В дальнейшем я буду ссылаться на комментарии к коду со знаком #. Вы также можете добавить текст после знака # в коде сценария, и он будет проигнорирован интерпретатором.

В этом курсе я рекомендую сначала посмотреть на картинку кода на пару минут и попытайтесь угадать, что делает код. И только после этого Прочтите код с комментариями.

Присвойте значение John переменной first_name:

```
first_name = "John"
```

Затем мы присваиваем Last_name значение, введенное пользователем с помощью input() функция. (\n после знака вопроса является разрывом строки, его можно удалить, если вы хотите):

```
last_name = input('What is your last name?\n')
```

И затем выводим значение обеих переменных с помощью функции print():

```
print("You are" + " " + first_name + " " + last_name)
```

Обратите внимание, что для текстовых строк мы используем как одинарные, так и двойные кавычки. Оба типа действительны в коде Python.

Здесь мы начинаем использовать функции. Функция — это объект, который принимает аргументы в качестве входных данных и возвращает определенное значение или выполняет определенное действие в response.Input() и print() — это встроенные функции Python, которые принимают текстовые строки в качестве аргументов.

В рамках этого курса вы научитесь создавать свои собственные функции.

Условное заявление

Это синтаксическая конструкция, позволяющая выполнять определенные действия, если определенное условие соблюдено. Давайте сразу рассмотрим пример.

Запустите условие.py:

The screenshot shows the VS Code interface. The Explorer sidebar on the left displays a file tree for a workspace named 'PYTHON-FOR-OSINT-21-DAYS'. The 'Day_2' folder is currently selected. The Editor pane on the right shows two files: 'variable.py' and 'condition.py'. The code in 'condition.py' is as follows:

```
age = input('How old are you?\n')
if int(age) > 27:
    print("You are so old")
elif int(age) < 27:
    print("You are so young")
```

The Terminal pane at the bottom shows the command-line interface with the following logs:

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_2
gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) $ python condition.py
How old are you?
26
You are so young
gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) $ python condition.py
How old are you?
29
You are so old
gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) $
```

Сначала мы используем функцию `input()`, чтобы узнать у пользователя, сколько ему лет.

```
age = input('How old are you?\n')
```

Если он введет значение больше 27, мы ответим, что он очень старый.

```
if int(age) > 27:
    print("You are so old")
```

Если ему меньше 27, он очень молод.

```
elif int(age) < 27:
    print("You are so young")
```

Список

Список — это упорядоченный набор элементов, каждый из которых имеет свой номер или индекс, что позволяет быстрый доступ к нему.

Запустите `list.py`:

```
Day_2 > list.py
1 girls = ["Anna", "Maria", "Eva"]
2 print(girls)
3
4 girls.append("Brenda")
5 print(girls)
6
7 print (girls[3])
8
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

- gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_2
- gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) \$ python list.py
- ['Anna', 'Maria', 'Eva']
- ['Anna', 'Maria', 'Eva', 'Brenda']
- Brenda
- gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) \$

Создайте список женских имен:

```
girls = ["Anna", "Maria", "Eva"]
```

Отобразите его с помощью функции print():

```
print(girls)
```

Добавьте к нему элемент с помощью встроенной функции Append()
(по умолчанию новый элемент будет добавить в конец списка):

```
girls.append("Brenda")
```

Распечатываем список:

```
print(girls)
```

Выведите элемент номер три (элементы в списке начинаются с нуля):

```
print (girls[3])
```

В этом курсе мы будем использовать списки много раз и изучим множество различных встроенных функций для работы с ними.

Если вы изучали другие языки программирования, вы, вероятно, знакомы с понятие массивов. В Python также есть эта концепция. Массивы Python отличаются от списки в частности тем, что в списках можно использовать данные разных типов (например, первый элемент списка может быть строкой, а второй — число), а массивы должны состоять только из чисел или только из строк.

Есть и другие отличия, которые делают списки более гибкими и удобными. инструмент. Для большинства задач, связанных с OSINT, достаточно уметь пользоваться списками и мы не будем изучать массивы в этом курсе.

Списки также могут быть многомерными. Когда каждый элемент списка также является списком из 2, 3 или более предметов. Они будут упомянуты в курсе, но не получат много внимания.

Петля

Цикл — это синтаксическая конструкция, позволяющая повторять определенный код определенное количество раз или перебрать все элементы массива один за другим один.

Запустите цикл.py:

```
Day_2 > loop.py
1  girls = ["Anna", "Maria", "Eva"]
2
3  for girl in girls:
4      print(girl +"; ")
5
6
7  for x in range(20):
8      print(x)
9

gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_2
gitpod /workspace/python-for-OSINT-21-days/Day_2 (main) $ python loop.py
Anna;
Maria;
Eva;
0
1
2
3
4
5
6
7
8
9
10
11
12
```

Создайте список имен девочек:

```
girls = ["Anna", "Maria", "Eva"]
```

Выведите их один за другим, добавив к ним точку с запятой:

```
for girl in girls:  
    print(girl +"; ")
```

Выведите числа от 0 до 19 (помните, что счет в Python начинается с нуля):

```
for x in range(20):  
    print(x)
```

При использовании циклов и условий всегда обращайте внимание на Количество отступы. Перед «внутренним» кодом всегда должно быть четыре пробела.

На мой взгляд, это минимум теории, необходимый для того, чтобы начать писать на Python сценарии. Завтра мы начнем осваивать практические навыки, которые пригодятся OSINT.

День 3. Установка и запуск инструментов командной строки Python

Если вы часто читаете мой [Твиттер](#), возможно, вы регулярно видите сообщения о команде линейные инструменты для OSINT. Большинство из них написаны на Python. JavaScript (Node.js), Go, Bash (скрипт Shell) и Rust также популярны.

The screenshot shows a Twitter search interface. On the left, there's a sidebar with navigation links: Home, Explore, Notifications, Messages, Bookmarks, Twitter Blue, Verified Organizations, Profile, and More. A blue 'Tweet' button is at the bottom of this sidebar. The main area shows a search bar with the query '#python from:cjb_detective'. Below the search bar, there are tabs for Top, Latest, People, Photos, and Videos. A tweet from 'Cyber Detective' (@cjb_detective) is displayed. The tweet content is:
A tool to collect emails registered on a certain domain from search results (google, bing, yahoo, ask) and save the results to a text file. Proxy support.
github.com/4383/EmailHarv... (test in the pic)
github.com/maldevel/Email... (newest version)
#python #osint
The tweet has 1 reply, 82 retweets, 220 likes, and 22.5K views. At the bottom of the main area, there's a terminal window showing the EmailHarvester tool running in a Linux environment, displaying search results and command-line options.

Сегодня мы узнаем, как настроить их для запуска. В качестве примера я буду использовать Thorndyke и Blackbird — инструменты для поиска на страницах пользователя в социальных сетях по прозвищу.

Первый способ

Установка из менеджера пакетов Puri (Индекс пакетов).

Индекс пакетов Python (PyPI) — это репозиторий программного обеспечения для Python язык программирования (pypi.org). Он содержит более 300 000 пакетов! Мы будем использовать его почти на каждом уроке этого курса.

Начнем с Торндейка (<https://github.com/rly0nheart/thorndyke>), очень простой инструмент, позволяющий проверить, существует ли пользователь с определенным ником в различных социальных сетях. сети.

Просто введите в командной строке:

```
pip install thorndyke
```

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install thorndyke
Collecting thorndyke
  Downloading thorndyke-2022.1.2.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: requests in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from thorndyke) (2.28.2)
Collecting tqdm (from thorndyke)
  Downloading tqdm-4.65.0-py3-none-any.whl (77 kB)
    77.1/77.1 kB 3.9 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer<4,>=2 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests->thorndyke) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests->thorndyke) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests->thorndyke) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests->thorndyke) (2022.12.7)
Installing collected packages: tqdm, thorndyke
Successfully installed thorndyke-2022.1.2.1 tqdm-4.65.0
```

Вот и все! Теперь инструмент можно запустить.

Введите thorndyke + интересующий вас ник в командной строке:

```
thorndyke johmsmith
```

```

gitpod /workspace/python-for-OSINT-21-days (main) $ thorndyke johmsmith
THORNDYKE
[~] Enumerating @johmsmith on:
  0% | 0/233 [00:00<?, ?it/s]
[-] YouTube: Not Found
[-] Facebook: Not Found
  1% | 1/233 [00:00<00:28, 8.18it/s]
[-] Instagram: Not Found
  1% | 2/233 [00:00<01:24, 2.73it/s]
[-] TikTok: Not Found
[-] Likee: Not Found
  2% | 3/233 [00:00<01:06, 3.43it/s]
[-] Snapchat: Not Found
[+] Pinterest: https://www.pinterest.com/johmsmith/
  3% | 4/233 [00:01<01:02, 3.59it/s]
[-] Wikipedia: Not Found
  3% | 5/233 [00:01<01:06, 3.43it/s]
[-] Wattpad: Not Found
  4% | 6/233 [00:02<00:55, 4.02it/s]
[-] Google PlayStore: Not Found
  4% | 7/233 [00:02<00:51, 4.34it/s]

```

Второй способ

К сожалению, не все разработчики инструментов добавляют свои проекты в PyPi (несмотря на тем более, что это довольно легко сделать). Поэтому иногда приходится их копировать из Github, самостоятельно установите соответствующие модули и запустите инструмент, обратившись к непосредственно в файл кода вместо имени команды.

```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_3
● gitpod /workspace/python-for-OSINT-21-days/Day_3 (main) $ git clone https://github.com/p1ngul1n0/blackbird
Cloning into 'blackbird'...
remote: Enumerating objects: 1273, done.
remote: Counting objects: 100% (352/352), done.
remote: Compressing objects: 100% (134/134), done.
remote: Total 1273 (delta 293), reused 257 (delta 218), pack-reused 921
Receiving objects: 100% (1273/1273), 6.45 MiB | 1.76 MiB/s, done.
Resolving deltas: 100% (687/687), done.
● gitpod /workspace/python-for-OSINT-21-days/Day_3 (main) $ cd blackbird
● gitpod /workspace/python-for-OSINT-21-days/Day_3/blackbird (main) $ pip install -r requirements.txt
Collecting aiohttp==3.8.3 (from -r requirements.txt (line 1))
  Downloading aiohttp-3.8.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.0 MB)
    1.0/1.0 MB 9.2 MB/s eta 0:00:00
Collecting beautifulsoup4==4.11.1 (from -r requirements.txt (line 2))
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
    128.2/128.2 kB 12.2 MB/s eta 0:00:00
Collecting colorama==0.4.4 (from -r requirements.txt (line 3))
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting Flask==2.1.1 (from -r requirements.txt (line 4))
  Downloading Flask-2.1.1-py3-none-any.whl (95 kB)
    95.2/95.2 kB 15.2 MB/s eta 0:00:00
Collecting Flask_Cors==3.0.10 (from -r requirements.txt (line 5))
  Downloading Flask_Cors-3.0.10-py3-none-any.whl (14 kB)

```

Теперь установим еще один инструмент для поиска страниц социальных сетей по ник: черный дрозд (<https://github.com/p1ngul1n0/blackbird>). Введите командная строка:

```
cd Day_3  
git clone https://github.com/p1ngul1n0/blackbird  
cd blackbird  
pip install -r requirements.txt
```

Файл requirements.txt содержит список пакетов, необходимых для запуска инструмента.

Еще раз напомню, что команда «cd» используется для перехода к другая папка.

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_3  
● gitpod /workspace/python-for-OSINT-21-days/Day_3 (main) $ cd blackbird  
○ gitpod /workspace/python-for-OSINT-21-days/Day_3/blackbird (main) $ python blackbird.py -u ivanov
```



Made with ❤ by p1ngul1n0

```
[!] Searching 'ivanov' across 582 social networks  
[+] - #4 Telegram account found - https://t.me/ivanov [200 OK]  
|--Name: Ivanov  
[+] - #11 Soundcloud account found - https://soundcloud.com/ivanov [200 OK]  
|--Name: Ivan Ivanov  
|--location: Yekaterinburg, Russian Federation  
|--picture:  
[+] - #38 Disqus account found - https://disqus.com/by/ivanov/ [200 OK]  
[+] - #10 Reddit account found - https://www.reddit.com/user/ivanov/about.json [200 OK]  
|--Name:  
|--Bio:  
|--picture:  
[+] - #23 Pastebin account found - https://pastebin.com/u/ivanov [200 OK]  
|--Member since: Thursday 4th of October 2012 07:53:53 AM CDT  
|--picture: https://pastebin.com/cache/img/4/7/20/340792.jpg  
[+] - #13 Steam account found - https://steamcommunity.com/id/ivanov/ [200 OK]  
|--Name:
```

Чтобы проверить, прошла ли установка успешно, попробуйте запустить инструмент:

```
python blackbird.py -u ivanov
```

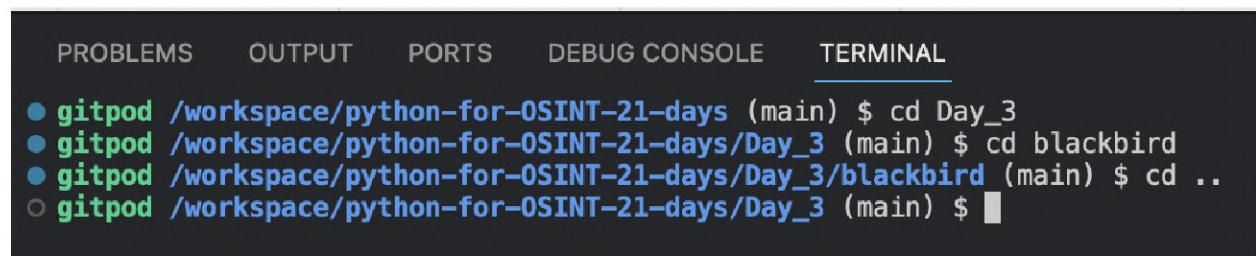
Третий способ

Вы также можете запускать инструменты непосредственно из кода сценария Python, используя Subprocess модуль (<https://docs.python.org/3/library/subprocess.html>), который позволяет вы можете запускать различные команды командной строки непосредственно в коде Python

Переместите файл launch.py из папки Day_3 в папку blackbird.

```
mv launch.py blackbird
```

Прежде чем запускать команду, убедитесь, что вы находитесь в папке Day_3.

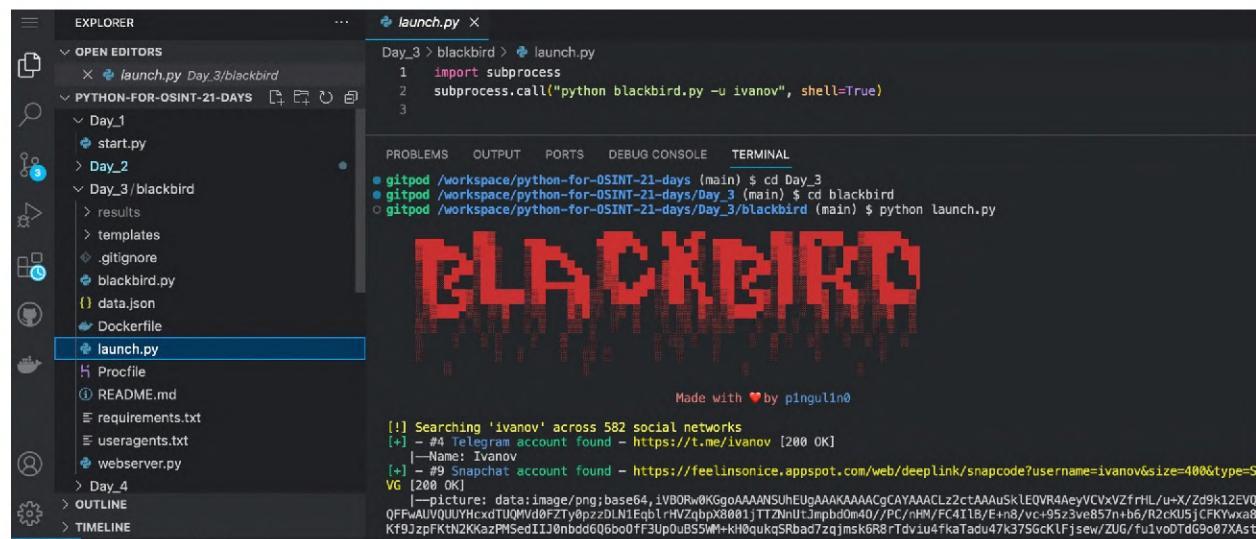


```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_3
● gitpod /workspace/python-for-OSINT-21-days/Day_3 (main) $ cd blackbird
● gitpod /workspace/python-for-OSINT-21-days/Day_3/blackbird (main) $ cd ..
○ gitpod /workspace/python-for-OSINT-21-days/Day_3 (main) $ █
```

Если вы находитесь в папке blackbird, используйте эту команду, чтобы перейти в один каталог выше:

```
cd ..
```

Запустите launch.py:



```
EXPLORER          launch.py ×
OPEN EDITORS
Day_3 > blackbird > launch.py
1 import subprocess
2 subprocess.call("python blackbird.py -u ivanov", shell=True)

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_3
● gitpod /workspace/python-for-OSINT-21-days/Day_3 (main) $ cd blackbird
● gitpod /workspace/python-for-OSINT-21-days/Day_3/blackbird (main) $ python launch.py

[!] Searching 'ivanov' across 582 social networks
[*] - # Telegram account found - https://t.me/ivanov [200 OK]
|--Name: Ivanov
[*] - # Snapchat account found - https://feelinsonice.appspot.com/web/deeplink/snapcode?username=ivanov&size=400&type=S
VC [200 OK]
|--picture: data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAKAAAAACgCAYAAACLzctAAAUSeL0VRA4eyVCVxVZfrHL+uX/zd9k12EV0QFFAUUVQUUYHcxTUQMV0fZTyppzDLN1Eqb1lHZqbx8801jTTZnlfIjepbd0m40//PC/nHM/F41IB/E+n8/vc-j95z3wv857n+h6/R2cKU5jCFKVwxw8Kf9JzpFKtN2KKazPMsedIIJ0nbdd6Q6bo0ff3Up0uBS5W+KH0quqSrbad7zqjmsk6R8rTduiu4fkaTadu47k37SGckFjsew/ZUG/fu1vodTdg9o07Xast
```

Импортируем модуль подпроцесса:

```
import subprocess
```

И запустите blackbird.py:

```
subprocess.call("python blackbird.py -u ivanov", shell=True)
```

Таким образом, вы можете запускать не только скрипты Python, но и скрипты, созданные на других языках языки программирования.

Самое главное, что вы должны запомнить из этого курса, это то, что Блэкберд и Торндайк — НЕ лучшие решения для никса перечисление.

Шерлок (<https://github.com/sherlock-project/sherlock>) и
Мегрэ (<https://github.com/soxoj/maigret>) проверьте гораздо больше сайтов.
Попробуйте установить и управляйте ими самостоятельно.

День 4. Чтение и запись файлов

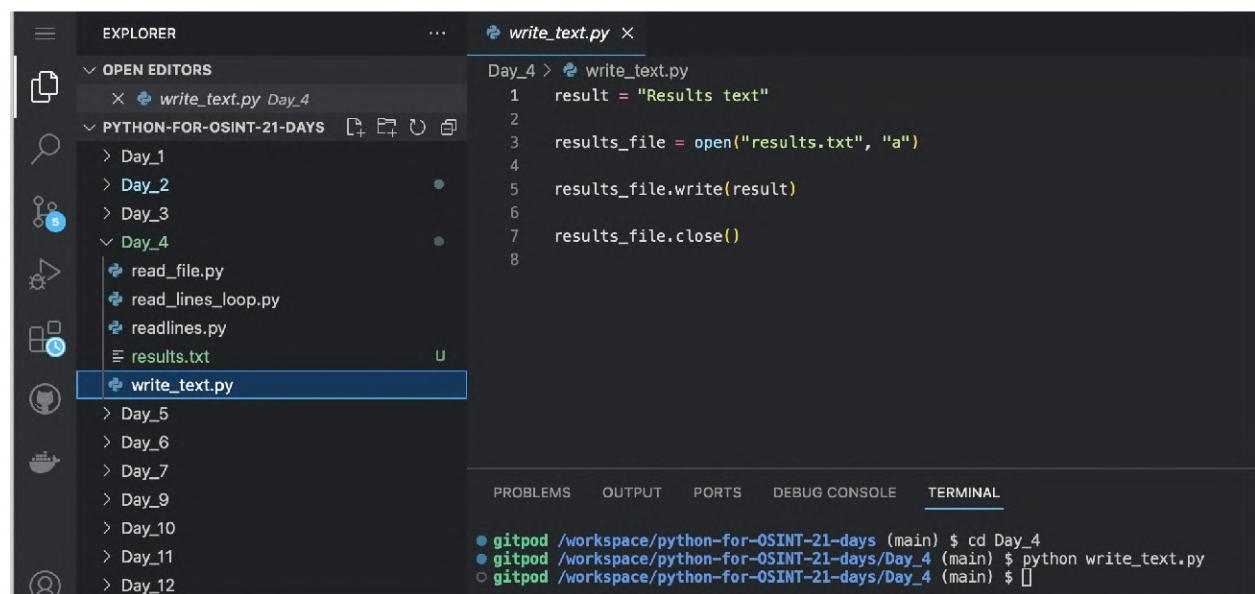
В комментариях к твитам в Twitter [@cyb_detective](#) о команде линейных инструментов часто можно увидеть комментарии «Зачем это нужно, когда есть веб-приложение с похожими функциями?».

Одним из преимуществ использования инструментов командной строки в расследованиях является возможность сохранять результаты в файлы, чтобы их можно было автоматически проанализировать позже.

В этом уроке мы научимся читать и записывать данные из текстовых файлов с помощью стандартные функции Python. Начнем с написания.

Запись файла

Запустите `write_text.py`:



```
Day_4 > write_text.py
1 result = "Results text"
2
3 results_file = open("results.txt", "a")
4
5 results_file.write(result)
6
7 results_file.close()
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_4
● gitpod /workspace/python-for-OSINT-21-days/Day_4 (main) $ python write_text.py
○ gitpod /workspace/python-for-OSINT-21-days/Day_4 (main) $ [ ]
```

Создайте переменную с определенным текстом:

```
result = "Results text"
```

Откройте (и одновременно создайте) файл results.txt:

```
results_file = open("results.txt", "a")
```

Записываем в нее значение переменной:

```
results_file.write(result)
```

Закрываем файл:

```
results_file.close()
```

Обратите внимание, что функция `open()` имеет два аргумента.

Первое – это имя файла, а второй — так называемый «режим открытия».

Примеры режимов открытия файлов:

«r» — открыть файл для чтения.

«a» — открыть файл для добавления (создать файл, если он не существует)

«w» — открыть файл для записи (создать файл, если он не существует)

«x» — создать новый файл.

Помните, что в некоторых ситуациях нет необходимости писать специальный код записать результаты работы скрипта в файл, потому что проще всего записать результаты скрипта Python в файл — это просто добавить > и имя файла в команда для его запуска:

The screenshot shows the VS Code interface with the 'ivanov_pages.txt' file open in the editor. The file contains a list of search results for the name 'ivanov'. The results are as follows:

```
1
2 97m
3 | | | | | | | | | | | | | | | | | | | | | |
4 31;1m
5 0m
6 97m [92m+ [97m] Enumerating @ [92mivanov [97m on: [0m
7 97m [31;1m [97m] YouTube: [31;1mNot Found [0m
8 97m [92m+ [97m] Facebook: [92mhttps://www.facebook.com/ivanov [0m
9 97m [31;1m [97m] Instagram: [31;1mNot Found [0m
10 97m [31;1m [97m] TikTok: [31;1mNot Found [0m
11 97m [31;1m [97m] Likee: [31;1mNot Found [0m
12 97m [31;1m [97m] Snapchat: [31;1mNot Found [0m
13 97m [92m+ [97m] Pinterest: [92mhttps://www.pinterest.com/ivanov [0m
14 97m [31;1m [97m] Wikipedia: [31;1mNot Found [0m
15 97m [31;1m [97m] Wattpad: [31;1mNot Found [0m
16 97m [31;1m [97m] Google PlayStore: [31;1mNot Found [0m
```

The terminal at the bottom shows the command used to generate the file: `thorndyke ivanov >ivanov_pages.txt`.

Теперь попробуем прочитать текст только что созданного нами файла.

Чтение файла

Запустите `read_file.py`:

The screenshot shows the VS Code interface with the 'read_file.py' file open in the editor. The code reads the contents of the 'results.txt' file:

```
Day_4 > read_file.py
1     results_file = open("results.txt", "r")
2
3     print(results_file.read())
4
```

The terminal at the bottom shows the command used to run the script: `python read_file.py`. The output is:

```
Results text
```

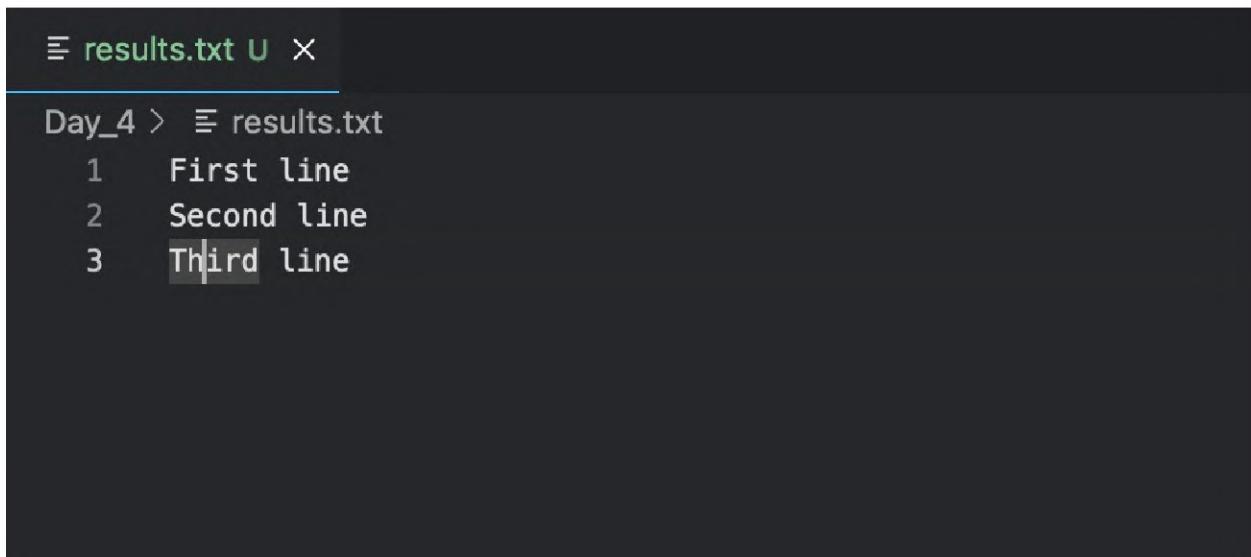
Откройте файл `results.txt`:

```
results_file = open("results.txt", "r")
```

Отображение содержимого файла results.txt:

```
print(results_file.read())
```

Есть еще один путь. С помощью простого цикла вы можете прочитать Строки файл по одному и выполнять некоторые действия над каждой строкой.



The screenshot shows a terminal window with the title bar 'results.txt'. The window displays the following text:

```
Day_4 > results.txt
1 First line
2 Second line
3 Third line
```

Пожалуйста, добавьте несколько строк в файл results.txt и запустите read_lines_loop.py:

```
results.txt read_lines_loop.py
Day_4 > read_lines_loop.py
1
2 stringNumber = 1
3
4 with open("results.txt") as f:
5     for line in f:
6         print(str(stringNumber) + ". " + line)
7         stringNumber += 1
8
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_4
gitpod /workspace/python-for-OSINT-21-days/Day_4 (main) $ python read_lines_loop.py
1. First line
2. Second line
3. Third line
gitpod /workspace/python-for-OSINT-21-days/Day_4 (main) $
```

Создайте переменную с номером строки:

```
stringNumber = 1
```

Откройте файл results.txt:

```
with open("results.txt") as f:
```

Пройдитесь по строкам и напечатайте каждую строку с номером строки:

```
for line in f:
    print(str(stringNumber) + ". " + line)
```

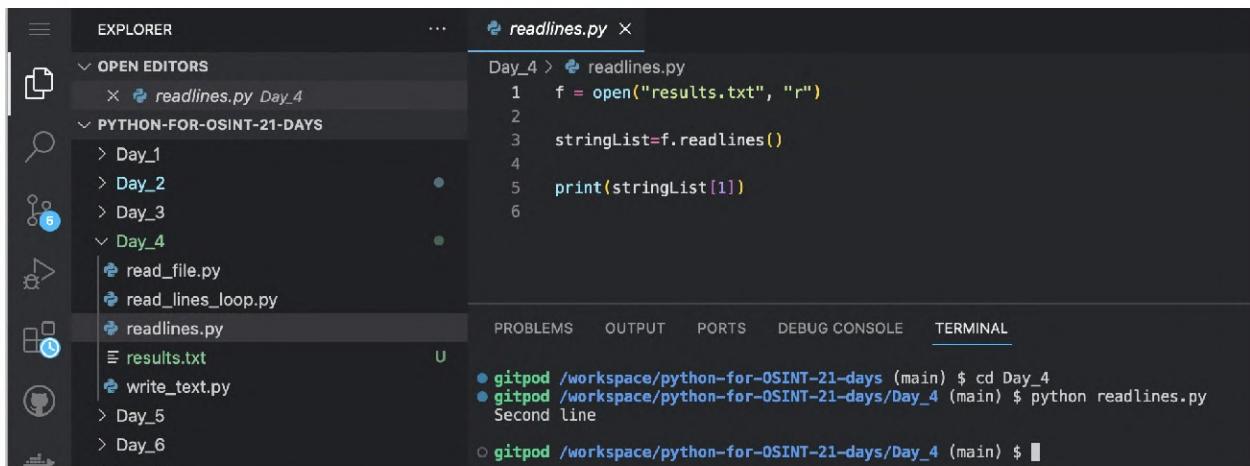
Увеличиваем номер строки на единицу:

```
    stringNumber += 1
```

Обратите внимание, что мы используем `str()` для преобразования переменной целочисленного типа в строку. Ты всегда следует делать это, когда вы объединяете текстовую переменную и число в одну строку.

Если вы не хотите печатать все строки файла, а только строки с определенными числами, вы можете использовать функцию `readlines()`, которая преобразует строки файла в элементы списка.

