Разбор задач конкурса "Звёзды Кассиопеи"

Надобных Дмитрий, Пугач Сергей, Чечеватов Роман

20.05.2021

1 Ручные задачи

Для решения данных задач не нужны никакие навыки программирования и они вам не сильно помогут. Вместо этого участникам предлагалось научиться искать информацию в файлах, анализировать их.

1.1 MP5

Эта задача засчитывалась как решенная после перехода по ссылке. Ничего сложного.

1.2 Автостоп

Как сказано в условии, ответом на данную задачу является ответ на Главный вопрос жизни, вселенной и всего такого. Вопрос этот был задан английским писателем Дугласом Адамсом в книге "The Hitchhiker's Guide to the Galaxy русским читателям известной как "Автостопом по галактике". Участники, читавшие этот роман могут спокойно ответить "42"и получить баллы, остальным придётся скопировать вопрос в Гугл и получить ответ в первом же результате. Ответ: 42

1.3 Язык программирования

Для решения этой задачи участникам необходимо было изучить HTML-код страницы задачи. Для этого можно воспользоваться встроенными в браузер инструментами разработчика, или скачать страницу и открыть в текстовом редакторе. Независимо от выбранного способа, ответ находился в комментарии рядом с формой ввода ответа.
Ответ: flag{html_is_the_best_language}

1.4 Фотосессия

В данной задаче участникам давался для анализа JPG-файл с картинкой. Однако, такие файлы зачастую содержат в себе не только данные изображения, но и дополнительную информацию. Например, дату съемки, модель камеры, имя автора. И среди этой дополнительной информации и был спрятан флаг.
Ответ: flag{haha_you_thought_it_will_be_ccv}

1.5 PI

Здесь участников просили ввести число Пи с некоторой точностью: строго заданное число знаков после запятой, не больше и не меньше. Достаточно точное число Пи можно взять на Википедии или, для пользователей Windows 10, Калькуляторе.
Отгадать необходимую точность можно с помощью метода проб и ошибок, или просто написав бота.
Ответ: 3.14159265358979323846264338327

1.6 Задача, которую невозможно решить

Вопреки сказанному в названии задачи, её возможно решить. Наличие решения этой задачи показывает, как внимательно участники прочитали инструкцию по работе с тестирующей системой. Ведь именно в разделе "Мануал в подразделе "Обработка персональных данных совершенно не к месту находилось предложение "Однако, для решения задачи, которую невозможно решить, вам нужно всего лишь отправить название конкурса". Остаётся лишь найти правильное с точностью до символа название.
Ответ: Звёзды Кассиопеи

1.7 Двадцать пятый кадр

В данной задаче участникам предлагался для анализа файл с видео. И хотя в метаданных на этот раз флага не было, там можно было обнаружить частоту кадров видео: 120 кадров в секунду. Это значит, что на экранах с частотой обновления 60 Гц и менее (а таких на момент проведения конкурсов подавляющее большинство) немалая часть кадров просто не будет отображаться. Но зачем же авторы дали видео с таким большим числом кадров, зная, что часть из них не будет показана? Неужели они хотели что-то там спрятать? Да. Смотрим видео покадрово и на 817 кадре замечаем серый текст на сером фоне.
Ответ: flag{you_have_great_eyes}

1.8 Eglishman

В очередной раз отправляемся в Гугл (Яндекс, Bing, DuckDuckGo, Рамблер, Поиск Маил.Ру, ...) чтобы узнать, как сервер может узнать местоположение/страну/язык пользователя и узнаём, два основных способа: по IP и по предпочитаемому языку пользователя. Методом проб и ошибок, испробовав все платные и бесплатные VPN-сервера узнаем, что в данной задаче IP ничего не значит. Остаётся вариант с языком. Браузер внутри запроса отправляет информацию о предпочитаемом языке веб-страниц. В настройках браузера выставляем предпочтение на английский, отправляем что угодно в качестве ответа и получаем баллы.

1.9 1C

В следующей задаче на анализ содержимого файла участникам был дан файл Excel. Изучив свойства файла или создав новый лист, можно обнаружить, что кроме видимого листа с многозначащим названием "Лист1"в книге есть ещё два: "Лист2"и "Лист3". В Microsoft Office Excel жмём ПКМ по списку листов, "Показать..."Лист2". Изучаем содержимое ранее скрытого листа рядом с красивой картинкой аэродинамических характеристик коровы обнаруживаем флаг. Ответ: flag{cow_aerodynamics_power}

1.10 Машина времени

Снова скачиваем приложенный файл. Распаковываем архив, наряду с привычными, но ничем не примечательными файлами обнаруживаем папку .git. Или не обнаруживаем, если у вас не включено отображение скрытых файлов. В таком случае, включаем, потому что без него заниматься поиском скрытой информации весьма непросто. В любом случае, закинув название папки в Гугл получаем понимание, что авторы дали вам Git-репозиторий. Git - одна из систем контроля версий, то есть позволяет отслеживать и сохранять изменения файлов, откатываться к предыдущей версии. Скачиваем Git, устанавливаем. Делаем коммит с помощью `git commit -a` , видим, что после последнего сохранения состояния файлов. Откатываемся к предыдущему коммиту `git checkout a23d2c7`, смотрим содержимое когда-то бывшего удаленным файла, находим флаг. Ответ: flag{wayback_machine_master}

1.11 Плёночный фотоаппарат

1.12 Плёночный фотоаппарат 2

2 Задачи на использование ботов

Для начала рассмторим общую для всех ботов часть: работу с сетью. Мы использовали для написания ботов `Python 3` и библиотеку `requests`. Ничто не мешало вам использовать другие варианты, а незнание любого из них компенсируется длиной конкурса и доступностью всё того же Гугла.

Импортируем библиотеку и задаём константы:

```
# -*- coding: utf-8 -*-                          # Волшебное слово, чтобы использовать кириллицу
import requests                                  # Импортируем requests

login = 'login'                                  # Сохраняем логин и токен
token = '23847634320439b65adafb7e3fefff802b1cde0a1b2977c3fe299a6dd111e4ce'
host = 'https://fetefot763.eu.pythonanywhere.com/'
```

Теперь нужно разобраться, как браузер отправляет запросы на сайт. Открываем панель разработчика и изучаем (см. рисунок 1):