Запустите `readlines.py`:



```
Day_4 > ❶ readlines.py
1   f = open("results.txt", "r")
2
3   stringList=f.readlines()
4
5   print(stringList[1])
6
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_4
gitpod /workspace/python-for-OSINT-21-days/Day_4 (main) $ python readlines.py
Second line
gitpod /workspace/python-for-OSINT-21-days/Day_4 (main) $
```

Откройте файл `results.txt`:

```
f = open("results.txt", "r")
```

Создайте массив, элементами которого являются строки файла `results.txt`:

```
stringList=f.readlines()
```

Выведите элемент массива с индексом один (вторая строка файла). Не забываем, что в списках отсчет идет с нуля:

```
print(stringList[1])
```

Если, наоборот, нужно записать элементы массива в файл так, чтобы каждый элемент записывается в отдельной строке, используйте метод [writelines\(\)](#).

Хранение данных в файлах не всегда является лучшим решением (хотя его легче всего освоить). Если вам регулярно приходится работать с файлами размером в десятки или сотни мегабайты, вам следует подумать о том, чтобы начать использовать базы данных. Хорошо немного коснитесь этой темы в Уроке 8.

День 5. Обработка HTTP-запросов и работа с ними API

Когда вы открываете веб-страницу в браузере, происходит запрос к серверу. В ответ на запрос сервер возвращает статус, заголовки и тело ответа (например, html-код веб-страницы, некоторые данные в формате CSV, JSON или XML).

The screenshot shows the REST Test tool interface. On the left, there's a form titled "HTTP request options" with fields for "Method" (set to "GET") and "Endpoint" (set to "https://api.github.com/search/users?q=javascript"). Below these are buttons for "Add authentication", "Add header", "Add parameter", and "Add file". A large green button at the bottom right says "Ajax request". To the right of the form is a green box labeled "HTTP 200 success" containing a JSON response from GitHub's search API. The response includes fields like "total_count", "incomplete_results", and a list of user items. At the bottom right of the main area is a smaller box showing the raw HTTP response headers.

```
[{"total_count": 128698, "incomplete_results": false, "items": [{"login": "javascript", "id": 48139711, "node_id": "MDEyOk9yZ2FuaXphdGlvbjQ4MTMSNzEx", "avatar_url": "https://avatars.githubusercontent.com/u/48139711?v=4", "gravatar_id": "", "url": "https://api.github.com/users/javascript", "html_url": "https://github.com/javascript", "followers_url": "https://api.github.com/users/javascript/followers", "following_url": "https://api.github.com/users/javascript/following{/other_user}", "gists_url": "https://api.github.com/users/javascript/gists{/gist_id}"}, {"starred_url": "https://api.github.com/users/javascript/starred{/owner}"}]}
```

```
cache-control: no-cache
content-type: application/json; charset=utf-8
link: <https://api.github.com/search/users?q=javascript&page=2>; rel="next", <https://api.github.com/search/users?q=javascript&page=3>; rel="last"
x-github-media-type: github.v3; format=json
x-github-request-id: 5C4B:BF6F:7C0EBC6:7D39AAD:6448DF00
x-ratelimit-limit: 10
x-ratelimit-remaining: 9
x-ratelimit-reset: 1682693948
x-ratelimit-resource: search
x-ratelimit-used: 1
```

Самый простой способ понять, что происходит, — визуально. Открыть <https://resttesttest.com>, скопируйте туда какую-нибудь ссылку и нажмите «AJAX-запрос» кнопка.

OSINT часто необходимо автоматизировать сбор данных с веб-страниц или различных API (интерфейс прикладного программирования). И базовые навыки, необходимые для этого это написание кода для отправки запросов на веб-серверы и обработки ответы.

API (интерфейс прикладного программирования) — это технология, позволяющая взаимодействовать с приложением, отправляя запросы на сервер. Например, API Github позволяет получать данные о пользователях Github, а также вносить изменения в репозитории и многое другое.

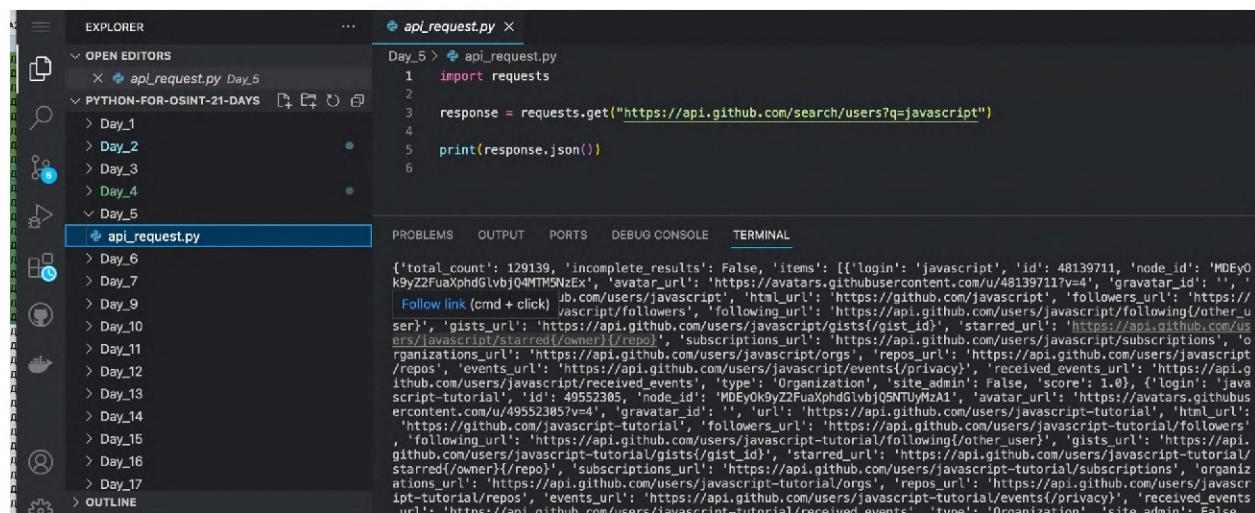
Для этого мы будем использовать запросы пакетов (<https://pypi.org/project/requests/>).

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install requests
Requirement already satisfied: requests in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from requests) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests) (2022.12.7)
```

Пакет запроса на установку:

`pip install requests`

Запустите `api_request.py`:



```
Day_5 > api_request.py
1 import requests
2
3 response = requests.get("https://api.github.com/search/users?q=javascript")
4
5 print(response.json())
6
```

```
{'total_count': 129139, 'incomplete_results': False, 'items': [{}], 'node_id': 'MDExOk9yZ2FuXphdGvbj04MTM5nZEx', 'avatar_url': 'https://avatars.githubusercontent.com/u/48139711?v=4', 'gravatar_id': '', 'Follow link (cmd + click)': 'jb.com/users/javascript', 'html_url': 'https://github.com/javascript', 'followers_url': 'https://api.github.com/users/javascript/following/other_user', 'gists_url': 'https://api.github.com/users/javascript/gists{/gist_id}', 'starred_url': 'https://api.github.com/users/javascript/subscriptions', 'organizations_url': 'https://api.github.com/users/javascript/orgs', 'repos_url': 'https://api.github.com/users/javascript/repos', 'events_url': 'https://api.github.com/users/javascript/events{/privacy}', 'received_events_url': 'https://api.github.com/users/javascript/received_events', 'type': 'Organization', 'site_admin': False, 'score': 1.0}, {'login': 'javascript', 'id': 48139711, 'node_id': 'MDExOk9yZ2FuXphdGvbj04NTUzMzA1', 'avatar_url': 'https://avatars.githubusercontent.com/u/49552305?v=4', 'gravatar_id': '', 'url': 'https://api.github.com/users/javascript-tutorial', 'html_url': 'https://github.com/javascript-tutorial', 'followers_url': 'https://api.github.com/users/javascript-tutorial/followers', 'following_url': 'https://api.github.com/users/javascript-tutorial/following/other_user', 'gists_url': 'https://api.github.com/users/javascript-tutorial/gists{/gist_id}', 'starred_url': 'https://api.github.com/users/javascript-tutorial/subscriptions', 'organizations_url': 'https://api.github.com/users/javascript-tutorial/orgs', 'repos_url': 'https://api.github.com/users/javascript-tutorial/repos', 'events_url': 'https://api.github.com/users/javascript-tutorial/events{/privacy}', 'received_events_url': 'https://api.github.com/users/javascript-tutorial/received_events', 'type': 'Organization', 'site_admin': False}
```

Добавьте пакет запросов в файл сценария с помощью команды импорта:

```
import requests

# Сделать запрос:

response =
requests.get("https://api.github.com/search/users?q=javascript")

# Отображение результата в формате JSON:

print(response.json())
```

Существует огромное количество API, как платных, так и бесплатных, которые предоставляют полезные данные для OSINT. Например:

информация о номерах телефонов, адресах и почтовых индексах;
информация о компаниях;
Информация о доменах и IP-адресах;
Информация о криптокошельках и транзакциях;
Информация о пользователях различных социальных сетей.

Список из более чем сотни полезных OSINT API можно найти на этом Github репозиторий:

<https://github.com/cipher387/API-s-for-OSINT>

Table of contents

- APIs
 - Phone Number Lookup and Verification
 - Address/ZIP codes lookup
 - People and documents verification
 - Business/Entity search
 - Domain/DNS/IP lookup
 - Mobile Apps Endpoints
 - Scraping
 - Whois
 - GEO IP
 - Wi-Fi lookup
 - Network
 - Finance
 - Email
 - Names/Surnames
 - PastebinLeaks
 - Archives
 - Hashes decrypt/encrypt
 - Crypto
 - IoT
 - Malware
 - Face Search and detection
 - Social Media and Messengers
 - UNOFFICIAL APIs
 - Search Engines
 - News analyze
 - Darknet
 - Torrents/file sharing
 - Flights
 - Webcams
 - Regular expressions
 - API testing tools
 - Curl converters
 - Create your own API
 - Distribute your own API
 - API Keys Info
- Contributing
- Contact
- License

GEO IP				Name/Surname			
Name	Link	Description	Price	Name	Link	Description	Price
ipstack	https://ipstack.com	Get country, region, city and zip code provides country, city, state, province, local, country code and longitude, latitude, location details	FREE	Geonameize.io	https://geonameize.io	Identify where the location of how many a certain name is to be found or how many names and shows the popularity of the name	10.00 names/day free
geolocation.io	https://geolocation.io	IP address, longitude, latitude, country, city, state, province, local, location details, location, time zone, ip address, subnet and more	20.00 requests per month, 100.00 FREE	Agefy.io	https://agefy.io	Predicts the age of a person given their name	10.00 names/day free
ipinfodb	https://ipinfodb.com/api/	Free Geolocation tools and APIs for country region, city and IP address lookup by IP address	FREE	Nationalize.io	https://nationalize.io	Predicts the nationality of a person given their name	10.00 names/day free
ip-api	https://ip-api.com/	Free domain/IP geo location info	FREE				

Wi-Fi lookup				Pestebin/Leaks			
Name	Link	Description	Price	Name	Link	Description	Price
MyNetwork API	https://www.my-network.org/	public API implementation of WiFi-Ear-EasyScanner database	FREE	FindRephorward	https://www.findrephorward.com/API/v5	Shows the list of new forwarded emails (email addresses and usernames)	\$2.00 per month
Wifiguru	https://wifiguru.net/	get location and other information by SSID	FREE	Peashape.ws	https://peashape.ws/api	Search in pestebin	\$0.00 per 10000 requests
PeeringDB	https://www.peeringdb.com/peeringdb/	Database of networks, and the go-to location for interconnection points	FREE	LeakPeek	https://leak-peek.ws/api	Search in leaks databases	\$8.99 per week unlimited addresses
PacketTotal	https://packettotal.com/api.html	Protocol analyzer	FREE	BreachDirectory.com	https://breachdirectory.com/api/	Search domain in data breaches databases	FREE

Network				Finance			
Name	Link	Description	Price	Name	Link	Description	Price
BitIntel.net	https://bit-intel.net/	get information about a link by URL	FREE	BitIntel.net	https://bit-intel.net/	get information about a link by URL	FREE
FDIC Bank Data API	https://fdicdata.firebaseio.godaddy.com/	Bank names, locations and history events	FREE	Amocoiner	https://www.amocoiner.com/currency-api/	Free currency API ...	FREE

Нет необходимости писать отдельный скрипт Python для тестирования разных API. Это лучше воспользоваться специальными онлайн-сервисами, которые позволяют моделировать разные виды запросы и методы авторизации.

Online REST & SOAP API Testing Tool

ReqBin is an online API testing tool for REST and SOAP APIs. Test API endpoints by making API requests directly from your browser. Test API responses with built-in JSON and XML validators. Load test your API with hundreds of simulated concurrent connections. Generate code snippets for API automation testing frameworks. Share and discuss your API requests online.

Status: 200 (OK) Time: 129 ms Size: 34.50 kb

The authorization header will be automatically generated when you send the request. Read more about HTTP Authentication.

```

"total_count": 128399,
"incomplete_results": false,
"items": [
  {
    "login": "javascript",
    "id": 48139711,
    "node_id": "MDExOk9yZ2FuaXphdGlvbjQ4MTM5NzEx",
    "avatar_url": "https://avatars.githubusercontent.com/u/48139711?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/javascript",
    "html_url": "https://github.com/javascript",
    "followers_url": "https://api.github.com/users/javascript/followers",
    "following_url": "https://api.github.com/users/javascript/following{/other_user}",
    "created_at": "2008-07-11T19:27:09Z",
    "updated_at": "2023-03-22T17:43:43Z"
  }
]
  
```

Например, reqbin.com или [REST API Tester](#).

Мы вернемся к теме сетевых запросов, когда будем говорить о файлах JSON, парсинге и использовании прокси-серверов. Мы научимся добавлять в запрос заголовки и извлекать данные из текстов ответов.

День 6. JSON

На прошлом уроке мы говорили о том, что много полезных данных для Исследования можно получить через различные API. Многие из них возвращают данные в формате JSON (JavaScript Object Notation) (а также CSV и XML, но об этих форматах мы поговорим в следующих уроках).

На прошлом уроке вы видели очень хороший пример данных JSON при работе с API Github ([документация](#)).

The screenshot shows a JSON viewer interface with two main panes. The left pane displays the raw JSON code, and the right pane shows the hierarchical structure of the JSON object.

Raw JSON (Left Pane):

```
1- [
2   "total_count": 128992,
3   "incomplete_results": false,
4   "items": [
5     {
6       "login": "javascript",
7       "id": 48139711,
8       "node_id": "MDEyOk9yZ2FuaXphdGlvbjQ4MTM5NzEx",
9       "avatar_url": "https://avatars.githubusercontent.com/u/48139711?v=4",
10      "gravatar_id": "",
11      "url": "https://api.github.com/users/javascript",
12      "html_url": "https://github.com/javascript",
13      "followers_url": "https://api.github.com/users/javascript/followers",
14      "following_url": "https://api.github.com/users/javascript/following/{other_user}",
15      "gists_url": "https://api.github.com/users/javascript/gists{/gist_id}",
16      "starred_url": "https://api.github.com/users/javascript/starred{/owner}{/repo}",
17      "subscriptions_url": "https://api.github.com/users/javascript/subscriptions",
18      "organizations_url": "https://api.github.com/users/javascript/orgs",
19      "repos_url": "https://api.github.com/users/javascript/repos"
]
```

JSON Viewer (Right Pane):

```
object > items >
object {3}
  total_count : 128992
  incomplete_results : false
  items [38]
    object {38}
      login : javascript
      id : 48139711
      node_id : MDEyOk9yZ2FuaXphdGlvbjQ4MTM5NzEx
      avatar_url : https://avatars.githubusercontent.com/u/48139711?v=4
      gravatar_id : null
      url : https://api.github.com/users/javascript
      html_url : https://github.com/javascript
      followers_url : https://api.github.com/users/javascript/followers
      following_url : https://api.github.com/users/javascript/following
      gists_url : https://api.github.com/users/javascript/gists
      starred_url : https://api.github.com/users/javascript/starred
      subscriptions_url : https://api.github.com/users/javascript/subscriptions
      organizations_url : https://api.github.com/users/javascript/orgs
      repos_url : https://api.github.com/users/javascript/repos
      events_url : https://api.github.com/users/javascript/events
      received_events_url : https://api.github.com/users/javascript/received_events
    
```

В ответ на запрос мы получаем список из 30 объектов (items[0], items[1], items[2] и т.д.), каждый из которых соответствует конкретному пользователю.

У каждого объекта есть свойства, в которых хранится информация о пользователе: логин, html_url, id, Follow_url и т.д.

Теперь попробуем извлечь данные из файлов JSON с помощью кода. Пакет JSON (<https://docs.python.org/3/library/json.html>) доступен в Python по умолчанию и не требует установки.

Чтение одного поля

Запустите `read_one_field.py`:

```
Day_6 > read_one_field.py
1 import json
2 import requests
3
4 response = requests.get("https://api.github.com/search/users?q=javascript")
5
6 json_data=response.json()
7
8 print (json_data['total_count'])
9
10 print (json_data['items'][0]['html_url'])
11
```

gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_6
gitpod /workspace/python-for-OSINT-21-days/Day_6 (main) \$ python read_one_field.py
129177
https://github.com/javascript
gitpod /workspace/python-for-OSINT-21-days/Day_6 (main) \$

Импортируем модули json и Requests:

```
import json
import requests
```

Затем сделайте тот же запрос к API Github, что и в предыдущем уроке

```
response =
requests.get("https://api.github.com/search/users?q=javascript")
```

Присвоим переменной значение ответа на запрос в формате json:

```
Json_data = response.json()
```

Вывести общее количество результатов:

```
print(json_data['total_count'])
```

Выводим ссылку на первый профиль Github из результатов:

```
print(json_data['items'][0]['html_url'])
```

Но чаще всего нам нужно извлечь не одно значение, а информацию о целом списке объектов. Например, ссылки на профили пользователей Github из примера выше.

Чтение списка полей

Запустите `read_list_of_fields.py`:

The screenshot shows the Visual Studio Code interface. The left sidebar (EXPLORER) displays a file tree with a folder named 'PYTHON-FOR-OSINT-21-DAYS' containing subfolders 'Day_1' through 'Day_16'. Inside 'Day_6', there are files named 'read_list_of_fields.py' and 'read_one_field.py'. The right pane (EDITOR) shows the code for 'read_list_of_fields.py'. The code uses the requests library to make a GET request to the GitHub API endpoint for searching users with the query 'javascript'. It then prints the URLs of the first 15 results. The bottom right pane (TERMINAL) shows the command being run: `gitpod /workspace/python-for-OSINT-21-days/Day_6 (main) $ python read_list_of_fields.py`. The output in the terminal lists 15 GitHub URLs related to 'javascript'.

```
import json
import requests

response = requests.get("https://api.github.com/search/users?q=javascript")
json_data=response.json()
usersCount = len(json_data['items'])-1
for x in range(usersCount):
    print (json_data['items'][x]['html_url'])
```

gitpod /workspace/python-for-OSINT-21-days/Day_6 (main) \$ python read_list_of_fields.py
https://github.com/javascript
https://github.com/javascript-tutorial
https://github.com/javascripter
https://github.com/javascriptit
https://github.com/hyperlink
https://github.com/JavaScriptErika
https://github.com/JavascriptDeNoobAPro
https://github.com/javascriptDev
https://github.com/javascriptdezero
https://github.com/javascript-forks
https://github.com/javascript666666
https://github.com/JavascriptMick

```
# Импортируем пакеты json и запросы:
```

```
import json  
import requests
```

Делаем запрос к API:

```
response =  
requests.get("https://api.github.com/search/users?q=javascript")
```

Получаем результат в формате JSON:

```
json_data=response.json()
```

Подсчитаем количество результатов:

```
usersCount = len(json_data['items'])-1
```

Распечатываем ссылку на каждый результат в цикле:

```
for x in range(usersCount):  
    print (json_data['items'][x]['html_url'])
```

JSONPath Online Evaluator - jsonpath.com

JSONPath
\$.items[0]["html_url"]

Output paths [Expand JSONPath expressions](#)

Inputs

```
1 - [  
2   "total_count": 125524,  
3   "incomplete_results": false,  
4   "items": [  
5     {  
6       "login": "javascript",  
7       "id": 48139711,  
8       "node_id": "MDIyOk9yZ2FuaXphdGlvbjQ4MTM5NzEx",  
9       "avatar_url": "https://avatars.githubusercontent.com/u/48139711?v=4",  
10      "gravatar_id": "",  
11      "url": "https://api.github.com/users/javascript",  
12      "html_url": "https://github.com/javascript",  
13      "followers_url": "https://api.github.com/users/javascript/followers",  
14      "following_url": "https://api.github.com/users/javascript/following{/other_us  
15      "gists_url": "https://api.github.com/users/javascript/gists{/gist_id}"}
```

Evaluation Results

```
1 - [  
2   "https://github.com/javascript"  
3 ]
```

Часто бывает, что структура файлов JSON достаточно сложна и сложно понять, как отметить путь к тем или иным данным. Особенный

Сервисы помогут вам в этом разобраться. Например, <https://jsonpath.com/> или <https://jsonpathfinder.com>

И прежде чем писать какой-либо код для обработки файлов JSON, помните, что иногда проще преобразовать их в файлы CSV и просто Вырезать столбцы с нужными вам данными:

[JSON to CSV online converter](#)

День 7. CSV

CSV (Значения, разделенные запятыми) — один из самых популярных форматов хранения табличных данных.

Вот пример того, как выглядит файл CSV при открытии в текстовом редакторе.

```
1 |created_on,updated_from_linkedin,full_name,first_name,last_name,headline,current_position,company_name ,person_city,person_state,person_country,person_industry,tags,person_skills,education_experience,compa ny_website,email1,email1_type,email1_status,email2,email2_type,email2_status,email3,email3_type,email3 _status,email4,email4_type,email4_status,phone1,phone1_type,phone1_status,phone2,phone2_type,phone2_st atus,phone3,phone3_type,phone3_status,phone4,phone4_type,phone4_status,company_size_from,company_size_ to,current_position_2,current_company_2,previous_position_2,previous_company_2,previous_position_3,pre vious_company_3,company_city,company_state,company_country,person_linkedin_url,person_angellist_url,pe rson_crunchbase_url,person_twitter_url,person_faceb ook_url,company_linkedin_url,person_image_url,company_logo_url,created_by,lead_status  
2 |2022-12-19,2022-12-19,Caitlin Maier,Caitlin,Maier,Assistant Vice President Talent Acquisition at Golub Capital,Assistant Vice President Talent Acquisition,Golub Capital,Chicago,Illinois,United States,Financial Services,"Recruiting, Customer Service, Sales, Management, Microsoft Word, Cold Calling, Entrepreneurship, Leadership, Microsoft Office, Microsoft Excel, Human Resources, Sales Process, Interviewing, Facebook, Training, Account Management, Marketing, Social Media, PowerPoint, Social Media Marketing, Onboarding",James Madison University (2007 - 2011),https://golubcapital.com/,maicerce@gmail.com,personal,valid,cmaier@golubcapital.com,main job,valid,,,,,,,,,,501,1000,,,Recruiting Coordinator,Capital One,Recruiter,Capital One,San Francisco,California,United States,https://www.linkedin.com/in/caitlinemaier,,,https://twitter.com/CaityMai_COF,,https://www.linkedin.com/company/
```

А вот как это выглядит, когда вы открываете Numbers (эквивалент MS Excel для Mac).

created_on	updated_from_linkedin	full_name	first_name	last_name	headline
2022-12-19	2022-12-19	Caitlin Maier	Caitlin	Maier	Assistant Vice President Talent Acquisition at Golub Capital
2022-12-19	2022-12-19	Brian Boje	Brian	Boje	Vice President Of Recruiting at STS Technical Services
2022-12-19	2022-12-19	Jim Burns	Jim	Burns	Vice President - Talent Acquisition at Thrive Skilled Pediatric Care
2022-12-19	2022-12-19	Bryan D. Schreiber	Bryan D.	Schreiber	Vice President Talent Acquisition at Fiserv #fisvproud #fintech
2022-12-19	2022-12-19	Brooke Wheeler	Brooke	Wheeler	Vice President Talent Acquisition at MerchantE
2022-12-19	2022-12-19	Brooke Govert	Brooke	Govert	Vice President Talent Acquisition Director at Edelman
2022-12-19	2022-12-19	Brittni Williamson Herbert	Brittni	Williamson Herbert	Assistant Vice President System Talent Acquisition- Talent Attractic

Попробуем создать CSV-файл, используя пакет CSV.
(<https://docs.python.org/3/library/csv.html>).

Запись CSV-файла

Запустите write_csv.py:

The screenshot shows the VS Code interface. The Explorer sidebar on the left lists files and folders: 'OPEN EDITORS' contains 'write_csv.py' (Day_7), and 'PYTHON-FOR-OSINT-21-DAYS' contains 'Day_1', 'Day_2', 'Day_3', 'Day_4', 'Day_5', 'Day_6', 'Day_7' (which contains 'csv_to_json.py', 'read_csv_one_column.py', 'read_csv.py', 'test.csv', and 'write_csv.py'), 'Day_9', 'Day_10', and 'Day_11'. The 'write_csv.py' file in Day_7 is open in the editor. The code in the editor is:

```
Day_7 > write_csv.py
1 import csv
2
3 csv_file = open('test.csv', 'w')
4
5 writer = csv.writer(csv_file, delimiter=';')
6
7 header = ['Last name', 'First name', 'Age', 'Country']
8
9 data = ['Smith', 'John', '35', 'USA']
10
11 writer.writerow(header)
12
13 writer.writerow(data)
14
15 csv_file.close()
```

The Terminal tab at the bottom shows the command being run: 'gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_7', 'gitpod /workspace/python-for-OSINT-21-days/Day_7 (main) \$ python write_csv.py', and 'gitpod /workspace/python-for-OSINT-21-days/Day_7 (main) \$'. The terminal output is in blue.

Импортируем пакет CSV (он доступен по умолчанию):

```
import csv
```

Откройте и создайте файл test.csv:

```
csv_file = open('test.csv', 'w')
```

Создайте объект записи CSV:

```
writer = csv.writer(csv_file, delimiter=',')
```

Создайте список с заголовками данных:

```
header = ['Last name', 'First name', 'Age', 'Country']
```

Создайте список с одной строкой данных:

```
data = ['Smith', 'John', '35', 'USA']
```

Запишите заголовки данных в файл:

```
writer.writerow(header)
```

Записываем в файл строку данных:

```
writer.writerow(data)
```

Закройте файл test.csv:

```
csv_file.close()
```

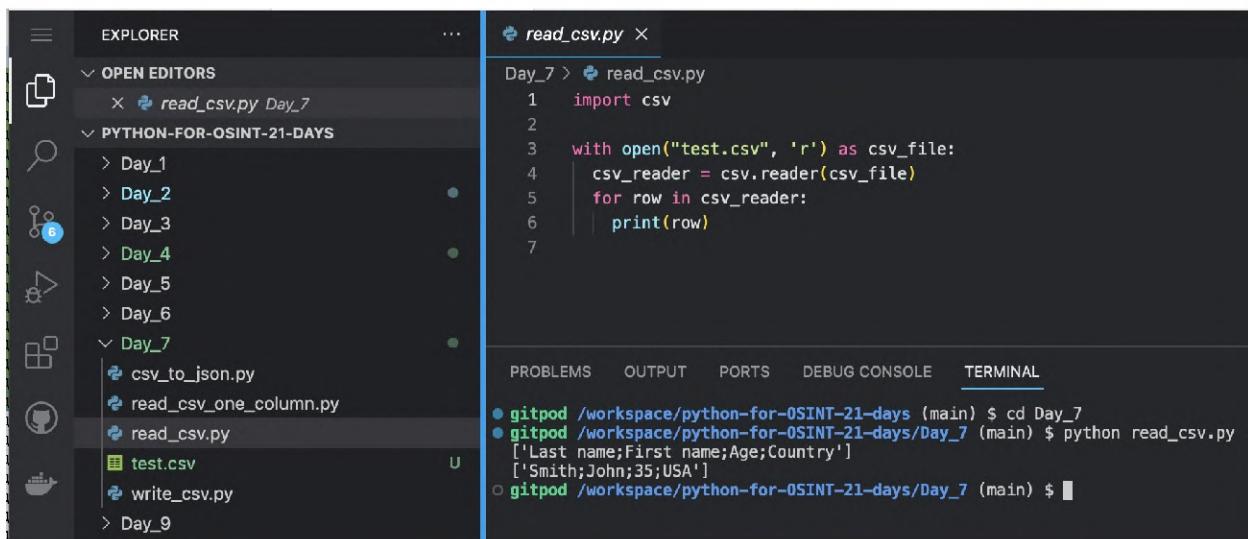
Созданный таким образом CSV-файл можно открыть в любом редакторе электронных таблиц: Excel, Numbers, Google Sheet и т. д.

Last name	First name	Age	Country
Smith	John	35	USA

Теперь попробуем прочитать содержимое CSV-файла.

Чтение CSV-файла

Запустите `read_csv.py`:



The screenshot shows the VS Code interface. The Explorer sidebar on the left displays a project structure under 'OPEN EDITORS' and 'PYTHON-FOR-OSINT-21-DAYS'. The 'Day_7' folder contains several files: `csv_to_json.py`, `read_csv_one_column.py`, `read_csv.py`, `test.csv`, and `write_csv.py`. The 'TERMINAL' tab at the bottom shows the command-line output of running the script:

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_7
gitpod /workspace/python-for-OSINT-21-days/Day_7 (main) $ python read_csv.py
['Last name;First name;Age;Country']
['Smith;John;35;USA']
```

Импортируем пакет csv:

```
import csv
```

Откройте файл test.csv:

```
with open("test.csv", 'r') as csv_file:
```

Создайте объект csv.reader:

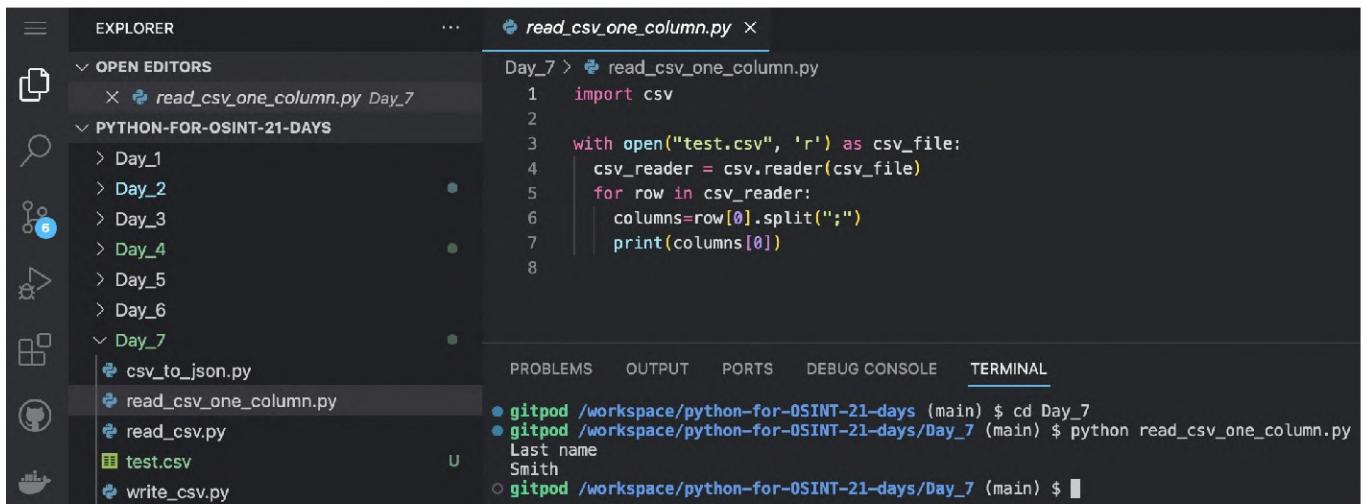
```
csv_reader = csv.reader(csv_file)
```

Печатаем строки одну за другой:

```
for row in csv_reader:  
    print(row)
```

Теперь попробуем считать данные из отдельного столбца.

Запустите `read_csv_one_column.py`:



The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view of files and folders. In the center is the code editor with the file `read_csv_one_column.py` open. The code reads a CSV file, creates a CSV reader, loops through rows, splits each row by the delimiter (";"), and prints the first column. On the right is the Terminal panel showing the command `python read_csv_one_column.py` being run and its output: "Last name Smith".

```
import csv  
with open("test.csv", 'r') as csv_file:  
    csv_reader = csv.reader(csv_file)  
    for row in csv_reader:  
        columns=row[0].split(",")  
        print(columns[0])
```

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_7  
gitpod /workspace/python-for-OSINT-21-days/Day_7 (main) $ python read_csv_one_column.py  
Last name  
Smith  
gitpod /workspace/python-for-OSINT-21-days/Day_7 (main) $
```

Импортируем пакет CSV:

```
import csv
```

Откройте файл test.csv:

```
with open("test.csv", 'r') as csv_file:
```

Создайте объект чтения CSV:

```
csv_reader = csv.reader(csv_file)
```

Поочередно делим строку на столбцы, используя разделитель - точку с запятой:

```
for row in csv_reader:  
    columns=row[0].split(",")
```

```
# И напечатаем первый столбец:
```

```
print(columns[0])
```

```
Day_7 > test.csv
1 javascript;https://github.com/javascript;https://avatars.githubusercontent.com/u/48139711?v=4
2 javascript-tutorial;https://github.com/javascript-tutorial;https://avatars.githubusercontent.com/u/49552305?v=4
3 javascripter;https://github.com/javascripter;https://avatars.githubusercontent.com/u/130407?v=4
4 javascriptit;https://github.com/javascriptit;https://avatars.githubusercontent.com/u/14077111?v=4
5 JavaScriptErika;https://github.com/JavaScriptErika;https://avatars.githubusercontent.com/u/19316487?v=4
6 hyperlink;https://github.com/hyperlink;https://avatars.githubusercontent.com/u/1333097?v=4
7 JavascriptDeNoobAPro;https://github.com/JavascriptDeNoobAPro;https://avatars.githubusercontent.com/u/72440926?v=4
8 javascriptDev;https://github.com/javascriptDev;https://avatars.githubusercontent.com/u/43432007?v=4
9 javascriptdezero;https://github.com/javascriptdezero;https://avatars.githubusercontent.com/u/44194645?v=4
10 javascript-forks;https://github.com/javascript-forks;https://avatars.githubusercontent.com/u/6993497?v=4
11 AmyDayday;https://github.com/AmyDayday;https://avatars.githubusercontent.com/u/4569236?v=4
12 javascript666666;https://github.com/javascript666666;https://avatars.githubusercontent.com/u/25980255?v=4
13 JavascriptMick;https://github.com/JavascriptMick;https://avatars.githubusercontent.com/u/9057287?v=4
14 javascripto;https://github.com/javascripto;https://avatars.githubusercontent.com/u/16804522?v=4
15 javascriptacademy-stash;https://github.com/javascriptacademy-stash;https://avatars.githubusercontent.com/u/706003807?v=4
16 JavascriptIsMagic;https://github.com/JavascriptIsMagic;https://avatars.githubusercontent.com/u/2086868?v=4
17 luckysc;https://github.com/luckysc;https://avatars.githubusercontent.com/u/22893023?v=4
18 javascripteverywhere;https://github.com/javascripteverywhere;https://avatars.githubusercontent.com/u/44604333?v=4
19 javascript-ninja-123;https://github.com/javascript-ninja-123;https://avatars.githubusercontent.com/u/19541825?v=4
20 javascripttextjs;https://github.com/javascripttextjs;https://avatars.githubusercontent.com/u/13724723?v=4
21 javascript-indonesias;https://github.com/javascript-indonesias;https://avatars.githubusercontent.com/u/28702747?v=4
22 javascriptisscary;https://github.com/javascriptisscary;https://avatars.githubusercontent.com/u/13318285?v=4
23 javascript-queen;https://github.com/javascript-queen;https://avatars.githubusercontent.com/u/906146207?v=4
24 javascriptlove;https://github.com/javascriptlove;https://avatars.githubusercontent.com/u/18117327?v=4
25 JavaScript-Packer;https://github.com/JavaScript-Packer;https://avatars.githubusercontent.com/u/120142347?v=4
26 javascriptjedi;https://github.com/javascriptjedi;https://avatars.githubusercontent.com/u/4425037?v=4
27 JavaScript-Resource;https://github.com/JavaScript-Resource;https://avatars.githubusercontent.com/u/18742886?v=4
28 JavaScriptNeha;https://github.com/JavaScriptNeha;https://avatars.githubusercontent.com/u/264874282?v=4
```

JSON в CSV

Иногда вам необходимо преобразовать данные из JSON в CSV, чтобы их можно было удобно просматривать и открывать в Microsoft Excel/Google Sheet.

Сделать это можно с помощью специальных сервисов типа csvjson.com (и это будет лучшим решением).

Но я покажу вам, как это сделать с помощью кода Python, чтобы закрепить то, что вы узнали за последние два дня.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists a workspace named 'PYTHON-FOR-OSINT-21-DAYS' containing sub-folders 'Day_1' through 'Day_17'. The 'Day_7' folder is currently expanded, showing files like 'json_to_csv.py', 'read_csv_one_column.py', 'read_csv.py', 'test.csv', and 'write_csv.py'. The Editor pane on the right contains the 'json_to_csv.py' script. The Terminal pane at the bottom shows the command-line history:

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_7
gitpod /workspace/python-for-OSINT-21-days/Day_7 (main) $ python json_to_csv.py
gitpod /workspace/python-for-OSINT-21-days/Day_7 (main) $
```

Запустите json_to_csv.py:

Импорт пакетов json, csv и запросов:

```
import json
import requests
import csv
```

Сделайте запрос к Github API:

```
response =
requests.get("https://api.github.com/search/users?q=javascript")
```

Получить данные в формате JSON:

```
json_data=response.json()
```

```
# Откройте и одновременно создайте файл test.csv:
```

```
csv_file = open('test.csv', 'w')
```

```
# Создайте объект csv_writer:
```

```
writer = csv.writer(csv_file, delimiter =';')
```

```
# Подсчитаем количество найденных пользователей:
```

```
usersCount = len(json_data['items'])-1
```

```
# Передайте каждую строку данных JSON одну за другой, создайте пустой строковый объект, добавьте логин, ссылку на профиль и ссылку на аватар, запишите строку в файл csv:
```

```
for x in range(usersCount):
    row = []
    row.append(json_data['items'][x]['login'])
    row.append(json_data['items'][x]['html_url'])
    row.append(json_data['items'][x]['avatar_url'])
    writer.writerow(row)
```

```
# Закройте файл test.csv:
```

```
csv_file.close()
```

```
1 javascript;https://github.com/javascript;https://avatars.githubusercontent.com/u/48139711?v=4
2 javascript-tutorial;https://github.com/javascript-tutorial;https://avatars.githubusercontent.com/u/49552305?v=4
3 javascripter;https://github.com/javascripter;https://avatars.githubusercontent.com/u/130407?v=4
4 javascriptit;https://github.com/javascriptit;https://avatars.githubusercontent.com/u/14077117?v=4
5 JavaScriptErika;https://github.com/JavaScriptErika;https://avatars.githubusercontent.com/u/19316487?v=4
6 hyperlink;https://github.com/hyperlink;https://avatars.githubusercontent.com/u/1333097?v=4
7 JavascriptDeNoobAPro;https://github.com/JavascriptDeNoobAPro;https://avatars.githubusercontent.com/u/72440926?v=4
8 javascriptDev;https://github.com/javascriptDev;https://avatars.githubusercontent.com/u/43432007?v=4
9 javascriptdezero;https://github.com/javascriptdezero;https://avatars.githubusercontent.com/u/44194645?v=4
10 javascript-forks;https://github.com/javascript-forks;https://avatars.githubusercontent.com/u/69934977?v=4
11 AmyDayday;https://github.com/AmyDayday;https://avatars.githubusercontent.com/u/45692367?v=4
12 javascript66666;https://github.com/javascript66666;https://avatars.githubusercontent.com/u/25980255?v=4
13 JavascriptMick;https://github.com/JavascriptMick;https://avatars.githubusercontent.com/u/9057287?v=4
14 javascript;https://github.com/javascripto;https://avatars.githubusercontent.com/u/16804522?v=4
15 javascriptacademy-stash;https://github.com/javascriptacademy-stash;https://avatars.githubusercontent.com/u/70600380?v=4
16 luckyssc;https://github.com/luckyssc;https://avatars.githubusercontent.com/u/228930237?v=4
17 JavascriptIsMagic;https://github.com/JavascriptIsMagic;https://avatars.githubusercontent.com/u/20868687?v=4
18 javascriptoverwhere;https://github.com/javascriptoverwhere;https://avatars.githubusercontent.com/u/44604333?v=4
19 javascript-indonesias;https://github.com/javascript-indonesias;https://avatars.githubusercontent.com/u/28702747?v=4
20 javascript-ninja-123;https://github.com/javascript-ninja-123;https://avatars.githubusercontent.com/u/19541625?v=4
21 javascriptextjs;https://github.com/javascriptextjs;https://avatars.githubusercontent.com/u/13724723?v=4
22 javascriptisscary;https://github.com/javascriptisscary;https://avatars.githubusercontent.com/u/133182857?v=4
23 javascript-queen;https://github.com/javascript-queen;https://avatars.githubusercontent.com/u/986146207?v=4
24 javascriptLove;https://github.com/javascriptLove;https://avatars.githubusercontent.com/u/18117327?v=4
25 JavaScript-Resource;https://github.com/JavaScript-Resource;https://avatars.githubusercontent.com/u/18742886?v=4
26 JavaScript-Packer;https://github.com/JavaScript-Packer;https://avatars.githubusercontent.com/u/120142347?v=4
```

Вот как должно выглядеть содержимое файла test.csv после запуска `csv_to_json.py` скрипта.

День 8. Базы данных

Существует множество пакетов Python для работы практически со всеми популярными базами данных: MySQL, PostgreSQL, MongoDB, Redis, Elasticsearch и т. д. В этом курсе я решил не рассматривать примеры кода для какой-то одной базы данных, а просто дать несколько универсальных советов.

Файлы формата SQL (язык структурированных запросов) очень часто встречаются в расследованиях. В этом формате хранятся дампы базы данных, в которых часто можно найти полезную контактную информацию (распространенный пример — список адресов электронной почты и номеров телефонов сотрудников компании).

Например, иногда они могут оказаться в результатах поиска Google.

The screenshot shows a Google search results page with the query "@gmail.com filetype:sql". The results include:

- ragebooter.txt**
... Dumping data for table `users` -- INSERT INTO `users` ('ID', `username`, ... 'anna-ellington-jones@hotmail.com', 0, 29, 1382677259, 0, "", 0), (108, ...)
- GitLab**
https://gitlab.com › GitLab.org › GitLab :
db/structure.sql · master · GitLab.org / GitLab
Quickly and easily edit multiple files in your project. ... Edit this file only.
- alohachicas.com**
https://alohachicas.com › search · Translate this page :
"也门高端男士休闲会所(微信696360) ...
ALOHA SHOP. C/Velázquez Moreno 26. 36202 – Vigo (Pontevedra). +34 637 914 746. +34 692 219 176. pedidosalohashop@gmail.com ...
- The University of Western Australia**
https://teaching.csse.uwa.edu.au › projects › juicdump3 :
SQL dump

Если вы выполните поиск в [базе данных хакеров Google](#) по запросу «sql» (особенно в разделе Juicy Info Dorks), вы найдете более 1500 примеров запросов для поиска данных в файлах .sql.

Google Hacking Database			
Show	15	Filters	Reset All
Date Added	Dork	Category	Author
2023-04-20	Re: Thank you for your submission! Re: intitle:index of db.sqlite3	Files Containing Juicy Info	Shebu
2023-03-14	intitle:"index of" "database.sql"	Files Containing Juicy Info	Prathamesh Pawar
2023-02-22	inurl:backup filetype:sql	Files Containing Juicy Info	Nox Mentor
2023-02-15	intext:index of" "backup/*.sql"	Files Containing Juicy Info	Ahmad Kataranje
2023-02-01	inurl: administrator/components/com_admin/sql/uploads/sqlazure	Files Containing Juicy Info	Mark Ivan David
2023-02-01	inurl: administrator/components/com_admin/sql/uploads/mysql/	Files Containing Juicy Info	Mark Ivan David
2022-09-19	inttext:index of" ".sql"	Files Containing Juicy Info	Gopalsamy Rajendran
2022-06-16	"index of" filetype:sql	Files Containing Juicy Info	Girish B O
2022-06-15	inttitle:"index of" filetype:sql	Files Containing Juicy Info	Ract Hack
2022-06-16	site:com.* inttitle: index of* *.sql	Files Containing Juicy Info	Girish B O
2022-06-16	inttext:"SQL" && "DB" inurl:"/runtime/log/"	Files Containing Juicy Info	Vitor guaxi
2021-11-15	inttitle:"index of" "/mysql"	Files Containing Juicy Info	Priyanshu Choudhary
2021-11-10	inttitle:"index of" " mod_auth_mysql "	Files Containing Juicy Info	Muhammad Al-Amin
2021-11-08	inttitle:"database" "backup" filetype:sql	Files Containing Juicy Info	Onkar Deshmukh
2021-11-05	inurl:admin ext:sql	Files Containing Juicy Info	Veeresh Appasaheb Patil

Showing 1 to 15 of 87 entries (filtered from 1,536 total entries)

FIRST PREVIOUS 1 2 3 4 5 6 NEXT LAST

Самый простой способ просмотреть такой файл — просто преобразовать его в CSV и открыть в Excel/Numbers/Google Sheet. Этот бесплатный онлайн-конвертер поможет вам. Например, [Rebsase SQL в CSV](#).

 RebaseData

Converters API Libraries Pricing Security Support About [Login](#) [Register](#)

Convert MySQL to CSV online

Choose the .SQL file..

Convert

Input file

Our API uses a .SQL file as input. This could be a MySQL dump from the mysqldump tool or an output of any other tool that returns SQL. The file should contain CREATE TABLE and INSERT/UPDATE statements.

Max file size for web uploads: 50 GB
[Register](#) to upload big files via Amazon S3.

Output file

The API will return a ZIP archive of .CSV files, one for each input file. The CSV files comply with our [CSV specification](#).

[Help](#)

Также полезна для расследования контактная информация, которую можно хранить, и базы данных других форматов: MS Access (.MDB), SQL Server (.MDF), SQLite (.sqlite, .sqlite3, .db, .db3, .s3db, .sl3), Firebird (.FBD) и многие другие.

The screenshot shows the RebaseData website interface. At the top is a yellow logo consisting of three stacked circles. Below it, the text "RebaseData" is displayed. The page is divided into several sections:

- MS Access converters**: A list of conversion options from Microsoft Access to various formats like CSV, Excel, MySQL, PostgreSQL, etc.
- SQLite converters**: A list of conversion options from SQLite to formats like CSV, Excel, MySQL, PostgreSQL, etc.
- MS SQL Server converters**: A list of conversion options from Microsoft SQL Server (MSSQL) to formats like CSV, Excel, MySQL, PostgreSQL, etc.
- Firebird and Interbase converters**: A list of conversion options from Firebird and Interbase databases to formats like CSV, Excel, MySQL, PostgreSQL, etc.

Each section includes a brief description of the service: "Convert your Microsoft Access files to other formats online without having to install Microsoft Access yourself. Privacy and security is ensured." or "Convert your SQLite database to other formats online without having to install SQLite yourself." etc.

Их также можно конвертировать в CSV с помощью онлайн-конвертеров:

Rebasedata.com

Anyconv.com

101convert.com

В этом уроке мы не будем запускать пример кода. В качестве практики я просто порекомендую вам найти в Google файлы баз данных с данными контактов, посмотреть, как они

упорядочены и конвертировать их в CSV. Используйте оператор filetype:pdf и примеры запросов из базы данных хакеров Google.

День 9. Автоматизируйте сбор результатов поиска

Существует огромное количество инструментов Python для сбора результатов поиска из разных поисковых систем. Многие из них предназначены для поиска уязвимых сайтов и пикантной информации (например, таблиц с личными контактными данными) с помощью Google Dorks.

Они экономят огромное количество времени, просматривая результаты поиска. Потому что они могут автоматически анализировать содержимое найденных веб-страниц.

Вот несколько примеров:

Email Finder (<https://github.com/Josue87/EmailFinder>) — поиск электронных писем из домена с помощью Google, Bing, Baidu.

StartPageParser (<https://github.com/knassar702/startpage-parser>) — собирает результаты поиска из поисковой системы стартовой страницы (на основе результатов google.com). Это позволяет вам собирать результаты поиска Google, не думая о том, что Google забанит вас.

Поисковик (<https://github.com/davemolk/searcher>) — собираете результаты поиска из Ask, Bing, Brave, Duck Duck Go, Yahoo и Яндекс.

DDGR (<https://github.com/jarun/ddgr>) — собираете результаты поиска DuckDuckGo.

Парсер поисковых систем (<https://github.com/tasos-py/Search-Engines-Scraper>) — собирает результаты поиска из 11 поисковых систем.

Сегодня мы научимся использовать пакет Duckduckgo-search.

<https://pypi.org/project/duckduckgo-search/>. На мой взгляд, это один из лучших вариантов по сочетанию «удобство использования» + «мощность функциональности».

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_9
● gitpod /workspace/python-for-OSINT-21-days/Day_9 (main) $ pip install duckduckgo_search
Collecting duckduckgo_search
  Downloading duckduckgo_search-2.9.2-py3-none-any.whl (30 kB)
Requirement already satisfied: click>=8.1.3 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from duckduckgo_search) (8.1.3)
Collecting diskcache>=5.6.1 (from duckduckgo_search)
  Downloading diskcache-5.6.1-py3-none-any.whl (45 kB)
     ┌─────────────────────────────────────────────────────────────────────────┐
     │          45.6/45.6 KB 2.1 MB/s eta 0:00:00
     └─────────────────────────────────────────────────────────────────┘
Collecting requests>=2.29.0 (from duckduckgo_search)
  Downloading requests-2.29.0-py3-none-any.whl (62 kB)
     ┌─────────────────────────────────────────────────────────────────┐
     │          62.5/62.5 KB 3.5 MB/s eta 0:00:00
     └─────────────────────────────────────────────────────────┘
Requirement already satisfied: charset-normalizer<4,>=2 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from requests>=2.29.0->duckduckgo_search) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests>=2.29.0->duckduckgo_search) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests>=2.29.0->duckduckgo_search) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests>=2.29.0->duckduckgo_search) (2022.12.7)
Installing collected packages: requests, diskcache, duckduckgo_search
  Attempting uninstall: requests
    Found existing installation: requests 2.28.1
      Uninstalling requests-2.28.1:
        Successfully uninstalled requests-2.28.1
Successfully installed diskcache-5.6.1 duckduckgo_search-2.9.2 requests-2.29.0
○ gitpod /workspace/python-for-OSINT-21-days/Day_9 (main) $
```

Установить пакет из pip:

```
pip install duckduckgo_search
```

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ python -m duckduckgo_search --help
Usage: duckduckgo_search [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]]...
Options:
  --help  Show this message and exit.

Commands:
  answers
  images
  maps
  news
  suggestions
  text
  translate
  version
  videos
○ gitpod /workspace/python-for-OSINT-21-days (main) $
```

Проверьте установку:

```
python -m duckduckgo_search --help
```

Пожалуйста, запомните этот флаг. Он работает для большинства пакетов Python. Если вы наберете `-help` или `-h` после имени команды будет справочная информация о ее использовании отображается.

Запустите `ddg_search.py`:

```
ddg_search.py
Day_9 > ddg_search.py
1 from duckduckgo_search import ddg
2
3 keywords = 'osint'
4 results = ddg(keywords, region='us-en', safesearch='Off', time='y')
5 print(results)
6
```

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_9
gitpod /workspace/python-for-OSINT-21-days/Day_9/main $ python ddg_search.py
[{"title": "What is Open-Source Intelligence?", "href": "https://www.sans.org/blog/what-is-open-source-intelligence/", "body": "Open Source Intelligence (OSINT) is the collection, analysis, and dissemination of information that is publicly available and legally accessible. Right now, OSINT is used by organizations, including governments, businesses, and non-governmental organizations. It is useful in information gathering for a wide range of topics such as security ..."}, {"title": "9 Best OSINT Tools for 2023 (Paid & Free) - Compartech", "href": "https://www.compartech.com/network-admin/osint-tools"}, {"body": "Here's our list of the eight best OSINT tools: OSINT Framework, a website directory of data discovery and gathering tool s for OSINT and open source intelligence. OSINT Framework allows users to create their own OSINT and security tool s based on cloud-based service."}, {"title": "Open Source Intelligence (OSINT) Fundamentals - NICECS", "href": "https://nics.cisa.gov/education-training/catalog/cybersecurity/open-source-intelligence-osint-fundamentals"}, {"body": "Open Source Intelligence (OSINT) Fundamentals. Online, Self-Paced. In this Open Source Intelligence (OSINT) Fundamentals training course, you will gain fundamental knowledge about OSINT, who uses it, and the ethical implications of using it. Upon completion, learners will have a basic understanding of OSINT, including OSINT Penetration Testing, Open Source Intelligence Tools & Techniques."}, {"title": "OSINT Goes Mainstream: How Security Teams Can Use Open-Source Intelligence to Help Companies Understand Risk", "href": "https://www.sans.org/cyber-security-exchange/penetration-testing/penetration-testing-with-open-source-intelligence/1"}, {"body": "According to a 2021 IBM report, the average organization did not detect a data breach for up to 212 days—and they did not fully contain the issue for another 75. In many instances, malicious hackers attack a company using publicly available information—open-source intelligence, often referred to as OSINT—."}, {"title": "Open Source Intelligence: The Beginner's Guide to OSINT", "href": "https://www.liferaffinc.com/blog/the-beginners-guide-to-osint"}, {"body": "Open source intelligence, or OSINT, refers to any information collected from free public sources about an organization or individual to provide actionable insights for decision-makers. Technically, this information could be anything from news stories to social media posts to public records."}]
```

Импортируем пакет ddg:

```
from duckduckgo_search import ddg
```

Создаём переменную с поисковым запросом:

```
keywords = 'osint'
```

Отправляем поисковый запрос, указав, что мы хотим видеть результаты поиска по США с отключенным безопасным поиском:

```
results = ddg(keywords, region='us-en', safesearch='Off', time='y')
```

Распечатываем результаты:

```
print(results)
```

Простое отображение результатов на экране не очень полезно. То же самое можно сделать и в браузере. Попробуем сохранить их в файл CSV, чтобы их можно было автоматически проанализировать позже.

Запустите ddg_search_to_csv.py:

```
ddg_search_to_csv.py
1  from duckduckgo_search import ddg
2  import csv
3
4  csv_file = open('search_results.csv', 'w')
5  writer = csv.writer(csv_file, delimiter=';')
6
7
8  keywords = 'osint'
9  results = ddg(keywords, region='us-en', safesearch='Off', time='y')
10
11 for x in range(len(results)):
12     row = [results[x]['title'], results[x]['body'], results[x]['href']]
13     writer.writerow(row)
14
15 csv_file.close()
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_9
gitpod /workspace/python-for-OSINT-21-days/Day_9 (main) $ python ddg_search_to_csv.py
gitpod /workspace/python-for-OSINT-21-days/Day_9 (main) $ []
```

Импорт модулей ddg и csv:

```
from duckduckgo_search import ddg
import csv
```

Откройте файл search_results.csv:

```
csv_file = open('search_results.csv', 'w')
```

Создайте объект csv.writer:

```
writer = csv.writer(csv_file, delimiter =';')
```

Создаем переменную с поисковыми запросами:

```
keywords = 'osint'
```

Отправляем поисковый запрос, указав, что мы хотим видеть результаты поиска по США с отключенным безопасным поиском:

```
results = ddg(keywords, region='us-en', safesearch='Off', time='y')
```

Просматриваем результаты поиска один за другим и записываем каждый в строку CSV-файла с тремя полями — title,body,href (обратите внимание, что каждый раз, когда вы пишете строку, создается список из трех элементов):

```
for x in range(len(results)):
    row = [results[x]["title"],results[x]["body"],results[x]["href"]]
    writer.writerow(row)
```

Закройте файл search_results.csv:

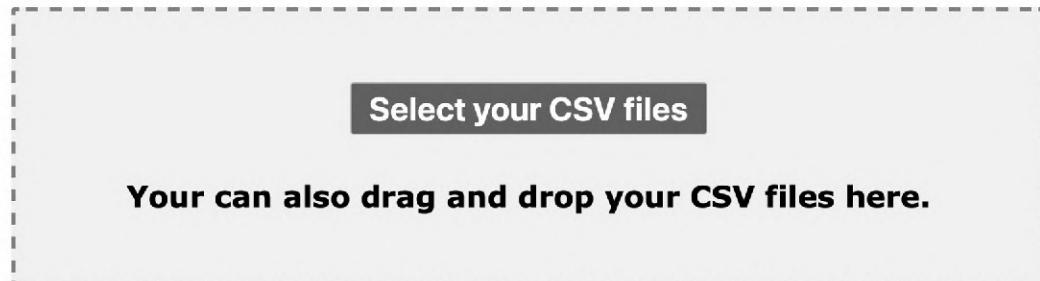
```
csv_file.close()
```

Вот как выглядит файл search_results.csv в Numbers:

The screenshot shows a Numbers spreadsheet with a single sheet named "Sheet 1". The table has two columns: A and B. The header row is labeled "search_results". The data rows consist of pairs of titles and URLs. Column A contains the titles, and Column B contains the corresponding URLs.

	search_results
1	What is OSINT (Open-Source Intelligence?) SANS Institute
2	9 Best OSINT Tools for 2023 (Paid & Free) - Comparitech
3	Top 17 OSINT tools to find anyone online - 2023 - News & Article
4	Spy agencies look to standardize use of open source intelligence
5	What Is Open Source Intelligence: The Importance of OSINT in 2022
6	What is OSINT? Defining open-source intelligence in 2022 - Social Links
7	Types of Intelligence Collection - Intelligence Studies - LibGuides
8	Osint (オシント) とは?!? メールアドレスだけで個人情報流出を防ぐ方法
9	Open-Source Intelligence (OSINT) - Cyber
10	Ukraine OSINT.org
11	Open Source Intelligence: The Beginners' Guide to OSINT
12	Separating OSINT from the Secret World Strengthens Both
13	What is OSINT? Open-Source Intelligence Tools 2023 Gridinsoft
14	12 Resourceful OSINT Tools You Should Know - MUO
15	What is Open Source Intelligence, and how is it used? - News & Article
16	Open-Source Intelligence is Indispensable for Countering Threats
17	GitHub - Datalux/Osintgram: Osintgram is a OSINT tool on Instagram
18	9 Open Source Intelligence (OSINT) Tools for Penetration Testing
19	OSINT Goes Mainstream: How Security Teams Can Use Open-Source Intelligence
20	Penetration Testing with Open-Source Intelligence (OSINT): Tips and Techniques
21	What is OSINT Open Source Intelligence? CrowdStrike

Python позволяет легко находить файлы в папках и манипулировать ими (подробнее об этом мы поговорим в уроке 14), но иногда удобнее просто объединить несколько CSV-файлов в один, чтобы сэкономить время при написании кода.



Merge CSV options:

Keep only the header (first row) of the first file

Remove duplicate rows

Remove empty rows

End line

Merge & Download

Вы можете сделать это с помощью любой службы, которую можете найти в Google, выполнив поиск «Объединить файлы CSV онлайн». Например, <https://extendsclass.com/merge-csv.html>

Этот пакет также позволяет загружать содержимое страниц из результатов поиска. Вы можете загрузить все найденные файлы (html, pdf, xlsx и т. д.), а затем автоматически проанализировать их или просто выполнить в них простой поиск по ключевым словам.

Запустите ddg_search_download_pdf.py:

```
ddg_search_to_csv.py ddg_search_download_pdf.py M X
Day_9 > ddg_search_download_pdf.py
1 from duckduckgo_search import ddg
2
3 keywords = 'open source intelligence filetype:pdf'
4 results = ddg(keywords, region='us-en', safesearch='Off', time='y', download=True)
5

gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_9
gitpod /workspace/python-for-OSINT-21-days/Day_9 (main) $ python ddg_search_download_pdf.py
Downloading documents [########################################] 28/28 100%
gitpod /workspace/python-for-OSINT-21-days/Day_9 (main) $
```

Импортируем пакет ddg:

```
from duckduckgo_search import ddg
```

Создайте переменную с поисковыми запросами (включите filetype:pdf):

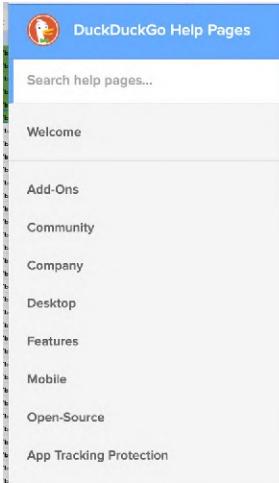
```
keywords = 'open source intelligence filetype:pdf'
```

Отправьте поисковый запрос, указав, что мы хотим видеть результаты поиска для США с включенной опцией загрузки:

```
results = ddg(keywords, region='us-en', safesearch='Off', time='y',
download=True)
```

Чтобы изменить количество загружаемых файлов, установите параметр max_results в ddg().

С помощью пакетов Duckduckgo_search вы также можете собирать ответы, изображения, видео, новости, карты, предложения и переводить текст.



Search Operators

Example	Result
cats dogs	Results about cats or dogs
"cats and dogs"	Results for exact term "cats and dogs". If no results are found, we'll try to show related results.
cats -dogs	Fewer dogs in results
cats +dogs	More dogs in results
cats filetype:pdf	PDFs about cats. Supported file types: pdf, doc(x), xls(x), ppt(x), html
dogs site:example.com	Pages about dogs from example.com
cats - site:example.com	Pages about cats, excluding example.com
intitle:dogs	Page title includes the word "dogs"
inurl:cats	Page url includes the word "cats"

Обратите внимание, что в последнем примере мы использовали оператор расширенного поиска filetype:pdf. Другие операторы расширенного поиска также можно использовать в запросах для `duckduckgo_search`.

[Список операторов поиска DuckDuckGo](#)

Стоит отметить, что умение использовать операторы расширенного поиска при каждом удобном случае – очень полезный навык для каждого OSINT-специалиста. Вот список справочных статей с операторами расширенного поиска для поисковых систем, социальных сетей, почтовых ящиков и других сервисов:

[Список операторов расширенного поиска](#)

День 10. Scraping

Скрапинг — это извлечение данных с веб-сайта.

Самое важное, что нужно знать о парсинге и Python, — написание собственного скрипта с нуля для каждой задачи чаще всего — не лучшее решение. Лучше сначала попробовать разные готовые средства.

Расширение Chrome для веб-скребка

(<https://chrome.google.com/webstore/detail/web-scraper-free-web-скра/jnhgnonknehpejjnehehllkliplmbmhn>)

Пока я пишу этот курс, мир быстро меняется, и парсинг с помощью ИИ активно развивается.

Обзор ИИ

<https://www/browse.ai>

AnyPicker <https://chrome.google.com/webstore/detail/anypicker-ai-power-no-c/bjkpgfhhekfmddfdphnniobddhkjlmmij>

ScrapeStorm

<https://www.scrapestorm.com>

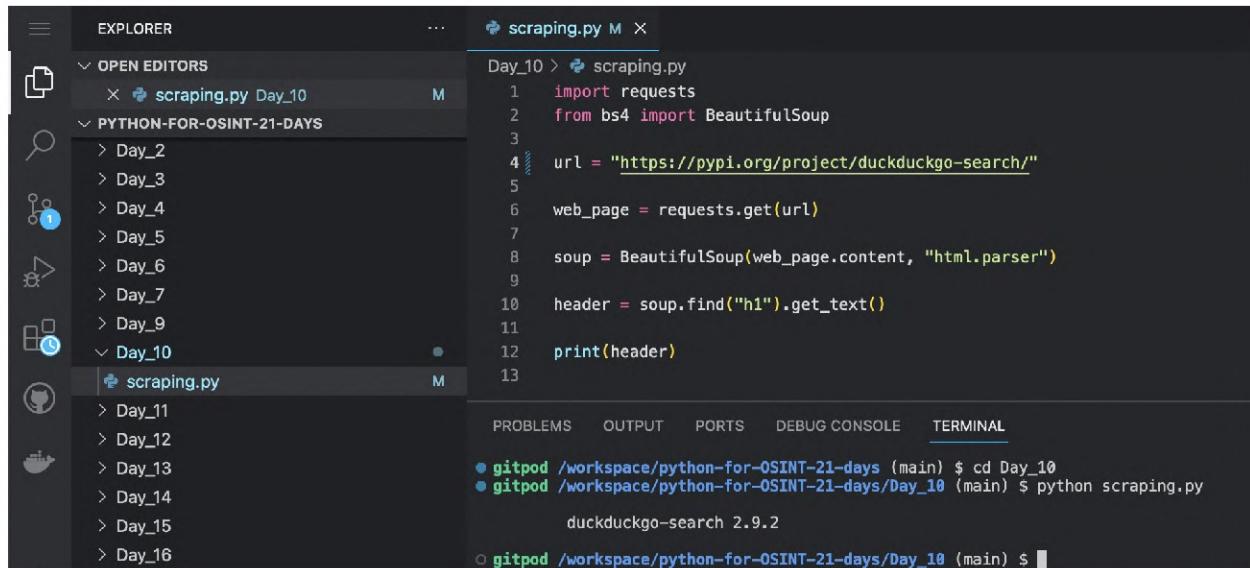
И почти каждый день появляется какой-нибудь новый инструмент парсинга AI.

Если вам нужно собрать данные из каких-либо популярных социальных сетей, попробуйте найти любое решение, созданное специально для конкретной платформы. Например, инструмент YouTube (https://github.com/nlitsme/youtube_tool) для YouTube или Stweet (<https://github.com/markowanga/stweet>) для Twitter.

Но иногда вы можете столкнуться с проблемой, для которой нет готовых решений и вам захочется написать собственный скрипт для ее решения. Существует множество пакетов Python для парсинга: Scrapy, Selenium, ZenRows и т. д.

Для парсинга мы будем использовать пакет BeautifulSoup (<https://pypi.org/project/beautifulsoup4/>). Он установлен по умолчанию.

Пакет BeautifulSoup также может быть полезен для работы с данными в формате XML (вы можете столкнуться с этим, в частности, при получении данных из какого-либо API). В этом случае помимо BeautifulSoup вам следует использовать пакет LXML (<https://lxml.de>).



```
Day_10 > 🏷 scraping.py M
1 import requests
2 from bs4 import BeautifulSoup
3
4 url = "https://pypi.org/project/duckduckgo-search/"
5
6 web_page = requests.get(url)
7
8 soup = BeautifulSoup(web_page.content, "html.parser")
9
10 header = soup.find("h1").get_text()
11
12 print(header)
13
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_10
● gitpod /workspace/python-for-OSINT-21-days/Day_10 (main) $ python scraping.py
duckduckgo-search 2.9.2
○ gitpod /workspace/python-for-OSINT-21-days/Day_10 (main) $
```

Запустите Scraping.py:

```
# Запросы на импорт и пакеты BeautifulSoup:
```

```
import requests
from bs4 import BeautifulSoup
```

```
# Создайте переменную с URL-адресом веб-страницы:
```

```
url = "https://pypi.org/project/duckduckgo-search/"
```

```
# Делаем https-запрос:
```

```
web_page = requests.get(url)
```

```
# Создайте объект html.parser:
```

```
soup = BeautifulSoup(web_page.content, "html.parser")
```

```
# Найдите заголовок h1 в html-коде веб-страницы:
```

```
header = soup.find("h1").get_text()
```

```
# Распечатываем заголовок h1:
```

```
print(header)
```

Мы используем CSS-селекторы для поиска элементов на веб-странице:

«h1» — элемент с тегом h1.

«.headers» — элементы с классом заголовков.

«#header» — элементы с идентификатором заголовка.

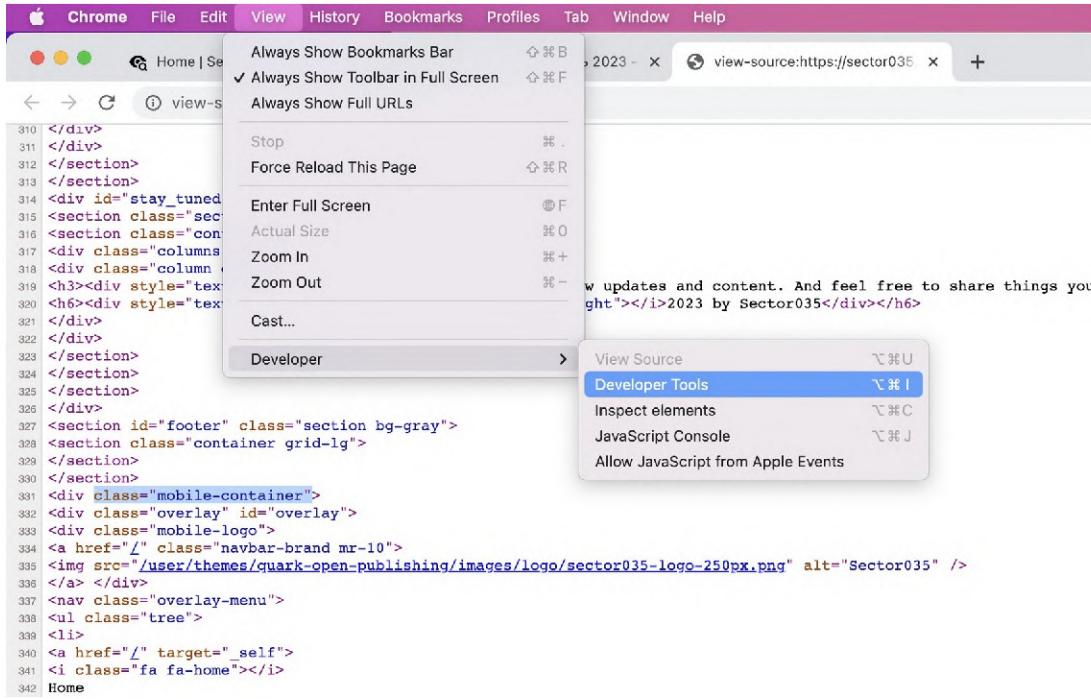
«div.redblocks» — элементы с тегом div и классом redblocks.

«[autofocus="true"]» — элементы с атрибутом autofocus со значением "истинный"

Еще CSS-селекторы:

<https://www.freecodecamp.org/news/css-selectors-cheat-sheet/>

Самый простой способ узнать, какой селектор соответствует определенному html-элементу, — посмотреть исходный код страницы с помощью инструментов разработчика, которые есть в каждом популярном браузере.



А для парсинга страниц со сложной структурой (содержащих множество вложенных элементов) можно использовать специальные расширения для браузера, отображающие полный «путь» к элементу. Например, [HTML-навигация по DOM](#).

SectorQ35

Home Week in OSINT Articles Links Search

Welcome!

IRB Format

```
.div(:id=>'page-wrapper').section(:id=> '').section(:class=>'container grid-lg').h1
```

Immediate Ancestors (Parent, GrandParent, Great-GrandParent and Great-Great-GrandParent)

```
div(:id=>'page-wrapper')
  Great-Grandparent
    ▲
  section(:id=> '', :class=>'section modular-hero hero text-dark', :style=> '')
    Grandparent
      ▲
    section(:class=>'container grid-lg', :style=>'text-align: center')
      Direct Parent
        ▲
      h1
      Current Element
```

other information!

SINT

Часто бывает, что код, который отображается при загрузке сайта Python-скриптами, сильно отличается от того, что отображается в браузере. Это вызвано тем, что некоторые элементы добавляются после загрузки страницы путем выполнения кода JavaScript.

Чтобы понять, что я имею в виду, попробуйте открыть код учетной записи Twitter в расширении [View Rendered Source](#).

The screenshot shows the 'View Rendered Source' extension's interface. It has three main sections: 'Raw', 'Rendered', and 'Difference'.

- Raw:** Shows the raw HTML received from the server. It includes various script tags and CSS links, such as 'RichTextCompose.js', 'ComposeMedia.js', and 'ComposeSchooling.js'.
- Rendered:** Shows the final HTML as it appears in the browser. This version is more polished, with styles applied and dynamic content like user profiles and tweets rendered.
- Difference:** Shows the differences between the raw and rendered versions. It highlights changes in red (removed) and green (added). For example, it shows the addition of CSS classes like 'css-1dbj04n' and 'css-1dbj04r' and the removal of some server-side code.

С его помощью вы можете наглядно сравнить, как выглядит HTML-код сразу после получения запроса от сервера, и как он выглядит после выполнения определенных действий на странице (попробуйте немного прокрутить ленту вниз и снова перезапустить расширение).

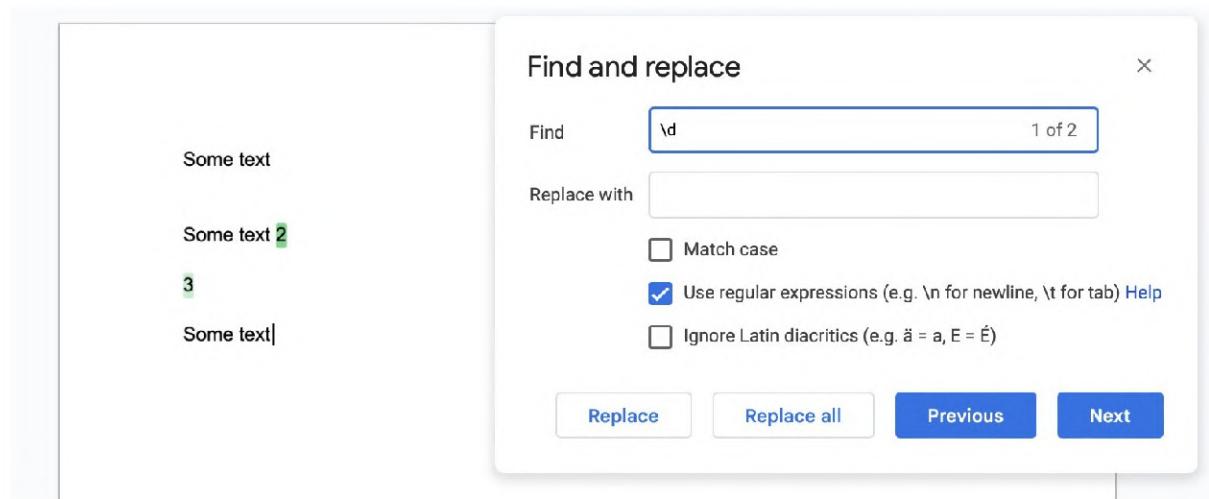
Для парсинга веб-сайтов, код которых сильно меняется после выполнения кода JavaScript в браузере, вы можете использовать такие пакеты, как Selenium (<https://selenium-python.readthedocs.io>). Он позволяет использовать Python для открытия разных браузеров и имитации действий пользователя в них.

День 11. Регулярные выражения

Регулярное выражение — это последовательность символов, позволяющая искать, извлекать и заменять в исходном документе фрагменты текста, соответствующие определенным шаблонам.

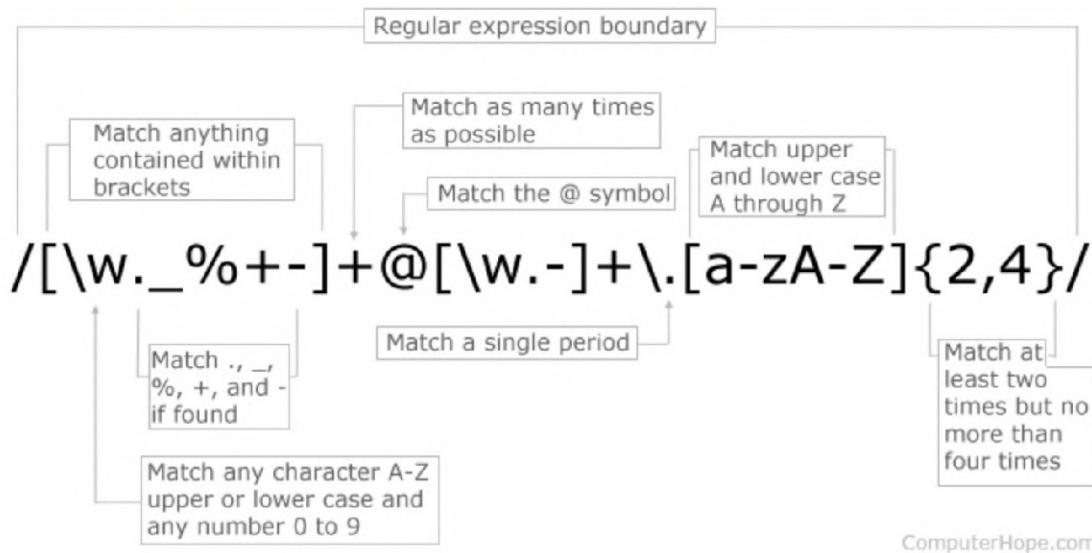
Регулярные выражения поддерживаются большинством популярных языков программирования и используются для поиска, проверки и извлечения различных данных. Например, номера телефонов, адреса электронной почты, IP-адреса, номера криптовалютных кошельков и т. д.

Один из простых способов попробовать регулярные выражения — создать новый документ Google, вставить в него текст с буквами и цифрами, а затем запустить «Найти и заменить».



\d — позволяет найти цифры. Вот краткая схема, которая дает вам базовое понимание синтаксиса регулярных выражений:

Regular Expression E-mail Matching Example



[Источник](#) этого изображения.

Для работы с регулярными выражениями в Python мы будем использовать пакет Re (<https://docs.python.org/3/library/re.html>). Он доступен в Python по умолчанию.

Запустите extract_emails.py:

The screenshot shows a Python development environment with the following details:

- EXPLORER**: Shows a tree view of files under "PYTHON-FOR-OSINT-21-DAYS". Opened files are "scraping.py" and "extract_emails.py".
- OPEN EDITORS**: Displays two tabs: "scraping.py" and "extract_emails.py".
- PYTHON-FOR-OSINT-21-DAYS**: A folder containing sub-folders for each day from Day_2 to Day_11.
- extract_emails.py** (active tab):

```
Day_11 > extract_emails.py
1 import requests
2
3 import re
4
5 |
6 url = "https://cleantalk.org/blacklists/ivanov@gmail.com"
7
8 html = requests.get(url).text
9
10 result = re.findall("[a-zA-Z0-9_.]+@[a-zA-Z0-9_.]+", html)
11
12 print(result)
13
```
- PROBLEMS**, **OUTPUT**, **PORTS**, **DEBUG CONSOLE**, **TERMINAL**: Standard IDE tabs.
- TERMINAL**: Shows command-line history for "gitpod" and "gitpod" commands, followed by a list of email addresses extracted from the URL.

В результате появится список всех адресов электронной почты, найденных на <https://cleantalk.org/blacklists/ivanov@gmail.com>

Импорт запросов и повторных пакетов:

```
import requests  
import re
```

Создайте переменную со ссылкой на страницу, с которой мы хотим получить данные:

```
url = "https://cleantalk.org/blacklists/ivanov@gmail.com"
```

Запрашиваем страницу и помещаем код в переменную:

```
html = requests.get(url).text
```

Попробуйте найти адреса электронной почты в коде:

```
result = re.findall("[a-zA-Z0-9-.]+@[a-zA-Z0-9-.]+", html)
```

Отображение найденных адресов электронной почты:

```
print(result)
```

Попробуйте изменить переменную URL и перезапустить скрипт.

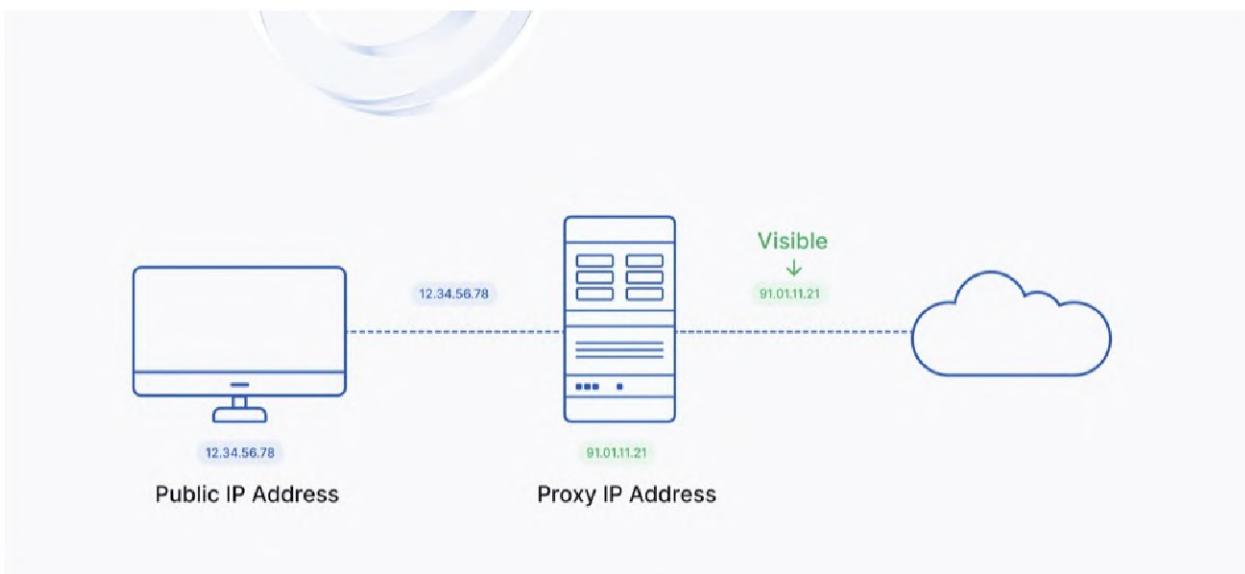
В этом уроке только один пример кода, так как ваша главная цель сегодня — детально изучить, как используются регулярные выражения в OSINT.

Пожалуйста, прочитайте эту статью в моем блоге на Medium:

[How regular expressions can be useful in OSINT. Theory and some practice using Google Sheets](#)

День 12. Прокси

Очень многие сайты и сервисы блокируют IP-адреса, которые отправляют большое количество запросов за короткое время. Обойти такую защиту можно с помощью прокси-серверов (не всегда срабатывает, но иногда срабатывает).

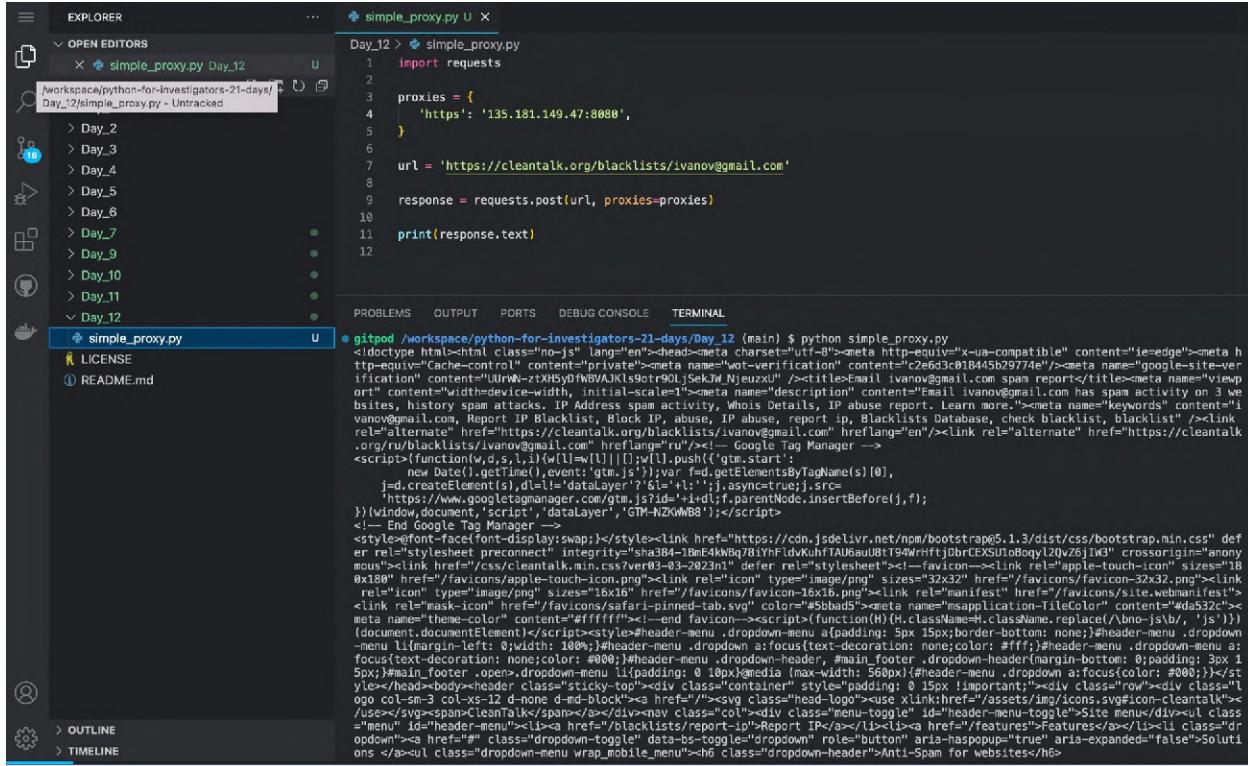


Спасибо за картинку, Upguard.com.

Прокси-сервер, также известный как шлюз уровня приложения, может представлять собой часть программного обеспечения или компьютер. В любом случае он функционирует как шлюз между вашим устройством и сервером, к которому вы подключены. Это похоже на посла, который действует как ваш представитель при совершении транзакций с различными серверами в Интернете (Techslang).

Использовать их в Python очень просто. Вам просто нужно при оформлении запроса указать адрес сервера, через который вы хотите перенаправить трафик.

Запустите simple_proxy.py:



```
simple_proxy.py U
Day_12 > simple_proxy.py
1 import requests
2
3 proxies = {
4     'https': '135.181.149.47:8080',
5 }
6
7 url = 'https://cleantalk.org/blacklists/ivanov@gmail.com'
8
9 response = requests.post(url, proxies=proxies)
10
11 print(response.text)
12
```

```
gitpod /workspace/python-for-investigators-21-days/Day_12 $ python simple_proxy.py
<!DOCTYPE html><html class="no-js" lang="en"><head><meta charset="utf-8"><meta http-equiv="x-ua-compatible" content="ie=edge"><meta http-equiv="Cache-control" content="private"><meta name="wot-verification" content="c2e6d3c01845b29774e"><meta name="google-site-verification" content="UUrWw-ztxBiydfwBAJkls9ot90LjSekjW_NjeuxlU"/><title>Email ivanov@gmail.com spam report</title><meta name="viewport" content="width=device-width, initial-scale=1"><meta name="Email ivanov@gmail.com has spam activity on 3 websites, history spam attacks. IP Address, Whois Details, IP abuse report. Learn more."><meta name="keywords" content="ivanov@gmail.com, Report IP Blacklist, Block IP, abuse, IP abuse, report_ip, Blacklists Database, check blacklist, blacklist" /><link rel="alternate" href="https://cleantalk.org/blacklists/ivanov@gmail.com" hreflang="en"/><link rel="alternate" href="https://cleantalk.org/blacklists/ivanov@gmail.com" hreflang="ru"/><link rel="alternate" href="https://ru.cleantalk.org/blacklists/ivanov@gmail.com" hreflang="ru"/><link rel="tag manager" href="https://www.googletagmanager.com/gtm.js?id=GTM-NZKWWB8"/></script>
<!-- End Google Tag Manager -->
<style>@font-face{font-display:swap!}</style><link href="https://con.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" defer rel="stylesheet preconnect" integrity="sha384-1BmE4kkBq781YHfdKuhTAU6au8t94WhrffjbrCEXSU108qy120vZ6jW0" crossorigin="anonymous"><link href="/css/cleantalk.min.css?ver03-03-2023n" defer rel="stylesheet"><!-- favicon --><link rel="apple-touch-icon" sizes="180x180" href="/favicons/apple-touch-icon.png"><link rel="icon" type="image/png" sizes="32x32" href="/favicons/favicon-32x32.png"><link rel="icon" type="image/png" sizes="16x16" href="/favicons/favicon-16x16.png"><link rel="manifest" href="/favicons/site.webmanifest"><link rel="mask-icon" href="/favicons/safari-pinned-tab.svg" color="#5bbad5"><meta name="msapplication-TileColor" content="#da532c"><meta name="theme-color" content="#fffff"><!-- favicon --><script>(function(H){H.className=H.className.replace(/\bno-js\b/, 'js'))(document.documentElement)</script><style>#header-menu .dropdown-menu a{padding: 5px 15px; border-bottom: none;}<!-- header-menu .dropdown-menu a:focus{text-decoration: none; color: #000;}<!-- header-menu .dropdown-header, #main_footer .dropdown-header{margin-bottom: 0; padding: 3px 15px;}<!-- main_footer .open .dropdown-menu li{padding: 0 10px;}media (max-width: 560px){#header-menu .dropdown a:focus{color: #000;}}</style></head><body><header><div><div><div><div><div><a href="#"><img alt="cleantalk logo" href="/assets/img/icons.svg#icon-cleantalk"></a></div></div></div></div></div><div><ul><li><a href="#">Report IP</a></li><li><a href="#">Features</a></li><li class="dropdown"><a href="#">Menu</a><ul><li><a href="#">Report IP</a></li><li><a href="#">Features</a></li><li><a href="#">Anti-Spam for Websites</a></li></ul></li></ul></div></header>
```

Пакет запросов на импорт:

```
import requests
```

Создайте переменную с прокси-сервером https и портом:

```
proxies = {
    'https': '135.181.149.47:8080',
}
```

Создайте переменную с URL-адресом для запроса:

```
url = 'https://cleantalk.org/blacklists/ivanov@gmail.com'
```

Делаем запрос через прокси-сервер:

```
response = requests.post(url, proxies=proxies)
```

Распечатать текст веб-страницы:

```
print(response.text)
```

Прокси-сервер, использованный в качестве примера в приведенном выше коде, вероятно, больше не работает. Поэтому, пожалуйста, замените его на другой. Огромное количество бесплатных серверов можно найти в Google за пару секунд.

The screenshot shows the homepage of hidemy.name. At the top, there's a navigation bar with the logo, links for 'What is a VPN?', 'Pricing', 'Download', 'Help', and a 'Buy access' button. Below the navigation is a search form with dropdowns for 'Country' (set to 'All country (81)'), 'Proxy speed' (set to 'ms'), and 'Proxy types' (HTTP is checked). There are also checkboxes for 'HTTPS', 'Socks 4', 'Socks 5', 'High', 'Average', 'Low', and 'no'. A text input field for 'Port number' is present, with a note that it can be separated by commas. Below the search form are buttons for 'Show' (highlighted in blue), 'Paid Features', and 'Export' (with options for 'IP:Port' or 'Excel'). A note at the bottom says 'Don't know how to use a proxy? Check the instructions for your browser' with icons for various browsers. The main content area displays a table of proxy servers with columns for IP address, Port, Country/City, Speed, Type, Anonymity, and Latest update. The first three rows are visible: 172.67.8.142 (United States, 160 ms, HTTP, no, 2 minutes), 191.101.251.42 (Netherlands Rotterdam, 580 ms, HTTP, no, 2 minutes), and 45.8.105.111 (Curacao, 600 ms, HTTP, no, 2 minutes).

IP address	Port	Country, City	Speed	Type	Anonymity	Latest update
172.67.8.142	80	United States	160 ms	HTTP	no	2 minutes
191.101.251.42	80	Netherlands Rotterdam	580 ms	HTTP	no	2 minutes
45.8.105.111	80	Curacao	600 ms	HTTP	no	2 minutes

Примеры списка прокси-серверов:

<https://hidemy.name/en/>

<https://github.com/clarktm/proxy-list>

<https://github.com/TheSpeedX/PROXY-List>

<https://github.com/ShiftyTR/Proxy-List>

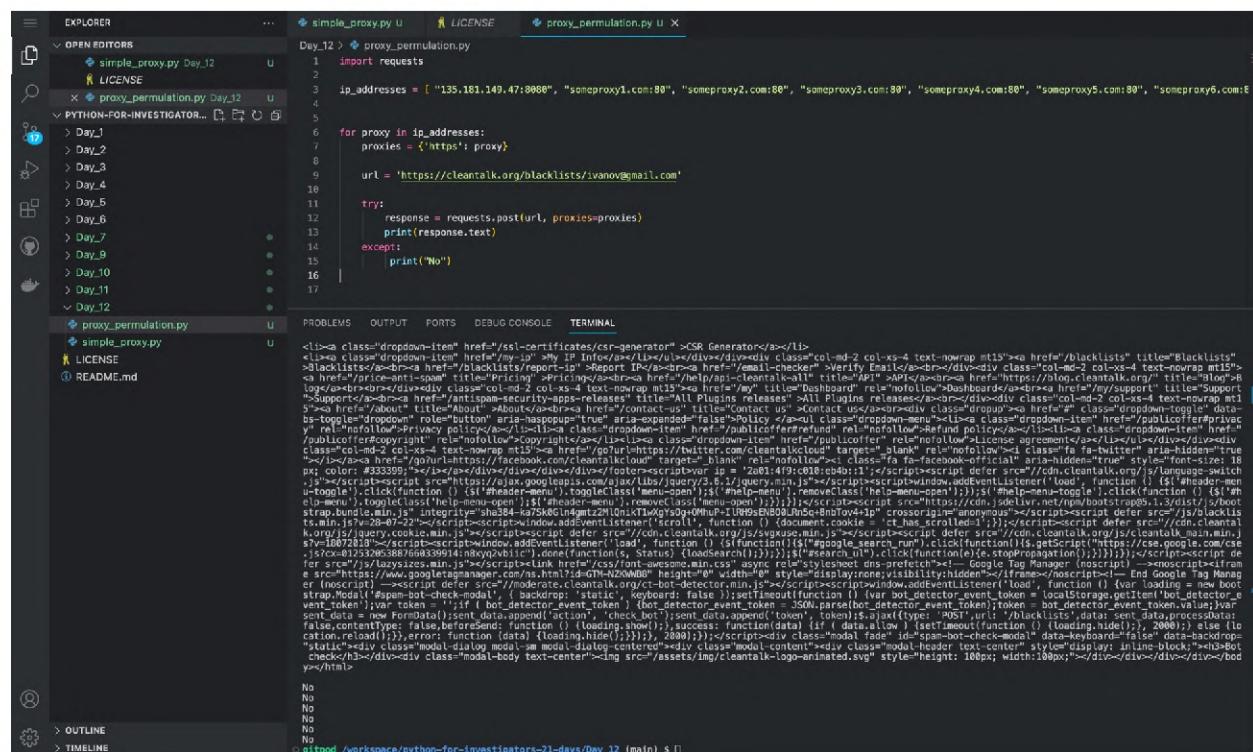
<https://github.com/jetkai/proxy-list>

Одного прокси-сервера недостаточно для успешного обхода защиты от парсинга. Ведь целевой сайт может блокировать их по одному, а кроме того, бесплатные прокси-серверы могут работать крайне нестабильно.

Поэтому вам может потребоваться поиск по прокси-серверам, чтобы найти работающий и НЕ заблокированный.

Запустите proxy_permutation.py:

Как и в первом случае, прокси-адреса из списка на момент публикации книги могут не работать. Поэтому замените их на другие (которые, как сказано выше, можно найти в свободных списках) перед запуском скрипта.



The screenshot shows a code editor interface with several files listed in the sidebar: simple_proxy.py, LICENSE, Day_1, Day_2, Day_3, Day_4, Day_5, Day_6, Day_7, Day_8, Day_9, Day_10, Day_11, Day_12, and proxy_permutation.py. The proxy_permutation.py file is currently selected and its content is displayed in the main editor area. The code is a Python script that iterates through a list of IP addresses, sends a POST request to a specific URL using each proxy, and prints the response text if successful or an error message if it fails. The code also includes imports for requests and os, and handles exceptions.

```
simple_proxy.py
LICENSE
proxy_permutation.py
Day_1
Day_2
Day_3
Day_4
Day_5
Day_6
Day_7
Day_8
Day_9
Day_10
Day_11
Day_12
proxy_permutation.py
LICENSE
README.md
```

```
import requests
from bs4 import BeautifulSoup
import os
ip_addresses = ['135.181.149.47:8088', 'someproxy1.com:80', 'someproxy2.com:80', 'someproxy3.com:80', 'someproxy4.com:80', 'someproxy5.com:80', 'someproxy6.com:80']

for proxy in ip_addresses:
    proxies = {'https': proxy}
    url = 'https://cleantalk.org/blacklists/ivanov@gmail.com'
    try:
        response = requests.post(url, proxies=proxies)
        print(response.text)
    except:
        print('No')
```

```
# Пакет запросов на импорт:
```

```
import requests
```

```
# Создайте список с https-прокси-серверами и портами:
```

```
ip_addresses = [ "135.181.149.47:8080", "someproxy1.com:80",
"someproxy2.com:80", "someproxy3.com:80", "someproxy4.com:80",
"someproxy5.com:80", "someproxy6.com:80"]
```

```
# Создайте переменную с URL-адресом для запроса:
```

```
url = 'https://cleantalk.org/blacklists/ivanov@gmail.com'
```

```
# Просматриваем список ip_addresses:
```

```
for proxy in ip_addresses:
    proxies = {'https': proxy}
```

```
# Попробуйте сделать запрос и распечатать результаты:
```

```
try:
    response = requests.post(url, proxies=proxies)
    print(response.text)
except:
    print("No")
```

Вы также можете использовать готовые инструменты для перенаправления трафика через прокси-серверы:

[XX-net](#)

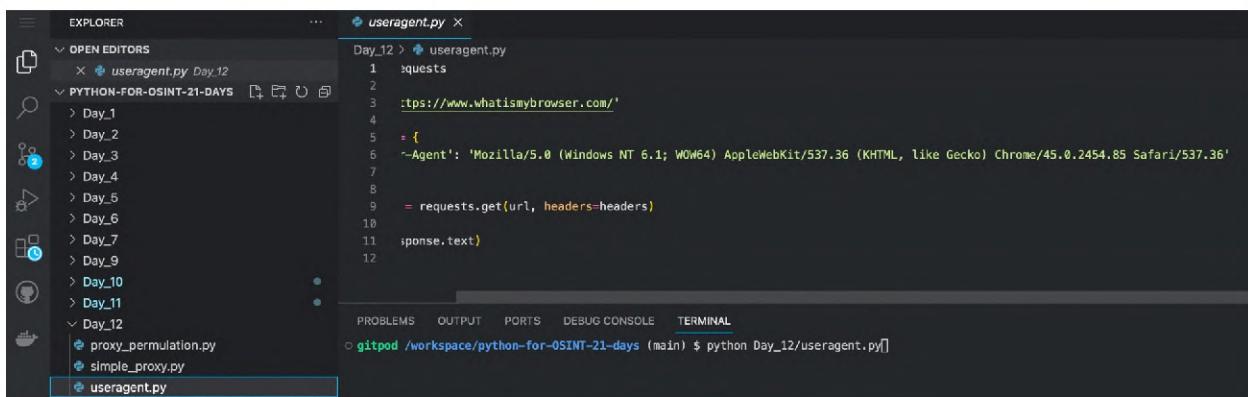
[mitmproxy](#)

[Proxify](#) (очень хороший инструмент, написанный на Go от Projectdiscovery)

Зачастую простой смены IP-адреса недостаточно для обхода заблокированных сайтов. Вам также придется добавить дополнительные параметры к запросу в заголовках запроса. Настройки будут индивидуальны для каждой задачи.

Вот простой пример запроса с заголовком User-Agent. (информация об устройстве пользователя) добавлена.

Запустите useragent.py:



```
#!/usr/bin/python3
# Day_12

import requests
url = 'https://www.whatismybrowser.com/'

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36'
}

response = requests.get(url, headers=headers)
print(response.text)
```

Пакет запроса на импорт:

```
import requests
```

Создайте переменную со ссылкой на сайт:

```
url = 'https://www.whatismybrowser.com/'
```

Создаем список с заголовками запросов (сейчас мы используем только заголовок User-Agent):

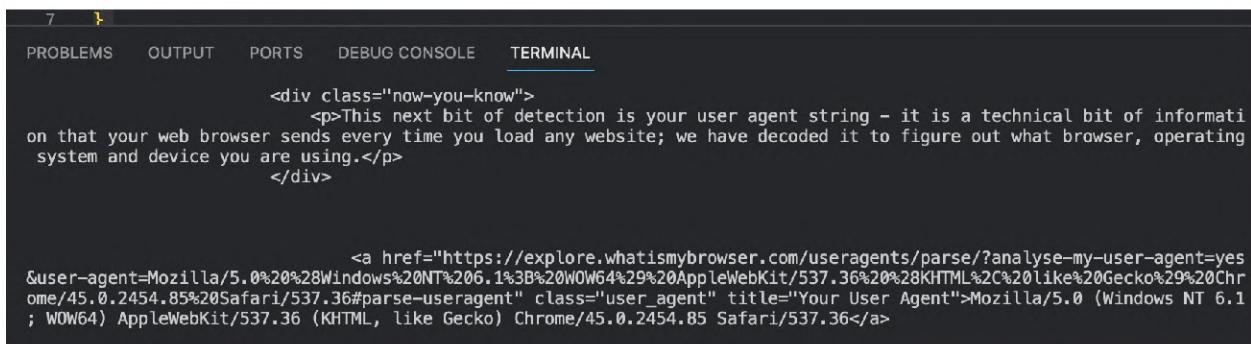
```
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36'
}
```

```
# Послать запрос:
```

```
response = requests.get(url, headers=headers)
```

```
# Распечатать текст ответа:
```

```
print(response.text)
```



```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

<div class="now-you-know">
    <p>This next bit of detection is your user agent string - it is a technical bit of information that your web browser sends every time you load any website; we have decoded it to figure out what browser, operating system and device you are using.</p>
</div>

<a href="https://explore.whatismybrowser.com/useragents/parse/?analyse-my-user-agent=yes&user-agent=Mozilla/5.0%20%28Windows%20NT%206.1%3B%20WOW64%29%20AppleWebKit/537.36%20%28KHTML%20like%20Gecko%29%20Chrome/45.0.2454.85%20Safari/537.36#parse-useragent" class="user_agent" title="Your User Agent">Mozilla/5.0 (Windows NT 6.1 ; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36</a>
```

В результате должен отобразиться html-код страницы, который будет содержать User-Agent, указанный в заголовках, передаваемых вместе с запросом.

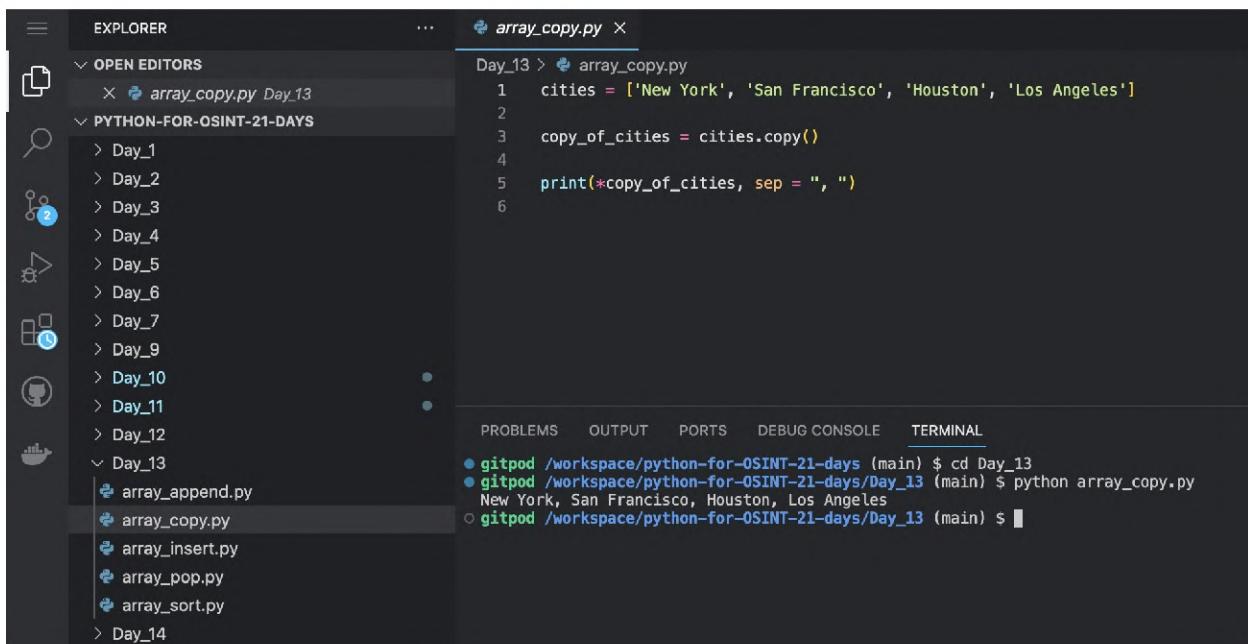
День 13. Функции для работы со списками

Как вы уже заметили, списки — это очень важный элемент синтаксиса Python, который мы использовали в большинстве наших уроков.

Сегодняшний урок будет чем-то вроде дня отдыха. Мы рассмотрим несколько очень простых, но очень полезных функций для работы со списками.

Для начала давайте научимся копировать массивы.

Запустите array_copy.py:



```
array_copy.py
1 cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
2
3 copy_of_cities = cities.copy()
4
5 print(*copy_of_cities, sep = ", ")
```

Создать список городов США:

```
cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
```

Копировать список городов США:

```
copy_of_cities = cities.copy()
```

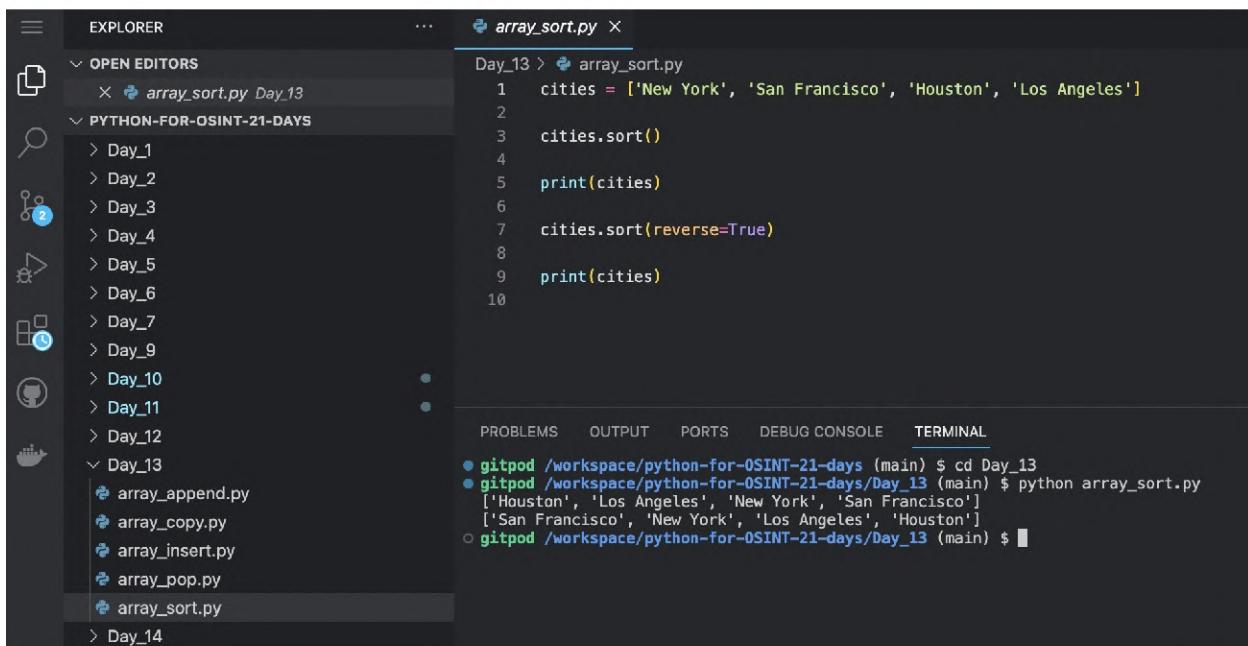
Распечатайте копию списка городов США с элементами, разделенными знаком «,»:

```
print(*copy_of_cities, sep = ", ")
```

Обратите внимание, что мы использовали новый способ вывода элементов массива (мы установили разделитель).

Теперь давайте отсортируем список.

Запустите array_sort.py:



```
array_sort.py
Day_13 > array_sort.py
1   cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
2
3   cities.sort()
4
5   print(cities)
6
7   cities.sort(reverse=True)
8
9   print(cities)
10
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

- gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_13
- gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) \$ python array_sort.py
- ['Houston', 'Los Angeles', 'New York', 'San Francisco']
- ['San Francisco', 'New York', 'Los Angeles', 'Houston']
- gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) \$

Создать список городов США:

```
cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
```

Сортируем список городов США по возрастанию:

```
cities.sort()
```

Результат печати:

```
print(cities)
```

Сортируем список городов США по убыванию:

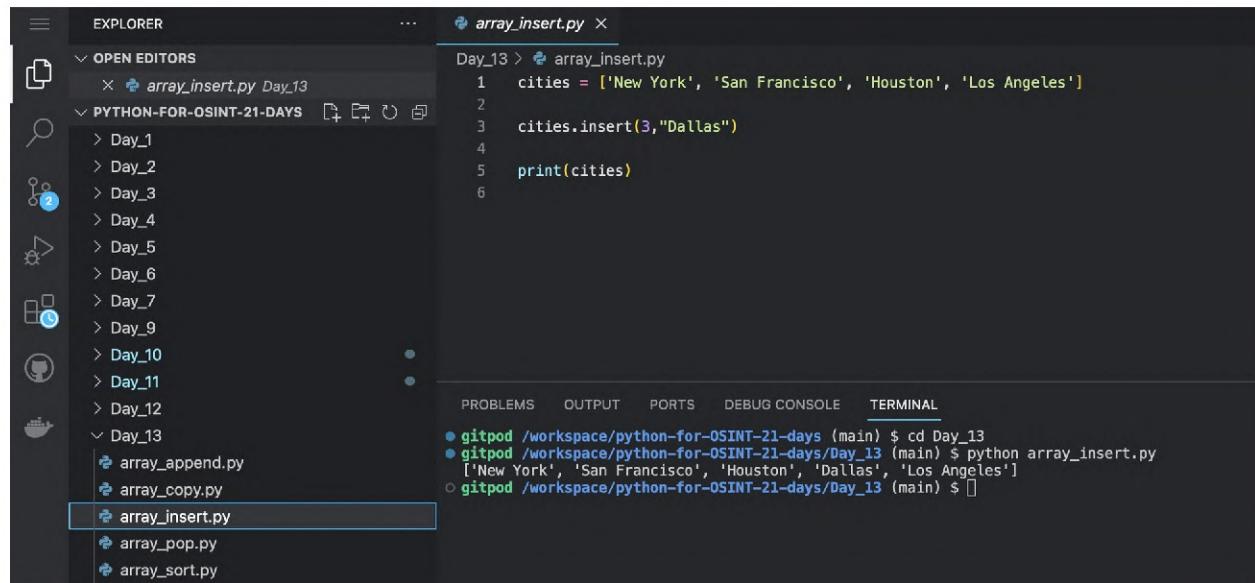
```
cities.sort(reverse=True)
```

Распечатать результаты:

```
print(cities)
```

А вот две функции для добавления элементов в массив — Insert() и Append().

Запустите array_insert.py:



```
Day_13 > array_insert.py
1  cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
2
3  cities.insert(3,"Dallas")
4
5  print(cities)
6
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_13
gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) $ python array_insert.py
['New York', 'San Francisco', 'Houston', 'Dallas', 'Los Angeles']
gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) $
```

Создайте список названий городов США:

```
cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
```

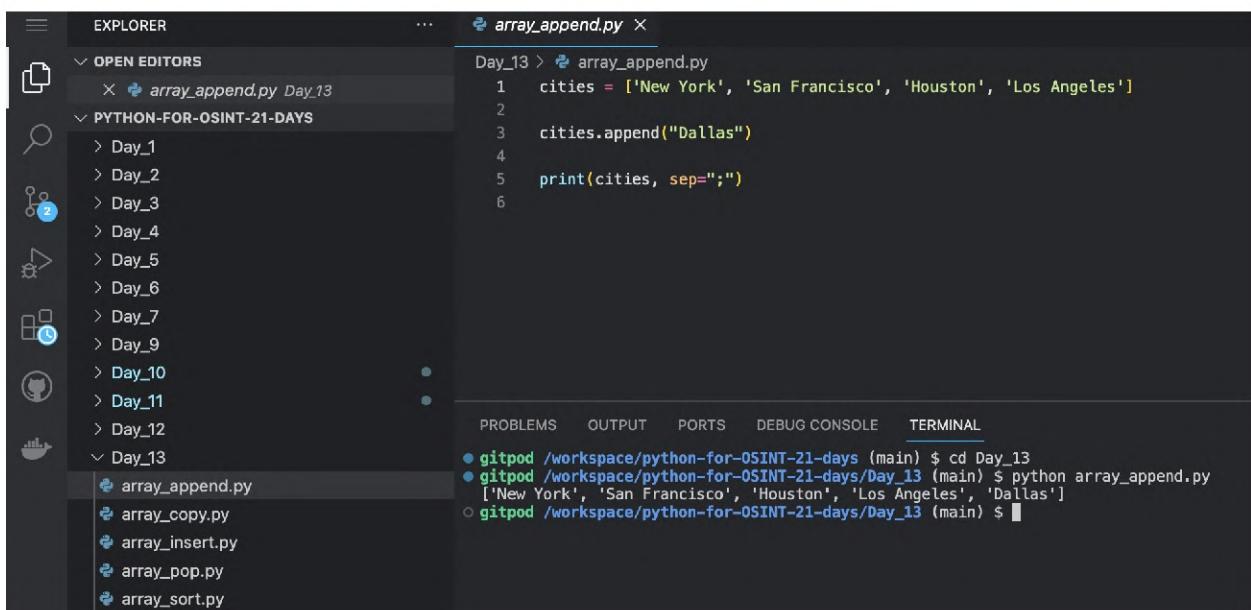
Добавьте элемент номер три и текст Dallas (помните, что счет начинается с нуля):

```
cities.insert(3,"Dallas")
```

Выводим измененный список на экран:

```
print(cities)
```

Запустите array_append.py:



```
array_append.py
Day_13 > array_append.py
1  cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
2
3  cities.append("Dallas")
4
5  print(cities, sep=";")
6
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_13
gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) $ python array_append.py
['New York', 'San Francisco', 'Houston', 'Los Angeles', 'Dallas']
gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) $
```

Создайте список названий городов США:

```
cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
```

Добавьте Даллас в конец списка:

```
cities.append("Dallas")
```

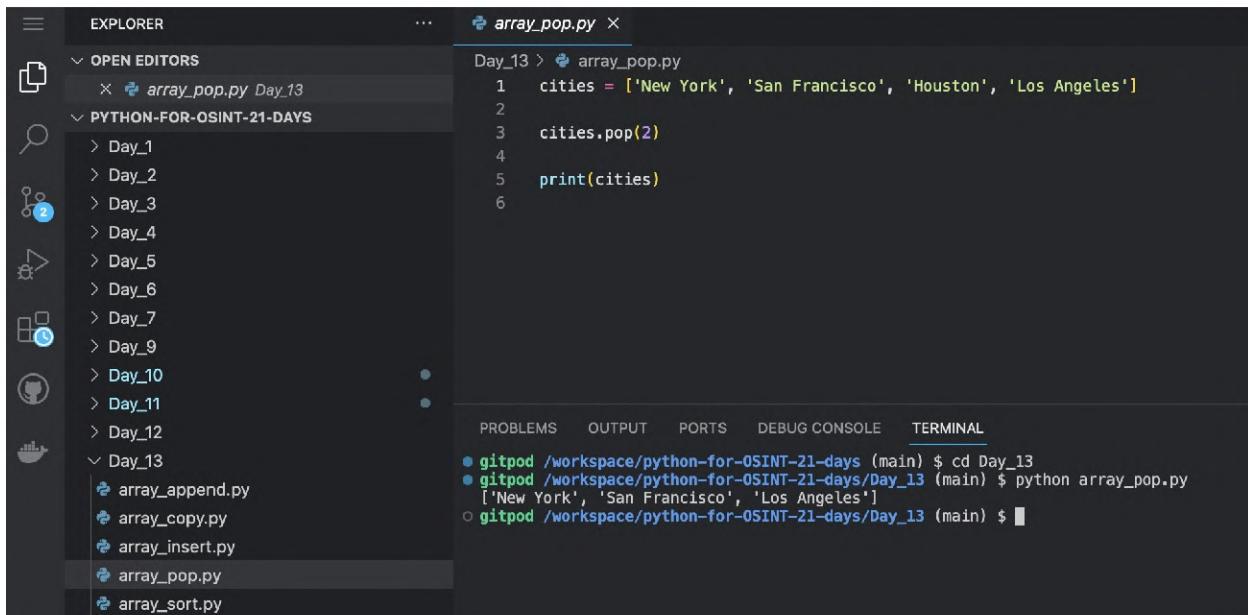
Выводим список на экран, разделяя элементы точкой с запятой:

```
print(cities, sep=";")
```

На всякий случай позвольте мне объяснить разницу между `Insert()` и `Append()`. `Insert()` вставляет элемент в определенное место (под определенным номером). А `Append()` добавляет элемент в конец массива (под последним номером).

Закончим этот урок функцией, удаляющей из массива элемент с определенным номером.

Запустите `array_pop.py`:



```
Day_13 > array_pop.py
1  cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
2
3  cities.pop(2)
4
5  print(cities)
6
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_13
gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) $ python array_pop.py
['New York', 'San Francisco', 'Los Angeles']
gitpod /workspace/python-for-OSINT-21-days/Day_13 (main) $
```

Создайте список названий городов США:

```
cities = ['New York', 'San Francisco', 'Houston', 'Los Angeles']
```

Удаляем из него элемент с индексом два (помним, что отсчет начинается с нуля):

```
cities.pop(2)
```

Выводим измененный список на экран:

```
print(cities)
```

В этом курсе мы вернемся к теме массивов и обсудим важнейшую функцию для работы с ними — map(), которая определенно достойна отдельного урока.

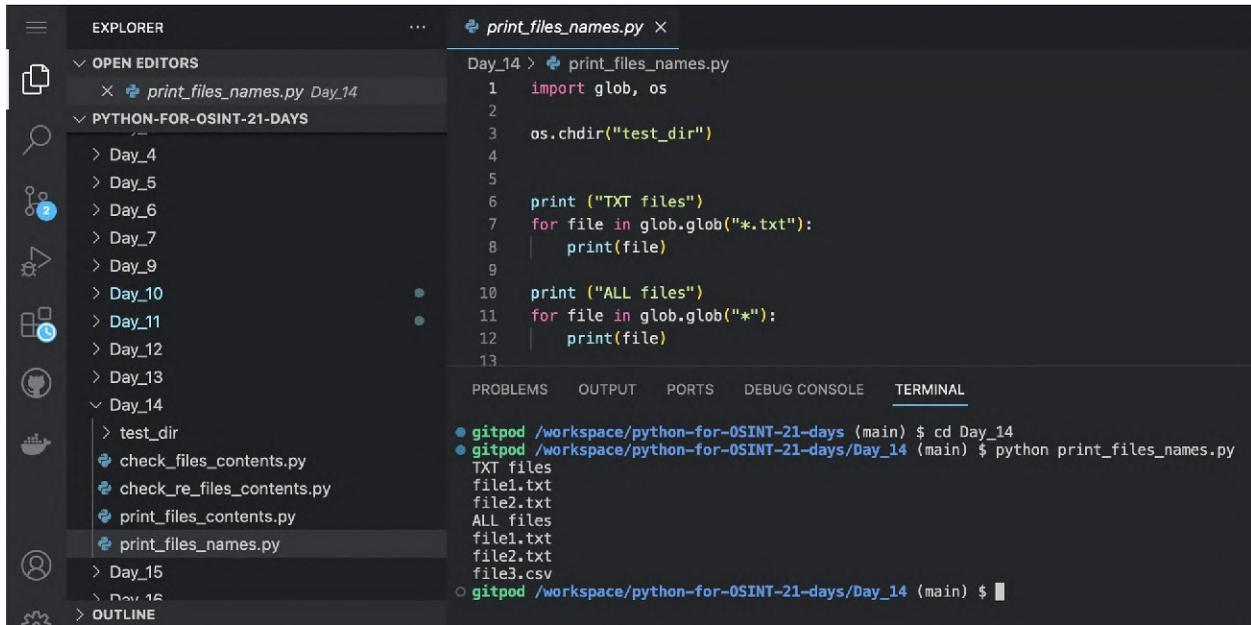
День 14. Работа с файловой системой

В этом курсе мы не имеем дела с базами данных и используем файлы csv, json, txt для хранения данных. А иногда может возникнуть желание написать скрипт, который будет обрабатывать данные не из одного файла с определенным именем, а данные из большой группы файлов (иногда расположенных в разных каталогах).

Поэтому вам могут понадобиться минимальные навыки работы с файловой системой на Python.

Мы будем использовать glob (<https://docs.python.org/3/library/glob.html>) и os. (<https://docs.python.org/3/library/os.html>) для поиска и чтения файлов.

Запустите print_files_names.py:



```
print_files_names.py
1 import glob, os
2
3 os.chdir("test_dir")
4
5
6 print ("TXT files")
7 for file in glob.glob("*.txt"):
8     print(file)
9
10 print ("ALL files")
11 for file in glob.glob("*"):
12     print(file)
13
```

The screenshot shows the VS Code interface with the 'print_files_names.py' file open in the editor. The code uses the `glob` and `os` modules to print file names from a directory named 'test_dir'. The terminal shows the output of running the script, which lists 'file1.txt', 'file2.txt', and 'file3.csv' under 'TXT files', and all three files under 'ALL files'.

Импортируем пакеты glob и os:

```
import glob, os
```

Перейдите в каталог «test_dir»:

```
os.chdir("test_dir")
```

Распечатать строку «TXT-файлы»:

```
print ("TXT files")
```

Найдите все файлы с расширением txt и выведите их имена:

```
for file in glob.glob("*.txt"):
    print(file)
```

Распечатать строку «ВСЕ файлы»:

```
print ("ALL files")
```

Найдите все файлы и выведите их имена:

```
for file in glob.glob("*"):
    print(file)
```

Теперь попробуем отобразить содержимое найденных файлов с помощью команды readlines().

Запустите print_files_contents.py:

```
print_files_contents.py
1 import glob, os
2
3 os.chdir("test_dir")
4
5 for file in glob.glob("*.txt"):
6
7     with open (file) as current_file:
8         print(current_file.readlines())
9
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_14
gitpod /workspace/python-for-OSINT-21-days/Day_14 (main) $ python print_files_contents.py
['Hello from file1.txt!\n']
['Hello from file2.txt!\n']
gitpod /workspace/python-for-OSINT-21-days/Day_14 (main) $
```

Импортируем пакеты glob и os:

```
import glob, os
```

Перейдите в каталог «test_dir»:

```
os.chdir("test_dir")
```

Найдите все файлы с расширением txt и выведите их имена:

```
for file in glob.glob("*.txt"):
```

Откройте каждый файл и распечатайте его содержимое:

```
with open (file) as current_file:
    print(current_file.readlines())
```

Содержимое полученных таким образом файлов можно автоматически анализировать.

Самый простой пример — проверка наличия какого-либо символа или слова.

Запустите check_files_contents.py:

```
Day_14 > check_files_contents.py
1 import glob, os
2
3 os.chdir("test_dir")
4
5 for file in glob.glob("*"):
6
7     with open (file) as current_file:
8         current_file_content=current_file.read()
9         if "2" in current_file_content:
10             print(current_file_content)
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_14
gitpod /workspace/python-for-OSINT-21-days/Day_14 (main) $ python check_files_contents.py
Hello from file2.txt!
gitpod /workspace/python-for-OSINT-21-days/Day_14 (main) $ 
```

Импортируем пакеты glob и os:

```
import glob, os
```

Перейдите в каталог «test_dir»:

```
os.chdir("test_dir")
```

#Проходим по всем файлам

```
for file in glob.glob("*"):
```

Откройте каждый файл:

```
    with open (file) as current_file:
```

Читаем содержимое каждого файла:

```
        current_file_content=current_file.read()
```

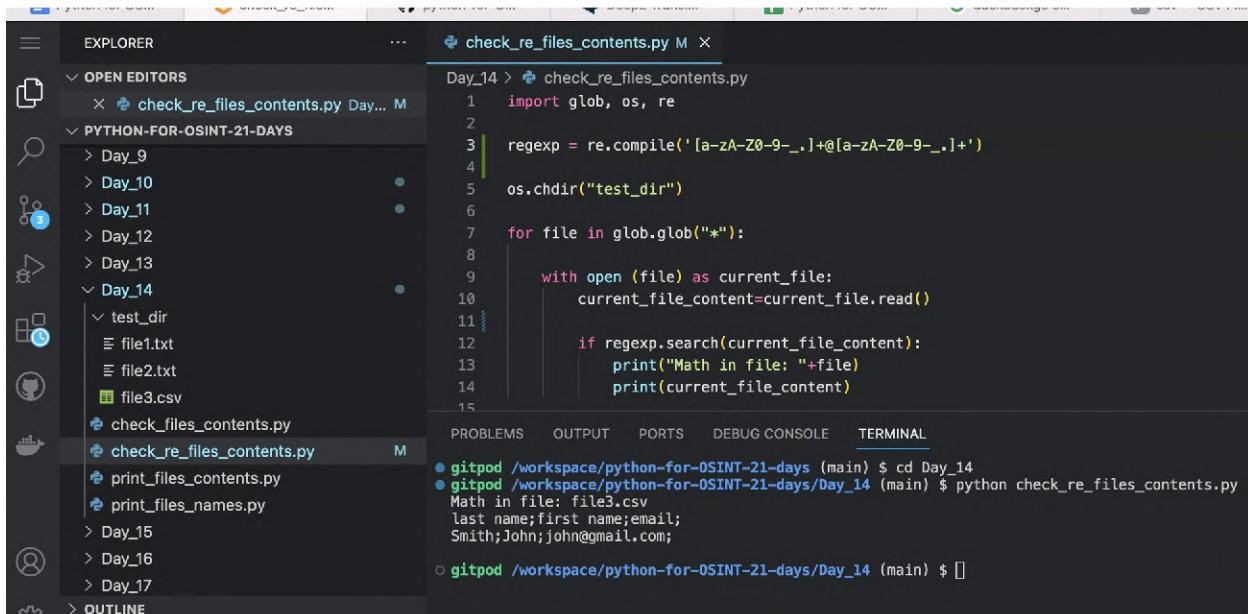
Если содержимое файла содержит цифру «2», выведите ее:

```
        if "2" in current_file_content:
```

```
print(current_file_content)
```

Теперь попробуем проверить каждый файл на наличие адреса электронной почты в его тексте с помощью регулярного выражения (мы изучили его на уроке 11).

Запустите check_re_file_contents.py:



The screenshot shows the Visual Studio Code interface. The left sidebar (EXPLORER) lists several Python files and a directory structure for 'Day_14'. The main area (CODEVIEW) displays the code for `check_re_file_contents.py`. The terminal at the bottom shows the command being run and its output, which includes a match found in `file3.csv`.

```
Day_14 > check_re_file_contents.py
1 import glob, os, re
2
3 regexp = re.compile('[a-zA-Z0-9-_]+@[a-zA-Z0-9-__.]+')
4
5 os.chdir("test_dir")
6
7 for file in glob.glob("*"):
8
9     with open(file) as current_file:
10         current_file_content=current_file.read()
11
12         if regexp.search(current_file_content):
13             print("Match in file: "+file)
14             print(current_file_content)
15
```

TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_14
gitpod /workspace/python-for-OSINT-21-days/Day_14 (main) $ python check_re_file_contents.py
Match in file: file3.csv
last name;first name;email;
Smith;John;john@gmail.com;
```

Импортируйте пакеты glob, os и re:

```
import glob, os, re
```

Создайте объект регулярного выражения:

```
regexp = re.compile('[a-zA-Z0-9-_]+@[a-zA-Z0-9-__.]+')
```

Перейдите в каталог «test_dir»:

```
os.chdir("test_dir")
```

#Проходим по всем файлам

```
for file in glob.glob("*"):
```

Откройте каждый файл:

```
    with open (file) as current_file:
```

Прочитать содержимое файла:

```
        current_file_content=current_file.read()
```

Проверяем, можно ли найти в тексте файла строку, соответствующую регулярному выражению:

```
if regexp.search(current_file_content):
```

Если он найден, выводим имя и содержимое файла:

```
    print("Math in file: "+file)
    print(current_file_content)
```

Если вы не знаете, где взять регулярные выражения для других типов данных (номеров телефонов, IP-адресов и т. д.), значит, вы невнимательно читали Урок 11 и не читали статью по ссылке.

День 15. Сбор информации о домене

Этот урок было труднее всего написать, потому что это тема, достойная отдельного курса под названием «Сеть на Python» или «Python для Pentester».

Сегодня я попытаюсь объяснить основные термины, связанные с исследованием предметной области, и показать вам некоторые пакеты Python, которые могут быть полезны для этой цели.

WHOIS — это протокол запросов и ответов, который предоставляет информацию о существующих доменных именах и все соответствующие данные о них ([Techslang](#)).

The screenshot shows the 'who.is' website interface. At the top, there is a search bar with 'nytimes.com' and a magnifying glass icon. To the right of the search bar are links for 'Premium Domains', 'Transfer', and 'Features'. Below the search bar, the page displays the WHOIS contact information for the domain 'nytimes.com'. It is organized into three sections: 'Registrant Contact Information', 'Administrative Contact Information', and 'Technical Contact Information'. Each section lists various fields such as Name, Organization, Address, City, State / Province, Postal Code, Country, Phone, Fax, and Email, with their corresponding values. The registrant and administrative contacts are listed as 'Domain Administrator' from 'The New York Times Company' at '620 8th Avenue, New York NY 10018' with phone numbers '+1.2125561234' and email 'hostmaster@nytimes.com'. The technical contact information is also listed as 'Domain Administrator' from 'The New York Times Company'.

Registrant Contact Information:	
Name	Domain Administrator
Organization	The New York Times Company
Address	620 8th Avenue,
City	New York
State / Province	NY
Postal Code	10018
Country	US
Phone	+1.2125561234
Fax	+1.2125561234
Email	hostmaster@nytimes.com

Administrative Contact Information:	
Name	Domain Administrator
Organization	The New York Times Company
Address	620 8th Avenue,
City	New York
State / Province	NY
Postal Code	10018
Country	US
Phone	+1.2125561234
Fax	+1.2125561234
Email	hostmaster@nytimes.com

Technical Contact Information:	
Name	Domain Administrator
Organization	The New York Times Company

Существует множество бесплатных онлайн-сервисов, которые отображают данные Whois для определенного домена. Например:

<https://who.is/whois/nytimes.com>

Там вы можете узнать дату регистрации домена, дату окончания оплаченного периода использования домена, контактную информацию компании или человека, который на данный момент владеет доменом.

Существуют также сервисы, которые позволяют находить домены, связанные с определенным адресом электронной почты (например, <https://www.whoxy.com/>), и просматривать историю данных whois домена (<https://research.domaintools.com/research /whois-история/>).

Для Python было создано множество пакетов для автоматизации обработки данных WHOIS. Давайте попробуем пакет Python-Whois (<https://pypi.org/project/python-whois/>).

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install python-whois
Collecting python-whois
  Downloading python-whois-0.8.0.tar.gz (109 kB)
    ┌─────────────────────────────────────────────────────────────────────────────────┐
    │          109.6/109.6 kB 4.0 MB/s eta 0:00:00
    └─────────────────────────────────────────────────────────────────────────┘
  Preparing metadata (setup.py) ... done
Collecting future
  Downloading future-0.18.3.tar.gz (840 kB)
    ┌─────────────────────────────────────────────────────────────────────────┐
    │          840.9/840.9 kB 12.1 MB/s eta 0:00:00
    └─────────────────────────────────────────────────────────────────┘
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: python-whois, future
  Building wheel for python-whois (setup.py) ... done
    Created wheel for python-whois: filename=python_whois-0.8.0-py3-none-any.whl size=103247 sha256=d2f3d8b0722183149ad48a0a35b57122da35686aac924c9a78635a05a6021a5e
    Stored in directory: /workspace/.pyenv_mirror/pip_cache/wheels/8a/d4/1d/bab4b44ad52eadf1b10c5c1ec7cb18a936f24b58bfb95b427e
      Building wheel for future (setup.py) ... done
    Created wheel for future: filename=future-0.18.3-py3-none-any.whl size=492022 sha256=938ca2bd301dabaf994397a06137f50bed71d80e101a1728862e82ddbf6bdd4
      Stored in directory: /workspace/.pyenv_mirror/pip_cache/wheels/da/19/ca/9d8c44cd311a955509d7e13da3f0bea42400c469ef825b580b
Successfully built python-whois future
Installing collected packages: future, python-whois
Successfully installed future-0.18.3 python-whois-0.8.0
```

pip install python-whois

Запустите `whois_info.py`:

```
Day_15 > whois_info.py
1 import whois
2
3 whois_info = whois.whois('sector035.nl')
4
5 print (whois_info)
6
7 print("Creation date")
8
9 print(whois_info["creation_date"])
10

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_15
● gitpod /workspace/python-for-OSINT-21-days/Day_15 (main) $ python whois_info.py
{
  "domain_name": "sector035.nl",
  "expiration_date": null,
  "updated_date": "2017-12-28 00:00:00",
  "creation_date": "2017-05-29 00:00:00",
  "status": "active",
  "registrar": "Easyhosting B.V.",
  "registrar_address": "Kabelweg 21",
  "registrar_postal_code": "1014BA",
  "registrar_city": "Amsterdam",
  "registrar_country": "Netherlands",
  "dnssec": "no",
  "name_servers": [
    "becky.ns.cloudflare.com",
    "jonah.ns.cloudflare.com"
  ]
}
Creation date
2017-05-29 00:00:00
● gitpod /workspace/python-for-OSINT-21-days/Day_15 (main) $
```

Импортируем пакет python-whois:

```
import whois
```

Создайте переменную с именем целевого домена:

```
whois_info = whois.whois('sector035.nl')
```

Распечатать всю информацию whois этого домена:

```
print (whois_info)
```

Вывести строку «Дата создания»::

```
print("Creation date")
```

Распечатать дату создания целевого домена:

```
print(whois_info["creation_date"])
```

DNS (система доменных имен) — это система именования компьютеров в Интернете. Это база данных, которая сопоставляет каждый числовой интернет-адрес (называемый IP-адресом) соответствующему доменному имени. ([Техсланг](#))

The screenshot shows the SuperTool Beta7 interface. At the top, there is a search bar with 'nytimes.com' and a 'DNS Lookup' button. Below the search bar, the domain 'a:nytimes.com' is selected. A green button labeled 'Find Problems' is visible. The main content area displays a table of DNS records:

Type	Domain Name	IP Address	TTL
A	nytimes.com	151.101.1.164 Fastly, Inc. (AS64113)	120 sec
A	nytimes.com	151.101.65.164 Fastly, Inc. (AS64113)	120 sec
A	nytimes.com	151.101.128.164 Fastly, Inc. (AS64113)	120 sec
A	nytimes.com	151.101.193.164 Fastly, Inc. (AS64113)	120 sec

Below the table, there is a 'Test' section with a table:

Test	Result
<input checked="" type="checkbox"/> DNS Record Published	DNS Record found

At the bottom, there is a message: 'Your DNS hosting provider is "NS1" Need Bulk Dns Provider Data?' and a footer with links: dns check, mx lookup, dmarc lookup, spf lookup, dns propagation, and Transcript. It also shows the report was generated by dns2.p06.nsone.net on 5/2/2023 at 1:28:09 AM (UTC -5), just for you.

Посмотреть, как выглядят DNS-данные домена, можно с помощью одного из бесплатных онлайн-сервисов (например, <https://mxtoolbox.com/>).

Вы можете использовать пакет DNSPython для автоматизации получения данных DNS (<https://pypi.org/project/dnspython/>).

```
● gitpod /workspace/python-for-OSINT-21-days/Day_15 (main) $ pip install dnspython
Collecting dnspython
  Downloading dnspython-2.3.0-py3-none-any.whl (283 kB)
                                           283.7/283.7 kB 6.7 MB/s eta 0:00:00
Installing collected packages: dnspython
Successfully installed dnspython-2.3.0
```

`pip install dnspython`

Запустите `python dns_info.py`:

```
dns_info.py
```

```
Day_15 > dns_info.py
1 import dns
2 import dns.resolver
3
4 result = dns.resolver.resolve('osintme.com', 'A')
5 for ipval in result:
6     print('IP', ipval.to_text())
7
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

- gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_15
- gitpod /workspace/python-for-OSINT-21-days/Day_15 (main) \$ python dns_info.py
- ○ gitpod /workspace/python-for-OSINT-21-days/Day_15 (main) \$ IP 2.57.138.1

Импортируем пакет pythondns и dns.resolver:

```
import dns
```

```
import dns.resolver
```

Получить записи A для доменов osintme.com:

```
result = dns.resolver.resolve('osintme.com', 'A')
```

Просмотрите результаты и распечатайте каждый из них:

```
for ipval in result:
```

```
    print('IP', ipval.to_text())
```

При сборе данных о сайте может оказаться полезным найти его поддомены, чтобы использовать их как дополнительный источник информации.

Давайте посмотрим, как работает инструмент Discosub (<https://pypi.org/project/dicosub/>).

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install discosub
Collecting discosub
  Downloading discosub-0.3.0-py2.py3-none-any.whl (399 kB)
    ━━━━━━━━━━━━━━━━━━━━ 399.5/399.5 kB 6.6 MB/s eta 0:00:00
Collecting click
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    ━━━━━━━━━━━━━━━━ 96.6/96.6 kB 8.9 MB/s eta 0:00:00
Installing collected packages: click, discosub
Successfully installed click-8.1.3 discosub-0.3.0
```

Установите пакет Discosub из pip:

```
pip install discosub
```

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ discosub run nytimes.com
We have 105 possibilities
Starting at 2023-05-04 14:34:18.667609

testing: 1.nytimes.com
testing: 1rer.nytimes.com
testing: 2tty.nytimes.com
testing: admin.nytimes.com
testing: app.nytimes.com
testing: autocfg.nytimes.com
testing: bbs.nytimes.com
testing: blog.nytimes.com
testing: clients.nytimes.com
testing: code.nytimes.com
testing: cvs.nytimes.com
testing: doc.nytimes.com
testing: email.nytimes.com
testing: f.nytimes.com
testing: feed.nytimes.com
testing: feeds.nytimes.com
testing: 2.nytimes.com
testing: api.nytimes.com
testing: autodiscover.nytimes.com
testing: beta.nytimes.com
testing: cloud.nytimes.com
testing: dev.nytimes.com
testing: files.nytimes.com
testing: 42.nytimes.com
testing: cdn.nytimes.com
testing: customers.nytimes.com
testing: exchange.nytimes.com
testing: finance.nytimes.com
testing: events.nytimes.com
testing: apps.nytimes.com
testing: forum.nytimes.com
testing: forums.nytimes.com
testing: ftp.nytimes.com
testing: git.nytimes.com
```

Запустите Discosub для домена nytimes.com:

```
discosub run nytimes.com
```

The screenshot shows a terminal window with the following content:

```
nytimes_subdomains.txt
1 We have 105 possibilities
2 Starting at 2023-05-04 14:35:14.041917
3
4 testing: 1.nytimes.com
5 testing: 1er.nytimes.com
6 testing: 2.nytimes.com
7 testing: 2tty.nytimes.com
8 testing: 42.nytimes.com
9 testing: admin.nytimes.com
10 testing: api.nytimes.com
11 testing: app.nytimes.com
12 testing: apps.nytimes.com
13 testing: autoconfig.nytimes.com
14 testing: autodiscover.nytimes.com
15 testing: bbs.nytimes.com
16 testing: beta.nytimes.com
17 testing: blog.nytimes.com
18 testing: cdn.nytimes.com
19 testing: clients.nytimes.com
20 testing: cloud.nytimes.com
21 testing: code.nytimes.com
22 testing: customers.nytimes.com
```

Below the terminal window, the file structure of the workspace is visible in the Explorer sidebar:

- OPEN EDITORS:
 - nytimes_subdomains.txt
- PYTHON-FOR-OSINT-21-DAYS:
 - Day_1
 - Day_2
 - Day_3
 - Day_4
 - Day_5
 - Day_6
 - Day_7
 - Day_8
 - Day_9
 - Day_10
 - Day_11
 - Day_12
 - Day_13
 - Day_14
 - Day_15
 - dns_info.py
 - whois_info.py
 - Day_16
 - Day_17
 - Day_18
 - Day_19
 - Day_20
- LICENSE
- nytimes_subdomains.txt
- README.md

Также есть возможность запустить Discosub с сохранением результатов в файле nytimes_subdomains.txt:

```
discosub run nytimes.com >nytimes_subdomains.txt
```

Стоит уточнить, что Discosub хорош своей простотой и поэтому выбран в качестве примера. Но по качеству получаемых результатов он отстает от аналогичных инструментов (находит меньше поддоменов).

Если ваша задача — найти максимальное количество поддоменов для конкретного домена (а лучше всех), то я рекомендую вам использовать несколько инструментов одновременно. Например:

[Sublist3r](#)

[SubDomainizer](#)

[Subscraper](#)

[Subfinder](#) (написан на языке Go, очень хороший инструмент от Projectdiscovery)

и многие другие.

Domain/DNS/IP lookup			Whois			GEO IP		
Name	Link	Description	Name	Link	Description	Name	Link	Description
API OSINT DS	https://github.com/davidozzo/apisintDS	Collect info about IP4/IPv6 URLs and file hashes in md5, sha1 or sha256	Whois Freaks	https://whoisfreaks.com/	well-parsed and structured domain WHOIS data for all domain names, registrars, countries and TLDs since the birth of internet	Ipstack	https://ipstack.com	Detect country, region, city and zip code
InfoDB API	https://www.ipinfodb.com/api	The API returns the location of an IP address (country, region, city, zipcode, latitude, longitude) and the associated timezone in XML, JSON or plain text format	WhoisXML API	https://whois.whoisxmlapi.com	gathers a variety of domain ownership and registration data points from a comprehensive WHOIS database	Ipgeolocation.io	https://ipgeolocation.io	provides country, city, state, province, local currency, latitude and longitude, company detail, ISP lookup, language, zip code, country calling code, time zone, current time, sunset and sunrise time, moonset and moonrise
Domainsdb.info	https://domainsdb.info	Registered Domain Names Search	IPtoWhois	https://www.ip2whois.com/developers-api	Get detailed info about a domain	IPInfoDB	https://ipinfodb.com/api	Free Geolocation tools and APIs for country, region, city and time zone lookup by IP address
BGPView	https://bgpview.docs.apiry.io/#	allowing consumers to view all sort of analytics data about the current state and structure of the internet				IP API	https://ip-api.com/	Free domain/IP geolocation info

Также не забывайте, что существует множество API для сбора информации о домене, на основе которых вы можете создавать свои скрипты, идеально подходящие для целей вашего расследования (подробнее об этом мы говорили в уроках 5 и 6).

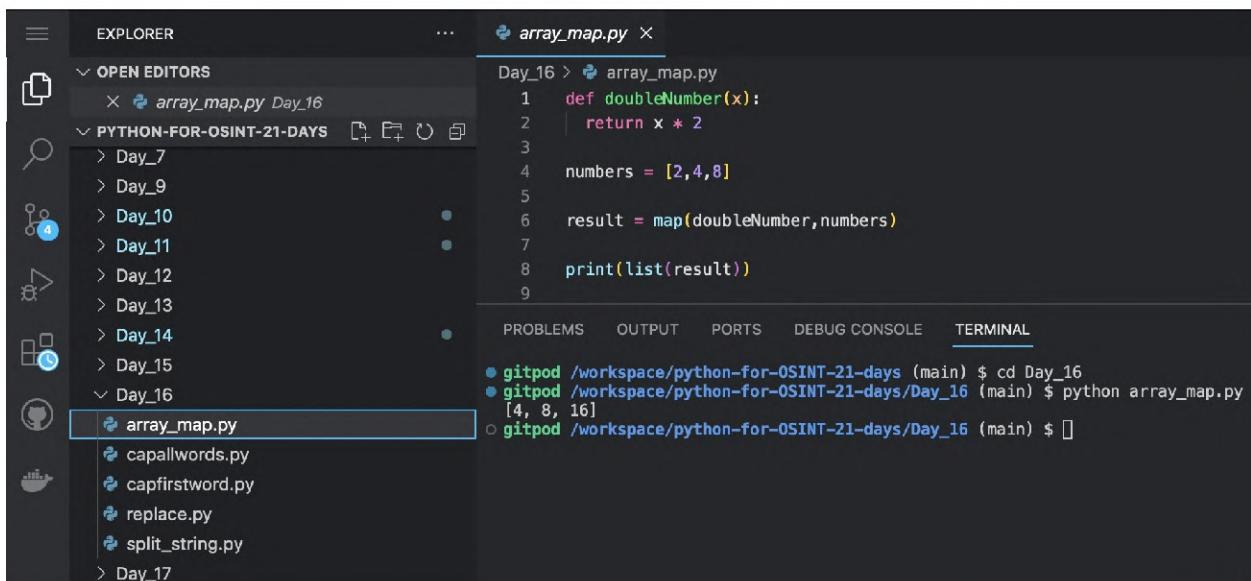
[API-s-for-OSINT \(Domain/DNS/IP lookup\)](#)

День 16. Отображение списка и функции для работы со строками

Map() — это функция, позволяющая применить какую-либо другую функцию к каждому элементу определенного списка. Это открывает перед нами огромные возможности работы с данными.

Например, мы можем произвести какие-то вычисления с каждым элементом списка, заменить или добавить какие-то символы в группе строк, преобразовать одни значения в другие, расшифровать хэши. Что сказать... Вы можете использовать map() практически с любой другой функцией.

Давайте посмотрим, как это работает. Запустите array_map.py:



```
array_map.py
1 def doubleNumber(x):
2     return x * 2
3
4 numbers = [2,4,8]
5
6 result = map(doubleNumber,numbers)
7
8 print(list(result))
```

Создайте функцию, которая умножает переданное ей число на два:

```
def doubleNumber(x):
    return x * 2
```

Создаём список из трёх чисел:

```
numbers = [2,4,8]
```

Примените функцию doubleNumbers по одному к каждому элементу списка чисел:

```
result = map(doubleNumber,numbers)
```

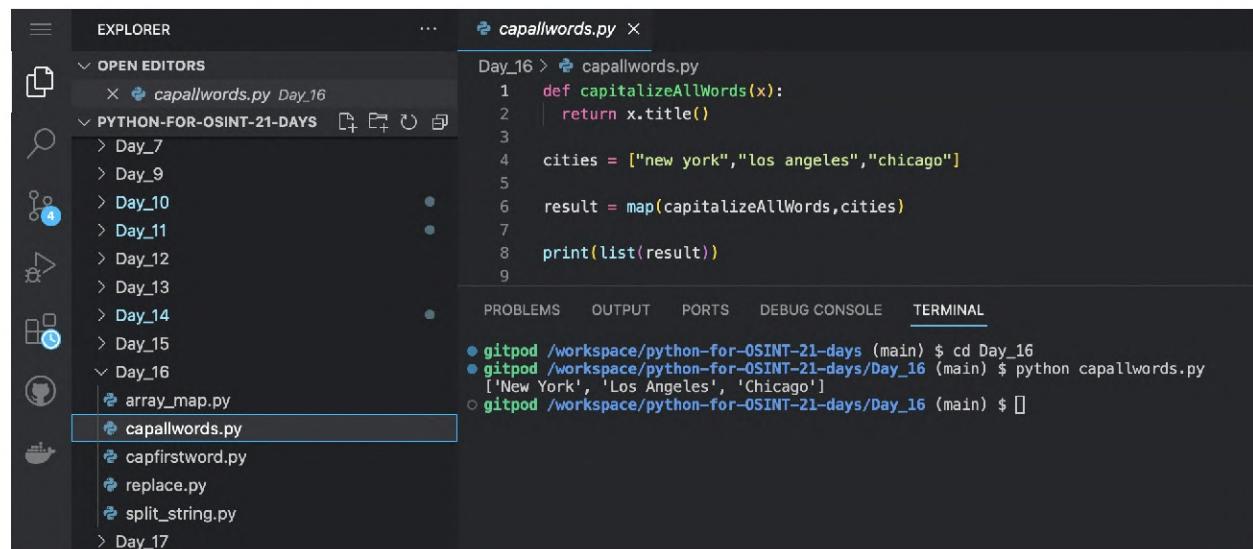
Отображение измененного списка:

```
print(list(result))
```

Поскольку я считаю функцию map() очень важной, в этом уроке будет много примеров ее использования. Я также покажу вам некоторые функции Python для работы со строками.

Давайте попробуем сделать первые буквы слов в каждом элементе списка заглавными.

Запустите capallwords.py:



```
def capitalizeAllWords(x):
    return x.title()

cities = ["new york","los angeles","chicago"]
result = map(capitalizeAllWords,cities)
print(list(result))
```

Создайте функцию, которая будет писать заглавными буквами во всех передаваемых ей словах:

```
def capitalizeAllWords(x):  
    return x.title()
```

Создаем список городов США:

```
cities = ["new york","los angeles","chicago"]
```

Примените функцию CapitalizeAllWords по одному к каждому элементу списка городов США:

```
result = map(capitalizeAllWords,cities)
```

Отображение измененного списка:

```
print(list(result))
```

Теперь давайте будем писать с заглавной только первую букву каждого элемента.

Запустите capfirstword.py:

```
Day_16 > capfirstword.py  
1 def capitalizeFirstWord(x):  
2 | return x.capitalize()  
3  
4 cities = ["new york","los angeles","chicago"]  
5  
6 result = map(capitalizeFirstWord,cities)  
7  
8 print(list(result))
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_16  
bash: cd: Day_16: No such file or directory  
gitpod /workspace/python-for-OSINT-21-days (main) $ python capfirstword.py  
[ 'New York', 'Los Angeles', 'Chicago' ]
```

Создайте функцию CapitalizerFirstWord, которая заменяет первую букву строки на заглавную:

```
def capitalizeFirstWord(x):  
    return x.capitalize()
```

Создайте список городов США:

```
cities = ["new york", "los angeles", "chicago"]
```

Запустите функцию map() для списка из трех городов США и функцию CapitalizeFirstWord:

```
result = map(capitalizeFirstWord, cities)
```

Выводим результат на экран:

```
print(list(result))
```

Еще один очень простой, но очень полезный навык — замена одних символов в строке другими.

Запустите replace.py:

```
Day_16 > replace.py
1 def replaceDash(x):
2     return x.replace("_","-")
3
4 words = ["six_pack","king_size","editor_in_chief"]
5
6 result = map(replaceDash,words)
7
8 print(list(result))
9
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_16
● gitpod /workspace/python-for-OSINT-21-days/Day_16 (main) $ python replace.py
['six-pack', 'king-size', 'editor-in-chief']
○ gitpod /workspace/python-for-OSINT-21-days/Day_16 (main) $
```

Создайте функцию replaceDash, которая заменяет подчеркивание обычным подчеркиванием:

```
def replaceDash(x):
    return x.replace("_","-")
```

Создайте список из трех слов:

```
words = ["six_pack","king_size","editor_in_chief"]
```

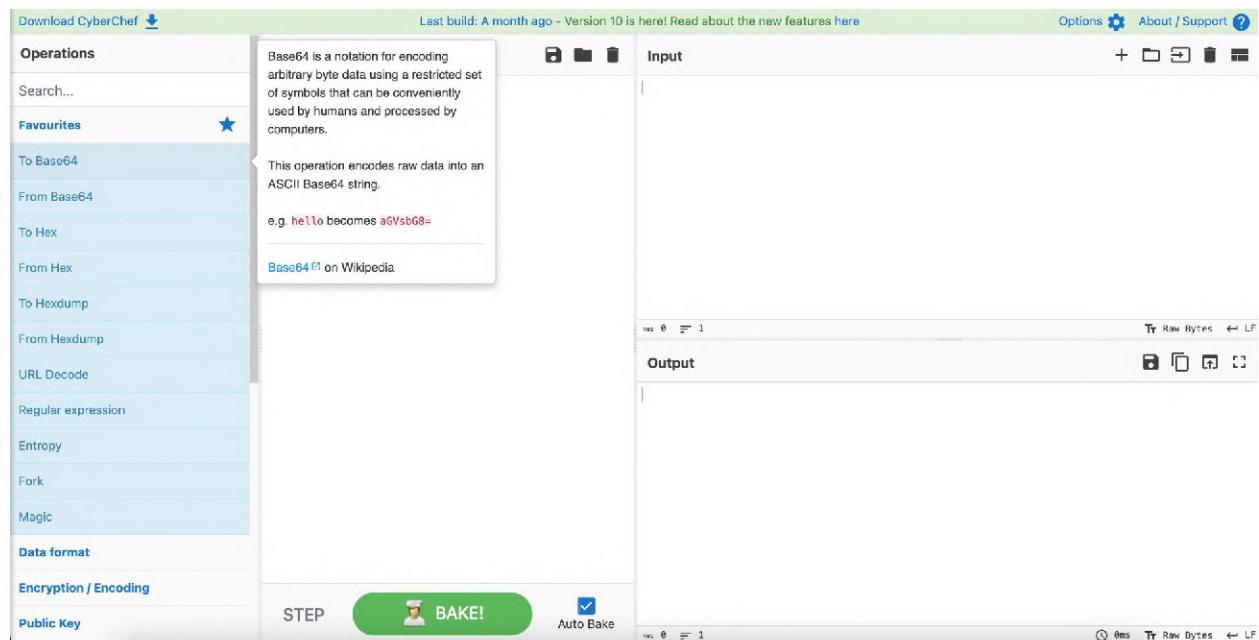
Запускаем функцию map() с аргументами в виде массива из трех слов и функцию replaceDash:

```
result = map(replaceDash,words)
```

Распечатываем результаты на экране:

```
print(list(result))
```

Python имеет множество других функций для работы со строками, и вы также можете установить дополнительные пакеты для расширения этой функциональности.



Например, Python-версия известного онлайн-конвертера данных Cyber Shef (<https://pypi.org/project/cheypy/>). Он может:

- декодировать URL-адреса;
- декодировать/кодировать base64;
- конвертировать двоичные числа в десятичные и наоборот;
- PGP шифрование/дешифрование.

и более.

И в конце этого урока я хочу показать еще одну простую, но очень важную функцию, умеющую разбивать одну строку на несколько по символу-разделителю.

Запустите `split_string.py`:

```
string = "first name;last name@email;address;"  
fields = string.split(";")  
  
for field in fields:  
    print (field)
```

gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_16
gitpod /workspace/python-for-OSINT-21-days/Day_16 (main) \$ python split_string.py
first name
last name
email
address

Создайте строку из нескольких строк, разделенных точкой с запятой:

```
string = "first name;last name@email;address;"
```

Запустите функцию Split(), используя точку с запятой в качестве аргумента:

```
fields = string.split(",")
```

Проходимся по элементам массива в цикле и отображаем каждый по очереди:

```
for field in fields:  
    print (field)
```

День 17. Генерация документов

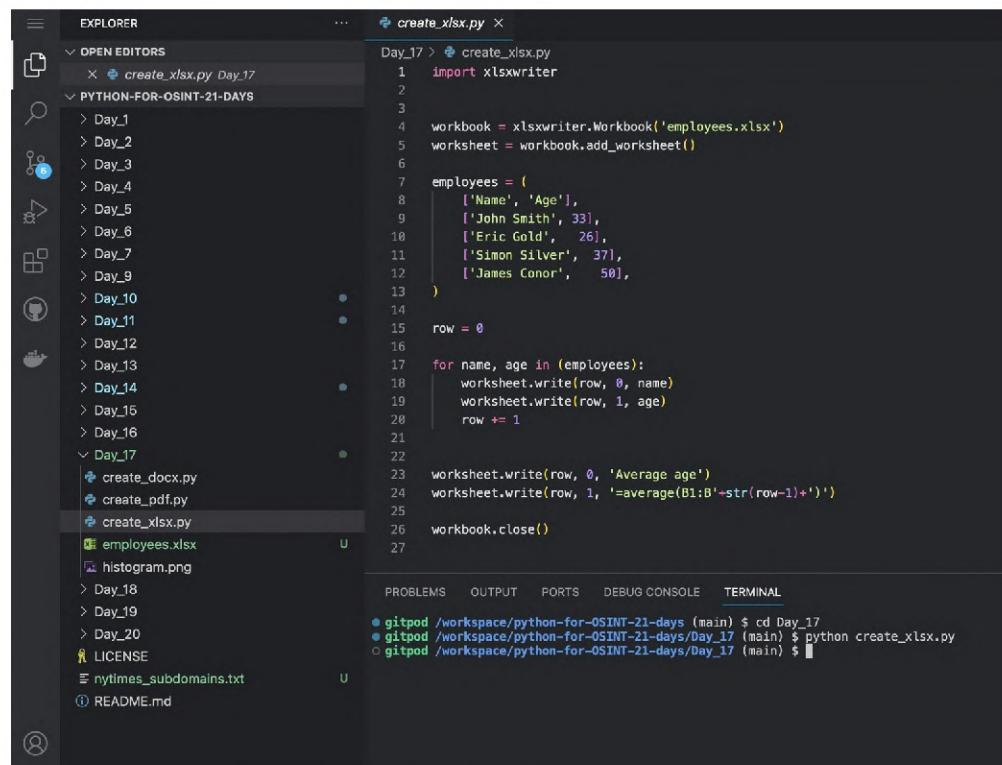
Одной из важных частей большинства исследований или расследований является написание итогового отчета. Если вы создаете много похожих отчетов на основе данных со схожей структурой, вы можете автоматизировать этот процесс с помощью Python. Попробуем создать простую книгу MS Excel.

```
gitpod /workspace/python-for-OSINT-21-days (main) $ pip install XlsxWriter
Collecting XlsxWriter
  Downloading XlsxWriter-3.1.0-py3-none-any.whl (152 kB)
    ━━━━━━━━━━━━━━━━━━━━ 152.7/152.7 kB 4.0 MB/s eta 0:00:00
Installing collected packages: XlsxWriter
Successfully installed XlsxWriter-3.1.0
```

Установите пакет XlsxWriter <https://pypi.org/project/XlsxWriter/>

`pip install XlsxWriter`

Запустите `create_xlsx.py`:



```
EXPLORER          create_xlsx.py ×
Day_17 > create_xlsx.py
OPEN EDITORS
  × create_xlsx.py Day_17
PYTHON-FOR-OSINT-21-DAYS
  > Day_1
  > Day_2
  > Day_3
  > Day_4
  > Day_5
  > Day_6
  > Day_7
  > Day_8
  > Day_9
  > Day_10
  > Day_11
  > Day_12
  > Day_13
  > Day_14
  > Day_15
  > Day_16
  > Day_17
    | create_docx.py
    | create_pdf.py
    | create_xlsx.py
    | employees.xlsx
    | histogram.png
  > Day_18
  > Day_19
  > Day_20
  LICENSE
  nytimes_subdomains.txt
  README.md

PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_17
gitpod /workspace/python-for-OSINT-21-days/Day_17 (main) $ python create_xlsx.py
```

Импортируем пакет XlsxWriter:

```
import xlsxwriter
```

Создайте новый файл (книгу) с именем employees.xlsx:

```
workbook = xlsxwriter.Workbook('employees.xlsx')
```

Добавляем в книгу пустой лист:

```
worksheet = workbook.add_worksheet()
```

Создайте двумерный список (имя сотрудника - возраст):

```
employees = (
    ['Name', 'Age'],
    ['John Smith', 33],
    ['Eric Gold', 26],
    ['Simon Silver', 37],
    ['James Conor', 50],
)
```

Пройдитесь по массиву и запишите имя сотрудника в столбце А и возраст в столбце В (при каждой попытке создается новая строка):

```
row = 0
```

```
for name, age in (employees):
    worksheet.write(row, 0, name)
    worksheet.write(row, 1, age)
    row += 1
```

В самую нижнюю строку добавьте формулу расчета среднего возраста:

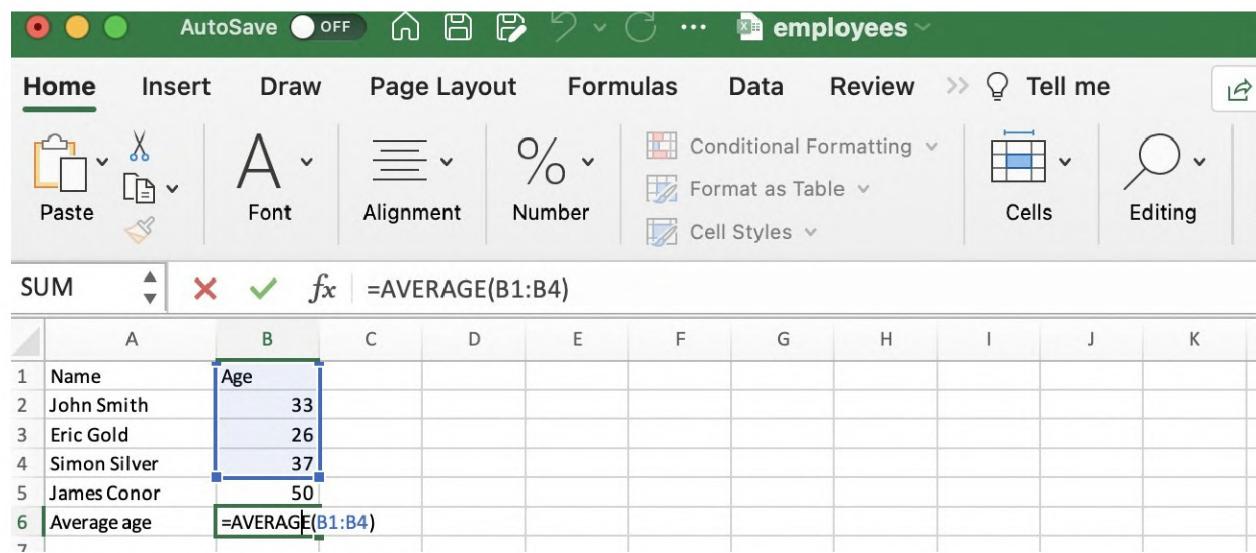
```
worksheet.write(row, 0, 'Average age')
```

```
worksheet.write(row, 1, '=average(B1:B'+str(row-1)+')')
```

Закрываем файл:

```
workbook.close()
```

Вот как будет выглядеть готовый документ, если открыть его в Excel:



The screenshot shows a Microsoft Excel spreadsheet titled "employees". The ribbon menu is visible at the top, showing tabs like Home, Insert, Draw, etc. The Home tab is selected. In the formula bar, the formula "=AVERAGE(B1:B4)" is entered. The table has columns A and B. Column A contains names: "Name", "John Smith", "Eric Gold", "Simon Silver", "James Conor", and "Average age". Column B contains ages: "33", "26", "37", "50", and the formula "=AVERAGE(B1:B4)". The cell containing the formula is selected.

	A	B	C	D	E	F	G	H	I	J	K
1	Name										
2	John Smith	33									
3	Eric Gold	26									
4	Simon Silver	37									
5	James Conor	50									
6	Average age	=AVERAGE(B1:B4)									
7											

Теперь попробуем создать документ Word. Для этого мы будем использовать пакет python-docx (<https://python-docx.readthedocs.io/en/latest/>).

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install python-docx
Collecting python-docx
  Downloading python-docx-0.8.11.tar.gz (5.6 MB)
    5.6/5.6 MB 21.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting lxml<=2.3.2
  Downloading lxml-4.9.2-cp311-cp311-manylinux_2_17_x86_64_manylinux2014_x86_64_manylinux_2_24_x86_64.whl (7.2 MB)
    7.2/7.2 MB 65.8 MB/s eta 0:00:00
Building wheels for collected packages: python-docx
  Building wheel for python-docx (setup.py) ... done
  Created wheel for python-docx: filename=python_docx-0.8.11-py3-none-any.whl size=184491 sha256=b7bc3f5e5dbc2e890532e6eaf57f7740702a4517dfc7036d862f07944a88856e
  Stored in directory: /workspace/.pyenv_mirror/pip_cache/wheels/b2/11/b8/209e41af524253c9ba6c2a8b8ecec0f98ecbc28c732512803c
Successfully built python-docx
Installing collected packages: lxml, python-docx
Successfully installed lxml-4.9.2 python-docx-0.8.11
```

```
pip install python-docx
```

Запустите `create_docx.py`:

The screenshot shows the VS Code interface. The Explorer sidebar on the left lists files and folders, including a folder named `PYTHON-FOR-OSINT-21-DAYS` which contains sub-folders from `Day_1` to `Day_17`, and several Python files like `create_docx.py`, `create_pdf.py`, and `create_xlsx.py`. The code editor window on the right displays the `create_docx.py` file with the following content:

```
Day_17 > 📁 create_docx.py X
1   from docx import Document
2   from docx.shared import Inches
3
4   document = Document()
5
6   document.add_heading('Report', 0)
7
8   document.add_heading('Report', level=1)
9
10  document.add_paragraph(
11      'Some text in report'
12  )
13
14  document.add_page_break()
15
16  document.add_picture('histogram.png', width=Inches(1.25))
17
18  document.save('report.docx')
19
20
```

Below the code editor are tabs for PROBLEMS, OUTPUT, PORTS, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab shows the command being run:

```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_17
gitpod /workspace/python-for-OSINT-21-days/Day_17 (main) $ python create_docx.py
gitpod /workspace/python-for-OSINT-21-days/Day_17 (main) $
```

Импорт пакетов:

```
from docx import Document
from docx.shared import Inches
```

Создайте новый документ:

```
document = Document()
```

Добавляем первый заголовок в документ:

```
document.add_heading('Report', 0)
```

Добавьте к документу второй заголовок:

```
document.add_heading('Report', level=1)
```

Добавляем текстовый абзац в документ:

```
document.add_paragraph( 'Some text in report' )
```

Добавляем разрыв страницы в документ:

```
document.add_page_break()
```

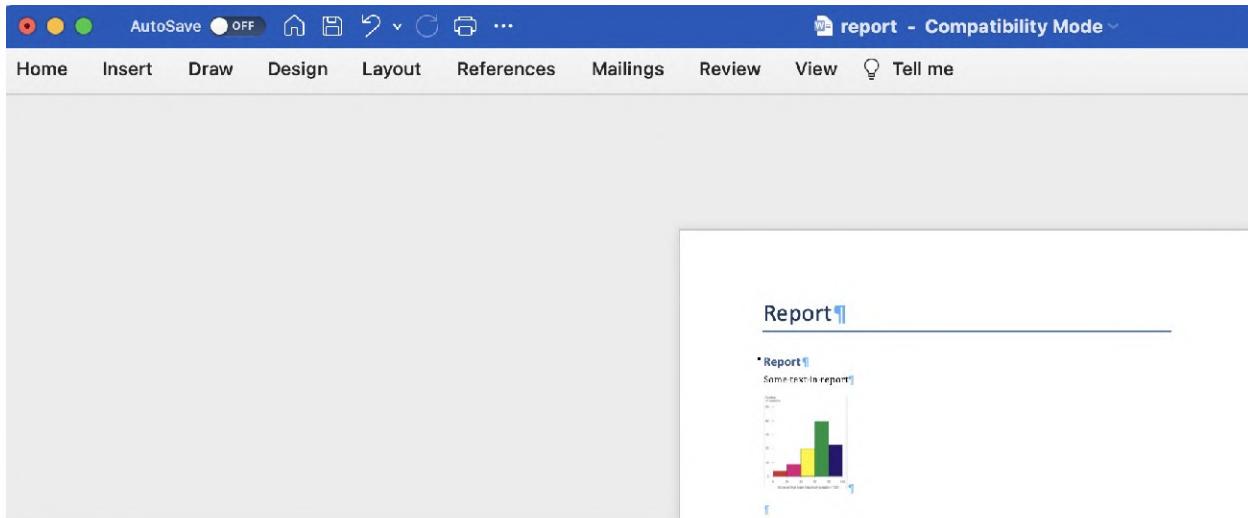
Добавляем изображение в документ:

```
document.add_picture('histogram.png', width=Inches(1.25))
```

Сохраняем документ:

```
document.save('report.docx')
```

Вот как будет выглядеть итоговый документ:



В заключительной части нашего урока давайте посмотрим, как создавать PDF-файлы. Для этого мы будем использовать пакеты FPDF (<https://pyfpdf.readthedocs.io/en/latest/>).

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install fpdf
Collecting fpdf
  Downloading fpdf-1.7.2.tar.gz (39 kB)
    Preparing metadata (setup.py) ... done
Building wheels for collected packages: fpdf
  Building wheel for fpdf (setup.py) ... done
    Created wheel for fpdf: filename=fpdf-1.7.2-py2.py3-none-any.whl size=40704 sha256=ddf9fdad6eb46d4b13e3453e9c8f500c8d7fbf4e378c1e0d
17beea95e7bc9f4a
    Stored in directory: /workspace/.pyenv_mirror/pip_cache/wheels/65/4f/66/bbda9866da446a72e206d6484cd97381cbc7859a7068541c36
Successfully built fpdf
Installing collected packages: fpdf
Successfully installed fpdf-1.7.2
```

pip install fpdf

Запустите `create_pdf.py`:

```
Day_17 > create_pdf.py
1 from fpdf import FPDF
2
3
4 pdfFile = FPDF()
5
6 pdfFile.add_page()
7
8 pdfFile.set_font("Arial", size = 12)
9
10 pdfFile.cell(20, 10, txt = "Report text", ln = 2, align = 'C')
11
12 pdfFile.image('histogram.png', 10, 40, 30)
13
14 pdfFile.output("report.pdf")
15
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

- gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_17
- gitpod /workspace/python-for-OSINT-21-days/Day_17 (main) \$ python create_pdf.py
- gitpod /workspace/python-for-OSINT-21-days/Day_17 (main) \$ []

OPEN EDITORS

- create_pdf.py Day_17 M

PYTHON-FOR-OSINT-21-DAYS

- > Day_1
- > Day_2
- > Day_3
- > Day_4
- > Day_5
- > Day_6
- > Day_7
- > Day_9
- > Day_10
- > Day_11
- > Day_12
- > Day_13
- > Day_14
- > Day_15
- > Day_16
- ✓ Day_17
- create_docx.py
- create_pdf.py M
- create_xlsx.py
- employees.xlsx U
- histogram.png
- report.docx U
- report.pdf U

Импорт пакета FPDF:

```
from fpdf import FPDF
```

Создаем пустой файл:

```
pdfFile = FPDF()
```

Создаём пустую страницу в файле:

```
pdfFile.add_page()
```

Настройте шрифт документа:

```
pdfFile.set_font("Arial", size = 12)
```

```
# Добавляем текст на страницу, указав координаты верхнего и нижнего  
отступов:
```

```
pdfFile.cell(20, 10, txt = "Report text", ln = 2, align = 'C')
```

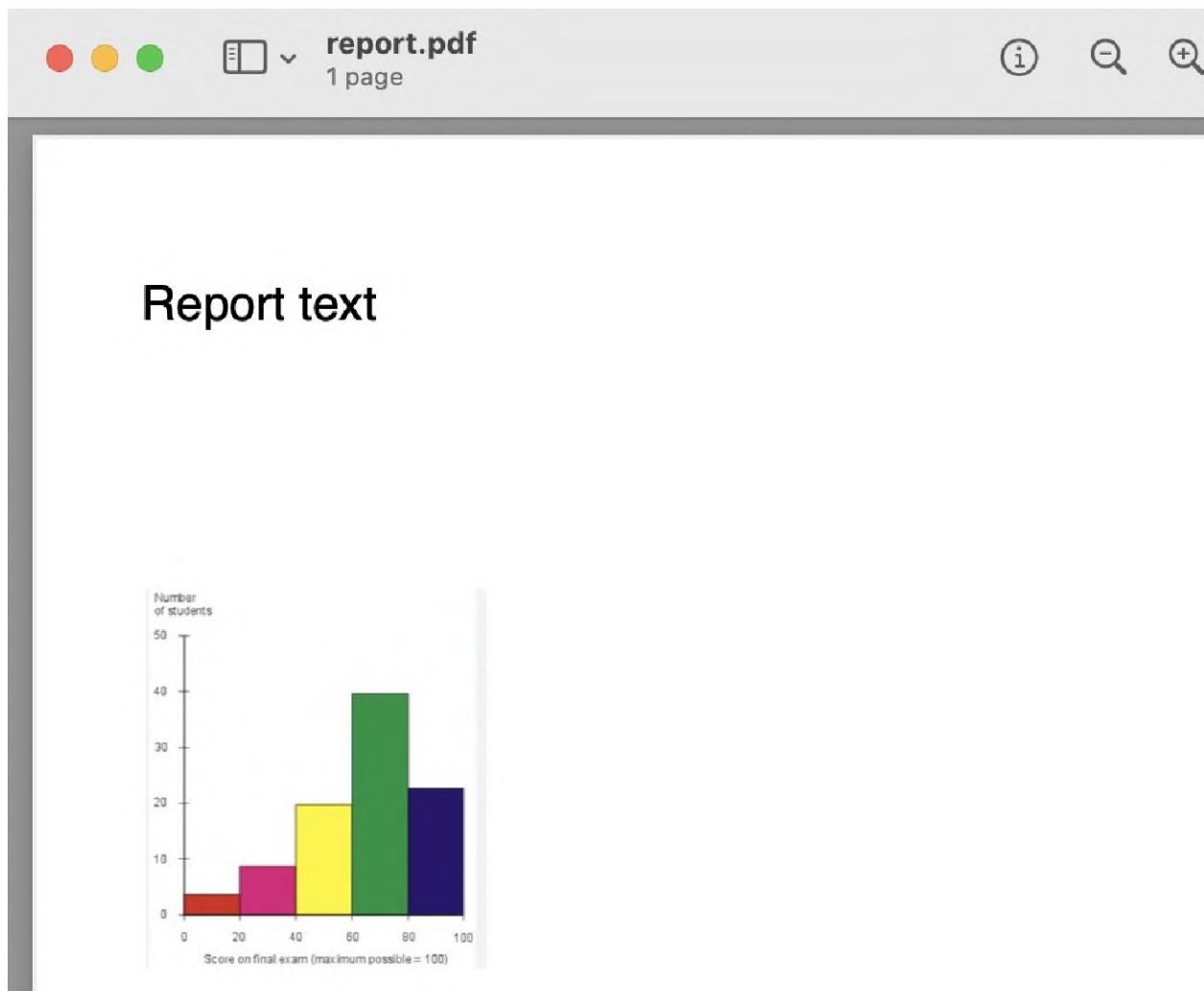
```
# Добавляем изображение на страницу, указав его размер и координаты  
верхнего и нижнего отступов:
```

```
pdfFile.image('histogram.png', 10, 40, 30)
```

```
# Сохраняем результат в файл report.pdf:
```

```
pdfFile.output("report.pdf")
```

Вот как будет выглядеть результат, если вы откроете его в программе просмотра PDF:



Презентации Microsoft Power Point также можно создавать с помощью Python с помощью пакета python-pptx (<https://python-pptx.readthedocs.io/en/latest/>).

День 18. Создание диаграмм и карт

На прошлом уроке мы вставили в документ изображение схемы. Такие снимки также могут быть созданы автоматически на основе данных, полученных в ходе расследования. Например, вы можете визуализировать данные из разных API.

Matplotlib (<https://matplotlib.org>) нам в этом поможет. Это библиотека для создания визуализаций данных. Мы также будем использовать пакет NumPy (<https://numpy.org>), который поможет нам работать с многомерными массивами.

Создание диаграмм

```
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.7.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
    11.6/11.6 MB 43.6 MB/s eta 0:00:00
Collecting contourpy>=1.0.1
  Downloading contourpy-1.0.7-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (299 kB)
    300.0/300.0 kB 43.7 MB/s eta 0:00:00
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.39.3-py3-none-any.whl (1.0 MB)
    1.0/1.0 MB 61.5 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.4 MB)
    1.4/1.4 MB 70.0 MB/s eta 0:00:00
Collecting numpy>=1.20
  Downloading numpy-1.24.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
    17.3/17.3 MB 83.5 MB/s eta 0:00:00
Requirement already satisfied: packaging>=20.0 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from matplotlib)
(23.0)
Collecting pillow>=6.2.0
  Downloading Pillow-9.5.0-cp311-cp311-manylinux_2_28_x86_64.whl (3.4 MB)
    3.4/3.4 MB 79.5 MB/s eta 0:00:00
Collecting pyparsing>=2.3.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
    98.3/98.3 kB 19.0 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.7 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from matplotlib)
(2.8.2)
Requirement already satisfied: six>=1.5 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from python-dateutil>=2.
7>matplotlib) (1.16.0)
Installing collected packages: pyparsing, pillow, numpy, kiwisolver, fonttools, cycler, contourpy, matplotlib
Successfully installed contourpy-1.0.7 cycler-0.11.0 fonttools-4.39.3 kiwisolver-1.4.4 matplotlib-3.7.1 numpy-1.24.3 pillow-9.5.0 pyp
arsing-3.0.9
```

Установите Matplotlib перед запуском скрипта (numpy устанавливается автоматически при установке Matplotlib):

```
pip install matplotlib
```

Запустите bar.py:

The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view of files and folders. The current folder is 'Day_18'. Inside, there's a file named 'bar.py' and several other files like 'map.py', 'population_barchart.png', 'LICENSE', 'nytimes_subdomains.txt', and 'README.md'. The main editor area shows the code for 'bar.py':

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 x = np.array(["Los Angeles", "San Francisco", "New York"])
5 y = np.array([3.8, 0.8, 8.4])
6
7 plt.bar(x,y)
8
9 plt.title ("Population")
10
11 plt.savefig('population_barchart.png')
12
```

Below the editor is a tab bar with 'PROBLEMS', 'OUTPUT', 'PORTS', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected, displaying the command history:

- gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_18
- gitpod /workspace/python-for-OSINT-21-days/Day_18 (main) \$ python bar.py
- gitpod /workspace/python-for-OSINT-21-days/Day_18 (main) \$

Импорт пакетов matplotlib и numpy:

```
import matplotlib.pyplot as plt
import numpy as np
```

Создайте список с названиями городов:

```
x = np.array(["Los Angeles", "San Francisco", "New York"])
```

Создайте список с числовыми значениями их численности в миллионах:

```
y = np.array([3.8, 0.8, 8.4])
```

Постройте график на основе этих двух списков:

```
plt.bar(x,y)
```

Добавляем заголовок к диаграмме:

```
plt.title ("Population")
```

Сохраняем результат в файл:

```
plt.savefig('population_barchart.png')
```

В этом примере мы использовали библиотеку NumPy для создания двух одномерных массивов, чтобы затем создать двумерный массив, на основе которого можно построить график.

В прошлом уроке мы создавали двумерные списки без использования питону, но если вы собираетесь использовать списки для создания каких-либо визуализаций с помощью Matplotlib, то лучше использовать питону-массивы (еще раз обратите внимание, что списки и массивы в Python — это разные вещи).).

Matplotlib позволяет создавать множество различных типов визуализации данных: линейную диаграмму, точечную диаграмму, пошаговую диаграмму, круговую диаграмму и сотни других. А в сочетании с набором инструментов Basemap (<https://matplotlib.org/basemap/>) вы даже можете визуализировать геоданные.

Создание карт

```

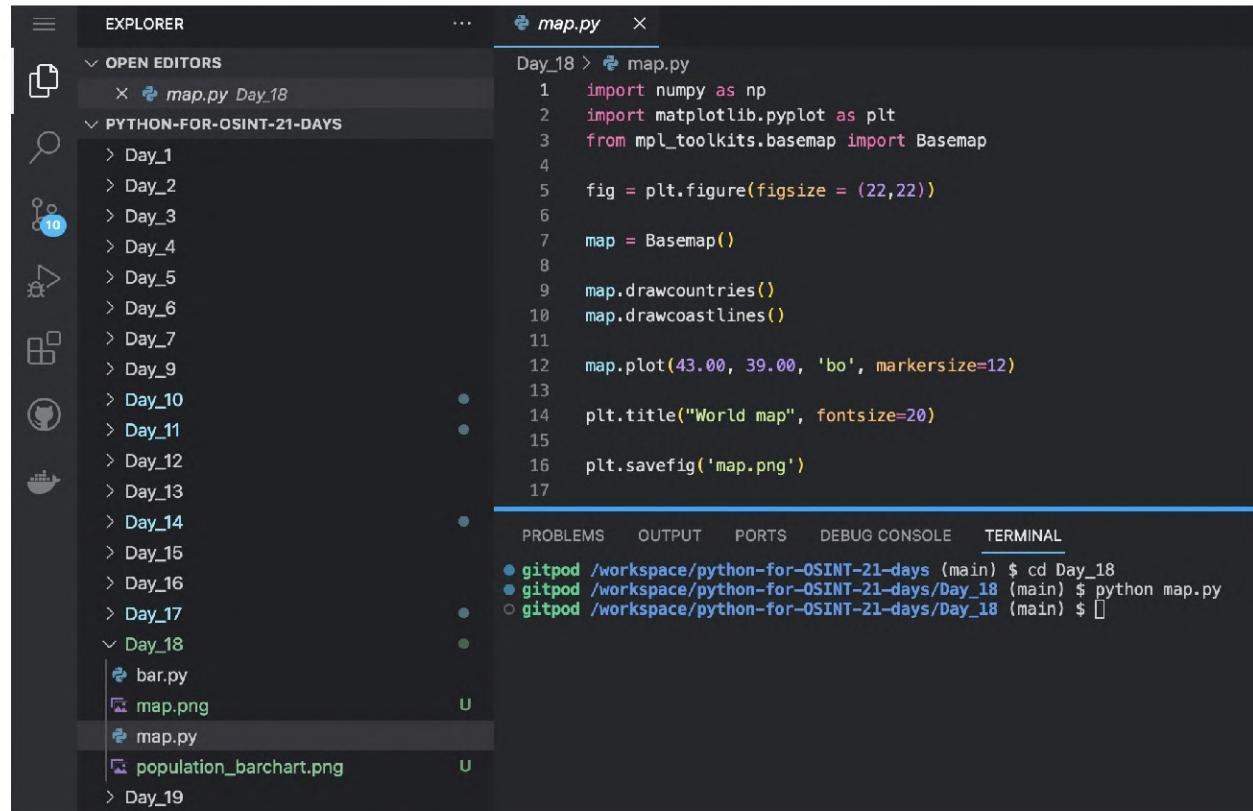
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install basemap
Collecting basemap
  Downloading basemap-1.3.6-cp311-cp311-manylinux1_x86_64.whl (860 kB)
    860.3/880.3 kB 8.0 MB/s eta 0:00:00
Collecting basemap-data<1.4,>=1.3.2
  Downloading basemap_data-1.3.2-py2.py3-none-any.whl (30.5 MB)
    30.5/30.5 MB 62.6 MB/s eta 0:00:00
Collecting pyshp<2.4,>=1.2
  Downloading pyshp-2.3.1-py2.py3-none-any.whl (46 kB)
    46.5/46.5 kB 11.6 MB/s eta 0:00:00
Collecting matplotlib<3.7,>=1.5
  Downloading matplotlib-3.6.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.8 MB)
    11.8/11.8 MB 112.1 MB/s eta 0:00:00
Collecting pyproj<3.5.0,>=1.9.3
  Downloading pyproj-3.4.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
    7.8/7.8 MB 115.6 MB/s eta 0:00:00
Collecting numpy<1.24,>=1.22
  Downloading numpy-1.23.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
    17.1/17.1 MB 106.6 MB/s eta 0:00:00
Requirement already satisfied: contourpy>=1.0.1 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from matplotlib<3.7,>=1.5->basemap) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from matplotlib<3.7,>=1.5->basemap) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from matplotlib<3.7,>=1.5->basemap) (4.39.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from matplotlib<3.7,>=1.5->basemap) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from matplotlib<3.7,>=1.5->basemap) (23.0)
Requirement already satisfied: pillow>=7.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib<3.7,>=1.5->basemap) (8.2.0)

```

Установите пакет базовой карты перед запуском скрипта:

`pip install basemap`

Запустите `map.py`:



```

EXPLORER          map.py
OPEN EDITORS      Day_18 > map.py
PYTHON-FOR-OSINT-21-DAYS
  Day_1
  Day_2
  Day_3
  Day_4
  Day_5
  Day_6
  Day_7
  Day_9
  Day_10
  Day_11
  Day_12
  Day_13
  Day_14
  Day_15
  Day_16
  Day_17
  Day_18
    bar.py
    map.png
    map.py
    population_barchart.png
  Day_19

map.py
Day_18 > map.py
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from mpl_toolkits.basemap import Basemap
4
5  fig = plt.figure(figsize = (22,22))
6
7  map = Basemap()
8
9  map.drawcountries()
10 map.drawcoastlines()
11
12 map.plot(43.00, 39.00, 'bo', markersize=12)
13
14 plt.title("World map", fontsize=20)
15
16 plt.savefig('map.png')
17

PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_18
● gitpod /workspace/python-for-OSINT-21-days/Day_18 (main) $ python map.py
○ gitpod /workspace/python-for-OSINT-21-days/Day_18 (main) $ 

```

Импорт пакетов matplotlib, num, basemap:

```
import numpy as np  
import matplotlib.pyplot as plt  
from mpl_toolkits.basemap import Basemap
```

Создаём пустое изображение:

```
fig = plt.figure(figsize = (22,22))
```

Создайте карту (по умолчанию создается карта мира):

```
map = Basemap()
```

Добавьте к нему границы страны:

```
map.drawcountries()
```

Добавьте к нему береговые линии:

```
map.drawcoastlines()
```

Добавьте к нему точку, указав географические координаты, тип и размер маркера:

```
map.plot(43.00, 39.00, 'bo', markersize=12)
```

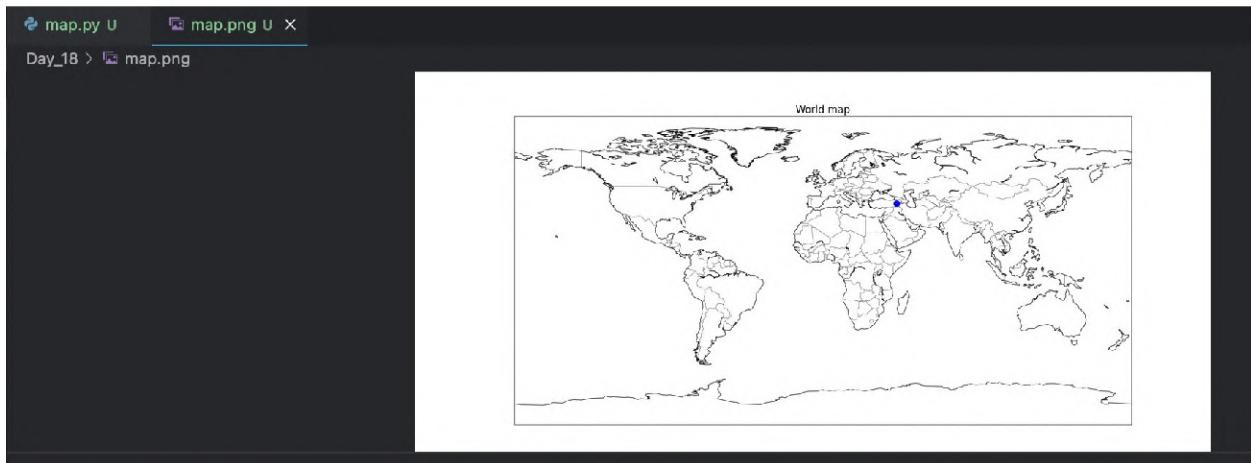
Добавьте заголовок к карте:

```
plt.title("World map", fontsize=20)
```

Сохраняем результат в файл:

```
plt.savefig('map.png')
```

Вот как должен выглядеть полученный файл map.png:

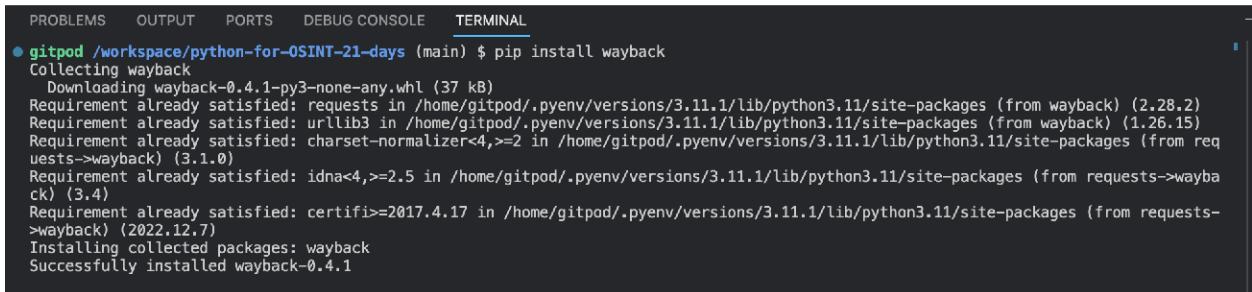


Базовая карта позволяет загружать любые карты из шейп-файлов (с расширением shp). Это позволяет делать визуализации для отдельных стран и регионов. Вы можете прочитать больше об этом здесь:
<https://basemaptutorial.readthedocs.io/en/latest/shapefile.html>

День 19. Wayback Machine и функции времени/даты

[Archive.org](#) — один из важнейших источников информации OSINT и SOCMINT (аналитика социальных сетей). Он позволяет найти удаленные контактные данные владельцев сайтов, проследить историю изменения информации в профиле в социальной сети или найти любой другой удаленный контент.

Мы будем использовать пакет Wayback (<https://pypi.org/project/wayback/>) для автоматизация работы с WaybackMachine. Также работа с архивами — отличный повод наконец изучить основы работы с датой и временем в Python.



```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
● gitpod /workspace/python-for-OSINT-21-days (main) $ pip install wayback
Collecting wayback
  Downloading wayback-0.4.1-py3-none-any.whl (37 kB)
    Requirement already satisfied: requests in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from wayback) (2.28.2)
    Requirement already satisfied: urllib3 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from wayback) (1.26.15)
    Requirement already satisfied: charset-normalizer<4,>=2 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests->wayback) (3.1.0)
    Requirement already satisfied: idna<4,>=2.5 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests->wayback) (3.4)
    Requirement already satisfied: certifi>=2017.4.17 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from requests->wayback) (2022.12.7)
    Installing collected packages: wayback
      Successfully installed wayback-0.4.1
```

Установите пакет wayback перед запуском скрипта:

```
pip install wayback
```

Запустите download_mementos.py:

```
EXPLORER          ⌂ download_mementos.py ×
OPEN EDITORS       Day_19 > ⌂ download_mementos.py
PYTHON-FOR-OSINT-21-DAYS
> Day_1
> Day_2
> Day_3
> Day_4
> Day_5
> Day_6
> Day_7
> Day_9
> Day_10
> Day_11
> Day_12
> Day_13
> Day_14
> Day_15
> Day_16
> Day_17
> Day_18
> Day_19
  date_time.py
  download_mementos.py
  ⌂ download_mementos.py
  ⌂ https://web.archive.org/web/19961...
  ⌂ https://web.archive.org/web/19970...
  ⌂ https://web.archive.org/web/1997071...
> Day_20
LICENSE
nytimes_subdomains.txt
README.md

Day_19 > ⌂ download_mementos.py
1 import wayback
2 from datetime import datetime
3
4 client = wayback.WaybackClient()
5
6 for record in client.search('http://nasa.gov', to_date=datetime(1999, 1, 1)):
7
8     memento = client.get_memento(record)
9
10    fileName=memento.memento_url.replace("/","-")+".html"
11
12    memento_file = open(fileName, "a")
13
14    memento_file.write(memento.text)
15
16    memento_file.close()
17
18    print (fileName)

PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL
● gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_19
● gitpod /workspace/python-for-OSINT-21-days/Day_19 (main) $ python download_mementos.py
https://web.archive.org/web/19961231235847id_...http://www.nasa.gov.html
https://web.archive.org/web/19970605230559id_...http://www.nasa.gov.html
https://web.archive.org/web/19970711094601id_...http://www.nasa.gov.html
```

Импорт пакетов pathback и datetime:

```
import wayback
from datetime import datetime
```

Создаем клиент веб-архива:

```
client = wayback.WaybackClient()
```

Поиск копий веб-страниц nasa.gov, сохраненных до 1999 года:

```
for record in client.search('http://nasa.gov', to_date=datetime(1999, 1, 1)):
```

Получить памятную запись (копию веб-страницы):

```
memento = client.get_memento(record)
```

Сгенерируем имя файла, в котором будем сохранять HTML-код копии веб-страницы (замените ссылку на страницу на / на - (чтобы при сохранении файла не возникало ошибок) и добавим расширение .html:

```
fileName=memento.memento_url.replace("/", "-")+".html"
```

Откройте файл, в котором мы сохраним HTML-код копии веб-страницы:

```
memento_file = open(fileName, "a")
```

Запишите HTML-код копии веб-страницы в файл:

```
memento_file.write(memento.text)
```

Закрыть файл:

```
memento_file.close()
```

Распечатать имя файла:

```
print (fileName)
```

Будьте готовы к тому, что запуск скрипта может занять некоторое время.

Обратите внимание, что в приведенном выше коде мы использовали `datetime()`, чтобы установить диапазон дат для поиска копий веб-страницы в веб-архиве.

Это очень важная функция. Давайте рассмотрим несколько примеров того, как с ним работать.

Запустите `date_time.py`:

```
date_time.py
1 import datetime
2
3 currentTime = datetime.datetime.now()
4
5 print(currentTime)
6
7 print("Current Year: "+str(currentTime.year))
8
9 print("Current Month: "+str(currentTime.month))
10
11 print(currentTime.strftime("%A %d %B %Y"))
12
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

- gitpod /workspace/python-for-OSINT-21-days (main) \$ cd Day_19
- gitpod /workspace/python-for-OSINT-21-days/Day_19 (main) \$ python date_time.py

```
2023-05-04 16:25:39.532865
Current Year: 2023
Current Month: 5
Thursday 04 May 2023
```

- gitpod /workspace/python-for-OSINT-21-days/Day_19 (main) \$

Импортируйте пакет `datetime` (доступен в Python по умолчанию):

```
import datetime
```

Помещаем текущую дату и время в переменную:

```
currentTime = datetime.datetime.now()
```

Отображение текущей даты и времени:

```
print(currentTime)
```

Отображение текущего года:

```
print("Current Year: "+str(currentTime.year))
```

Отображение текущего месяца:

```
print("Current Month: "+str(currentTime.month))
```

Отображает текущий день недели, день месяца, месяц и год:

```
print(currentTime.strftime("%A %d %B %Y"))
```

Вы можете заменить `datetime.datetime.now()` на другую дату. Например, `datetime.datetime(2023, 5, 4)`.

День 20. Создание веб-приложений

После сообщений о каких-либо инструментах командной строки OSINT читатели иногда спрашивают меня: «А нет ли веб-версии этого инструмента?»

После прохождения этого курса у вас вряд ли возникнут серьезные трудности с использованием инструментов командной строки. Ведь теперь вы знаете, как это легко.

Но тем не менее они есть у многих людей.

И если вы создали какой-нибудь полезный скрипт Python и хотите, чтобы его использовало как можно больше людей, вам следует подумать о том, чтобы превратить его в веб-приложение.

Существует множество способов создания веб-приложений на Python. Например, вы можете использовать такие фреймворки, как Django, Flask, Dash, Falcon и т. д. Они достаточно просты в использовании и освоении, но все же для этого курса я выбрал самый простой и быстрый вариант — пакет Streamlit (<https://streamlit.io>).

Пожалуйста, установите его с помощью pip:

```
gitpod /workspace/python-for-OSINT-21-days (main) $ pip install streamlit
Collecting streamlit
  Downloading streamlit-1.22.0-py2.py3-none-any.whl (8.9 MB)
    8.9/8.9 MB 28.2 MB/s eta 0:00:00
Collecting altair<5,>=3.2.0
  Downloading altair-4.2.2-py3-none-any.whl (813 kB)
    813.6/813.6 kB 57.9 MB/s eta 0:00:00
Collecting blinker>=1.0.0
  Downloading blinker-1.6.2-py3-none-any.whl (13 kB)
Collecting cachetools>=4.0
  Downloading cachetools-5.3.0-py3-none-any.whl (9.3 kB)
Requirement already satisfied: click>=7.0 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from streamlit) (8.1.3)
Requirement already satisfied: importlib-metadata>=1.4 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from streamlit) (6.1.0)
Requirement already satisfied: numpy in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from streamlit) (1.23.5)
Requirement already satisfied: packaging>=14.1 in /home/gitpod/.pyenv/versions/3.11.1/lib/python3.11/site-packages (from streamlit) (23.0)
Collecting pandas<3,>=0.25
  Downloading pandas-2.0.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
    12.2/12.2 MB 86.6 MB/s eta 0:00:00
Requirement already satisfied: pillow>=6.2.0 in /workspace/.pyenv_mirror/user/current/lib/python3.11/site-packages (from streamlit) (9.5.0)
Collecting protobuf<4,>=3.12
  Downloading protobuf-3.20.3-py2.py3-none-any.whl (162 kB)
    162.1/162.1 kB 35.4 MB/s eta 0:00:00
Collecting pyarrow>=4.0
  Downloading pyarrow-12.0.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (38.9 MB)
    38.9/38.9 MB 56.2 MB/s eta 0:00:00
Collecting pympler>=0.9
  Downloading Pympler-1.0.1-py3-none-any.whl (164 kB)
```

pip install streamlit

Теперь давайте запустим наше первое веб-приложение:

Запустите webapp.py (обратите внимание, что мы делаем это не так, как с другими файлами в этом уроке):

streamlit run webapp.py

The screenshot shows a terminal window with the following output:

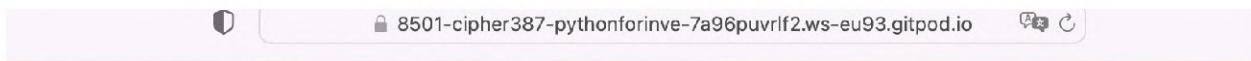
```
gitpod /workspace/python-for-OSINT-21-days (main) $ cd Day_20
gitpod /workspace/python-for-OSINT-21-days/Day_20 (main) $ streamlit run webapp.py
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.

You can now view your Streamlit app in your browser.
Network URL: http://10.0.5.2:8501
External URL: http://34.79.95.243:8501
```

Below the terminal, there is a message: "A service is available on port 8501". At the bottom right, there are buttons for "Source: Gitpod W...", "Make Public", "Open Preview", and "Open Browser".

После запуска скопируйте URL-адрес сети и откройте его в браузере.

Если вы используете Gitpod, просто нажмите «Открыть браузер».



Simple web app

Enter some text

yes

Start

You entered: yes

В результате должно получиться простое веб-приложение, отображающее текст, введенный пользователем после нажатия кнопки «Пуск».

Давайте посмотрим на код приложения.

Импортируем пакет Streamlit:

```
import streamlit as st
```

Создаем заголовок веб-страницы:

```
st.title("Simple web app")
```

Создаем текстовое поле:

```
textInput = st.text_input("Enter some text", "...")
```

Создаем кнопку, при нажатии на которую будет отображаться введенный в текстовое поле текст:

```
if(st.button("Start")):
```

```
nickname = textInput.title()  
st.text("You entered: "+textInput)
```

В конце этого урока предлагаю вам прочитать статью, в которой я расскажу, как с помощью Streamlit превратить Мегрэ (инструмент для подсчета никнеймов) в веб-приложение:

Самый простой способ превратить скрипт OSINT Python в веб-приложение. Объединение Мегрэ и Streamlit за три простых шага. ([The easiest way to turn an OSINT Python script into a web application. Combining Maigret and Streamlit in three simple steps](#))

Из него вы узнаете больше о различных полезных функциях Streamlit.

День 21. Инструменты, помогающие работать с кодом

Если вы напишете свой собственный код Python после окончания этого курса, вы, вероятно, постоянно будете сталкиваться с различными проблемами.

Во-первых, иногда у вас могут возникнуть сценарии, которые не запускаются из-за синтаксических ошибок.

The screenshot shows the ExtendsClass Python syntax checker interface. At the top, there's a navigation bar with 'ExtendsClass' logo, 'About', and links for 'Free Online Toolbox for developers', 'Browse Python file', and 'Check Python syntax'. Below the navigation is a code editor titled 'Python code' containing the following Python script:

```
3 from mpl_toolkits.basemap import Basemap  
4  
5 fig = plt..figure(figsize = (22,22))  
6  
7 map = Basemap()  
8  
9 map.drawcountries()  
10 map.drawcoastlines()  
11  
12 map.plot(43.00, 39.00, 'bo', markersize=12)  
13  
14 plt.title("World map", fontsize=20)  
15  
16 plt.savefig('map.png')
```

A yellow horizontal bar highlights the line 'fig = plt..figure(figsize = (22,22))'. Below the code editor, a pink box displays the error message:

Syntax errors detected :
Line 5:
fig = plt..figure(figsize = (22,22))
^
SyntaxError: invalid syntax

Но к счастью, их можно очень быстро найти с помощью многочисленных Онлайн-сервисов. Сервисы проверки синтаксиса. Например, <https://extendsclass.com/python-tester.html>

PythonChecker Makes Your Code Great Again

88%

Excellent

Lines: 16
Hints: 2

Your Code Sucks?



```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

fig = plt.figure(figsize = (22,22))

map = Basemap()

map.drawcountries()
map.drawcoastlines()

map.plot(43.00, 39.00, 'bo', markersize=12)

plt.title("World map", fontsize=20)

plt.savefig('map.png')
```

Если вы хотите показать свой код кому-то другому и сделать его максимально простым для чтения, я рекомендую использовать <https://www.pythonchecker.com>

Он подскажет вам, где поставить лишние пробелы и разрывы строк, а также укажет на слишком длинные комментарии в коде. Другими словами, он найдет проблемы, которые не мешают выполнению кода, но делают его менее понятным для других разработчиков.

This code is using three libraries - NumPy, Matplotlib and Basemap - to create a world map and plot a blue dot on the coordinates (43.00, 39.00).

First, the code imports the required libraries using the `import` statement. `numpy` is imported as `np` for convenience, `matplotlib.pyplot` is imported as `plt`, and `Basemap` is imported from the `mpl_toolkits.basemap` library.

Next, a new `figure` object is created using the `plt.figure()` method, with a size of 22x22 inches.

A new `Basemap` object is created using the default constructor. This creates a new map with default settings, centered at (0, 0) and spanning the entire globe.

The `drawcountries()` and `drawcoastlines()` methods are called on the `Basemap` object to draw the country borders and coastlines on the map.

Finally, a blue dot is plotted on the map using the `plot()` method with the coordinates (43.00, 39.00), and saved as a PNG file using the `savefig()` method with the filename 'map.png'.

The code ends with a title added to the plot using the `title()` method with the text "World map" and a font size of 20.

[Regenerate response](#)

Output in English Tone Default Writing Style Default Continue

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts.

И, конечно же, вы можете просто попросить [ChatGPT](#) доработать ваш код и предложить решения любых проблем. Это также поможет вам разобраться в коде других людей.

Но самое главное, что нужно помнить при работе с ChatGPT, это то, что он допускает ОЧЕНЬ много ошибок.

Поэтому лучше стремиться самостоятельно найти решение разных проблем, чтобы разобраться во всех деталях.

The screenshot shows the Stack Overflow homepage with a search bar containing '[python]'. The left sidebar is expanded, showing navigation links like Home, PUBLIC Questions, Tags, Users, Companies, COLLECTIVES, and TEAMS. The 'Questions' link is highlighted. A central content area displays 'Questions tagged [python]'. It includes a brief description of Python, a count of 2,126,413 questions, and a list of filters (Newest, Active, Bountied 30, Unanswered, More). Below this is a specific question card:

How can I change a column to a column from another dataframe based on a condition?
0 votes | 0 answers | 9 views
I have this data frame "cus" enter image description here and need to replace column "nationality_code" with respective weight from this dataframe "scores" enter image ...
python pandas replace conditional-statements
Andraws Santos 1 asked 6 mins ago

[Stackoverflow](#) — крупнейшее в мире онлайн-сообщество, где можно обсуждать вопросы, связанные с программированием и компьютерными технологиями в целом. Уже создано более 2 миллионов вопросов, связанных с Python, и вы действительно можете найти решения большинства проблем.

phind

SESSION #1

```
what this code to do. import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

fig = plt.figure(figsize = (22,22))

map = Basemap()

map.drawcountries()
map.drawcoastlines()

map.plot(43.00, 39.00, 'bo', markersize=12)

plt.title("World map", fontsize=20)

plt.savefig('map.png')
```

319/3000

Expert Concise Creative

what this code to do. import numpy as np import matplotlib.pyplot as plt from mpl_toolkits.basemap import Basemap fig = plt.figure(figsize = (22,22)) map = Basemap() map.drawcountries() map.drawcoastlines() map.plot(43.00, 39.00, 'bo', markersize=12) plt.title("World map", fontsize=20) plt.savefig('map.png')

ANSWER

The code provided imports three libraries: numpy, matplotlib.pyplot, and Basemap. It then creates a new figure with a size of 22x22 using `fig = plt.figure(figsize = (22,22))`. Next, it creates a new Basemap object using `map = Basemap()`. The Basemap object is used to draw country borders and coastlines using `map.drawcountries()` and `map.drawcoastlines()`. Finally, the code plots a blue dot on the map at the coordinates (43.00, 39.00) using `map.plot(43.00, 39.00, 'bo', markersize=12)` and saves the resulting figure as a PNG file named `'map.png'` using `plt.savefig('map.png')`. matplotlib.org

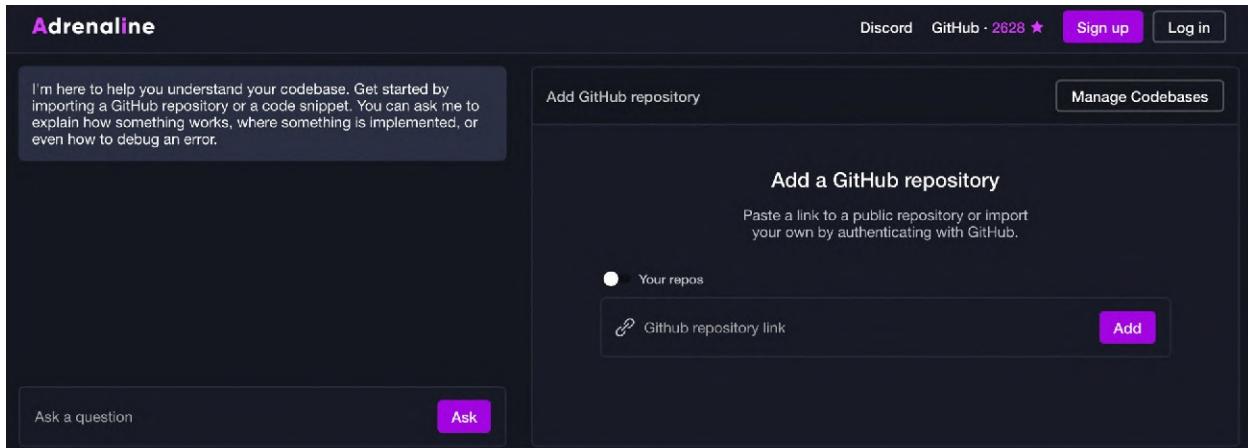
The `plot()` function is a method in the `matplotlib.pyplot` module that is used to draw lines and/or markers on a graph. It takes two arguments, `x` and `y`, which are the data to be plotted on the x and y axes, respectively. The `plot()` function can also take additional parameters to customize the appearance of the plot, such as the line color, line style, and marker style. In the provided code, the `plot()` function is used to plot a blue circle marker at the coordinates (43.00, 39.00) on the map. matplotlib.org

Basemap is a matplotlib toolkit for plotting 2D data on maps in Python. It is used to create maps with different projections, such as cylindrical or conic projections, and to add features to the map such as coastlines, country borders, and geographic labels. Basemap is no longer actively developed, and users are encouraged to switch to Cartopy, a similar package that is actively maintained. matplotlib.org

There are several approaches to installing matplotlib. One way is to use pip, a package installer for Python, to install matplotlib. To install matplotlib using pip, open a terminal or command prompt and type `pip install matplotlib`. If you are using a virtual environment, be sure to activate it before running the `pip install` command. Another approach is to install matplotlib using a package manager such as Anaconda or Homebrew. [\[2\]](#) [\[4\]](#)

If you encounter an error such as "No module named 'matplotlib'", it may be because matplotlib is not installed or is not installed correctly. In this case, you can try reinstalling matplotlib using pip or another package manager. If you are using a Jupyter notebook, you may need to install matplotlib and import it before importing other libraries such as pandas. You can do this by running the command `import matplotlib.pyplot as plt; %matplotlib inline` before importing pandas. [\[2\]](#) [\[4\]](#)

Phind.com — это инструмент поиска на основе искусственного интеллекта, который ищет релевантные ответы в Stackoverflow, анализирует их и генерирует на их основе решение. Иногда он также использует другие IT-сайты в качестве источников информации.



Иногда, чтобы найти код, выполняющий конкретную задачу в более крупном проекте, вам приходится просмотреть огромный репозиторий Github, содержащий множество связанных файлов.

<https://useadrenaline.com/app> поможет автоматизировать эту работу. Просто добавьте ссылку на один или несколько репозиториев, а затем задайте вопросы об их содержимом чат-боту с искусственным интеллектом.

Дополнительные инструменты, упрощающие работу с кодом, можно найти в этом разделе репозиторий в моем профиле Github:

<https://github.com/cipher387/code-understanding-tools>

Что делать дальше

Это зависит от ваших целей. Если вы OSINT-специалист, то я рекомендую вам больше практиковаться в расследованиях и пробовать использовать некоторые инструменты, написанные на Python (их очень много в моем [аккаунте в Твиттере](#)), а также получать дополнительные знания по мере необходимости, когда вы сталкиваетесь с новыми задачами.

Если ваша цель — разработать какой-то серьезный проект на Python, вам обязательно понадобятся дополнительные курсы. Например, вы можете начать с прохождения бесплатного официального курса <https://www.learnpython.org/> от Python.org и Datacamp.

Вполне вероятно, что после прохождения этого курса у вас возникли идеи относительно новых простых инструментов для OSINT. У вас уже достаточно знаний, чтобы сделать много полезных вещей.

Если вы хотите сделать свой инструмент общедоступным и найти своих пользователей, я рекомендую читая эту статью:

10 очень простых советов для разработчиков OSINT-инструментов ([10 very simple tips for OSINT tools developers](#))

И не забудьте подписаться на меня в Twitter, Mastodon, Telegram, Substack, Discord, Medium, Substack, чтобы регулярно получать новости о новых OSINT-инструментах, а также не пропускать новые обучающие курсы.

Оглавление

Для кого этот курс?	2
Кому следует избегать этого курса?	3
Как пройти этот курс	4
День 0. Подготовка к работе	6
День 1. Запускаем первый скрипт	9
День 2. Минимальный базовый синтаксис	13
День 3. Установка и запуск инструментов командной строки Python	20
День 4. Чтение и запись файлов	26
День 5. Обработка HTTP-запросов и работа с API	33
День 6. JSON	38
День 7. CSV	43
День 8. Базы данных	52
День 9. Автоматизируйте сбор результатов поиска	56
День 10. Scraping	64
День 11. Регулярные выражения	69
День 12. Прокси	72
День 13. Функции для работы со списками	79
День 14. Работа с файловой системой	85
День 15. Сбор информации о домене	91
День 16. Отображение списка и функции для работы со строками	99
День 17. Генерация документов	106
День 18. Создание диаграмм и карт	115
День 19. Wayback Machine и функции времени/даты	121
День 20. Создание веб-приложений	126
День 21. Инструменты, помогающие работать с кодом	130
Что делать дальше	136
Оглавление	137