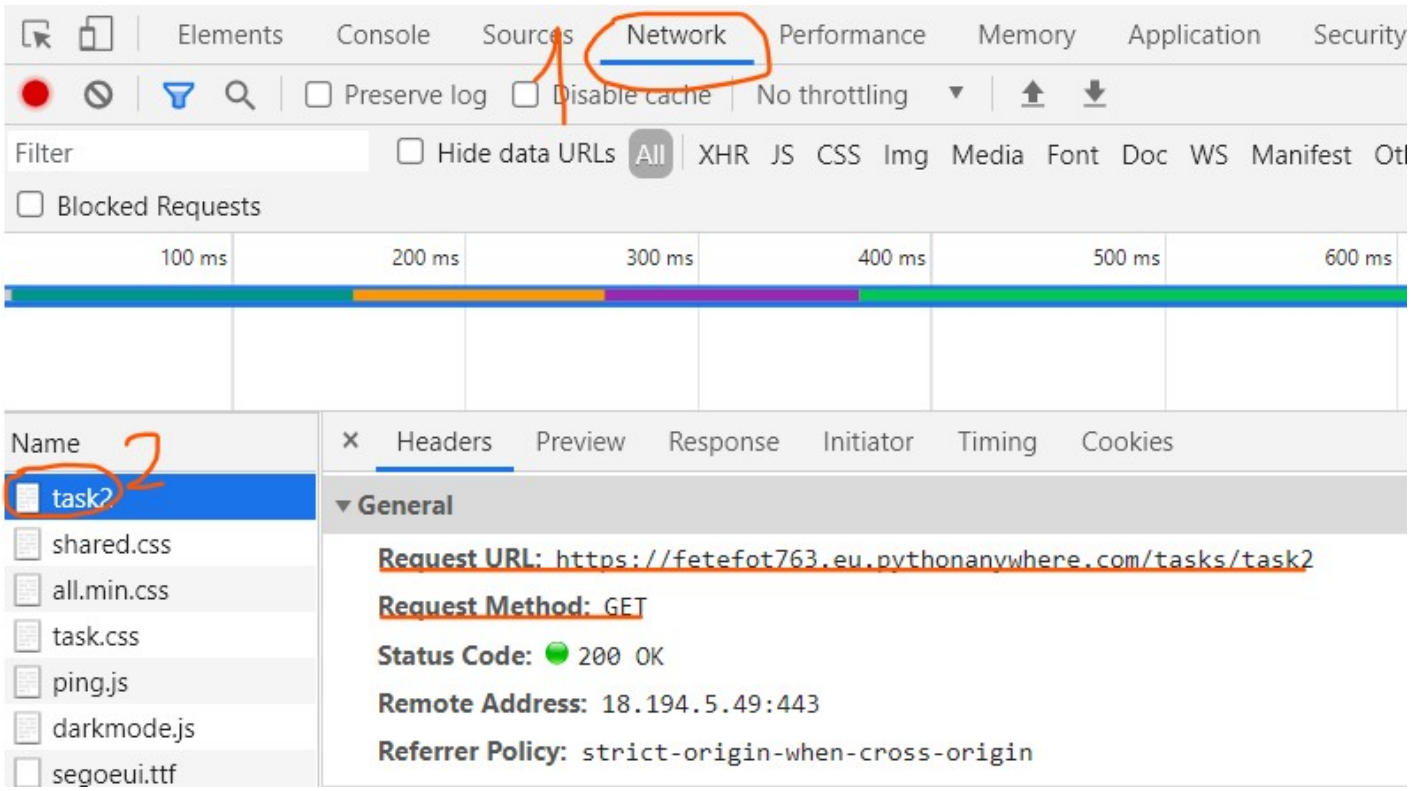

Рис. 1: Изучение процесса получения страницы браузером.

Видим, что браузер отправляет GET-запрос на страницу задачи. Продолжаем изучение, промотываем вниз (см. рисунок 2):

Рис. 2: Изучение отправляемых браузером cookie-файлов.

Теперь видим, что браузер отправляет cookie-файлы, чтобы получить страницу. Почитав в мануале обнаруживаем, что для работы необходимо только две печеньки: `login` и `token`. Их значения сохраняем к себе и используем дальше.
Используя полученные знания мы можем написать функцию получения задачи. Функция - это обособленный фрагмент кода, выполняющий узкую задачу.
Для задач, где в приложении дан текст, мы можем получить этот текст прямо со страницы задачи:

```
# Для текстовых задач (на примере задачи №1 - Водолей)
def get_data():                                     # Объявляем функцию
    text = requests.get(                            # Отправляем запрос
        host + 'tasks/task1',                       # На страницу задачи
        cookies={'login': login, "token": token}  # С cookies авторизации
    ).text                                          # И сохраняем текст ответа в переменную text
    return text                                     # Возвращаем text
```

Заметим, что функция `get_data` возвращает нам весь HTML-код страницы. Нам же нужно только приложение к заданию. Приложения всегда находятся в блоке с `id="task_data"`. И не содержат блоков внутри. Пользуясь этим, создадим новую функцию для извлечения этого приложения:

```
def get_text_from_data(data):
    marker_beg = '<div id="task-data">'
    marker_end = '</div>'
    ind_beg = data.rfind(marker_beg) + len(marker_beg)  # Индекс начала содержимого блока в строке
    ind_end = data[ind_beg:].find(marker_end)           # Индекс конца содержимого блока в строке
    text = data[ind_beg:ind_beg + ind_end]              # Сохраняем содержимое блока в переменную
    return text                                         # Возвращаем text
```

Следует чётко понимать, какую задачу вы решаете и какие функции из приложенных вам следует использовать. Так, функция получения текста задачи вернет мусор, если текста в задаче не окажется. А что произойдет с вашей программой при попадании внутрь мусора никому не известно.

Если же в приложении дана ссылка на файл, то его нужно скачать. Тогда функция может ничего не возвращать: результат запроса уже сохранен в файл и будет прочитан из него.

```
# Для файловых задач (на примере задачи №2 - WinRar 3000)
def get_data():                                  # Объявляем функцию
    requests.get(                                # Отправляем запрос
        host + 'tasks/task2',                    # На страницу задачи
        cookies={'login': login, "token": token}  # С cookies авторизации
    )
    with open('2.zip', 'wb') as file:            # Создаём и открываем файл 2.zip для записи
        file.write(                              # Записываем в файл
            requests.get(                        # Отправляем запрос
                host + f'task-generated-content/2/task2_{team_id}.zip', # За файлом
                cookies={'login': login, "token": token} # С cookies авторизации
            ).content                            # Записываем в файл ответ на запрос
        )
```

Теперь проанализируем отправку браузером нашего решения задачи. Напишем что-нибудь в поле ответа и нажмем кнопку отправки. Посмотрим, что же нам показывает браузер в уже знакомой панели (см. рисунок 3):

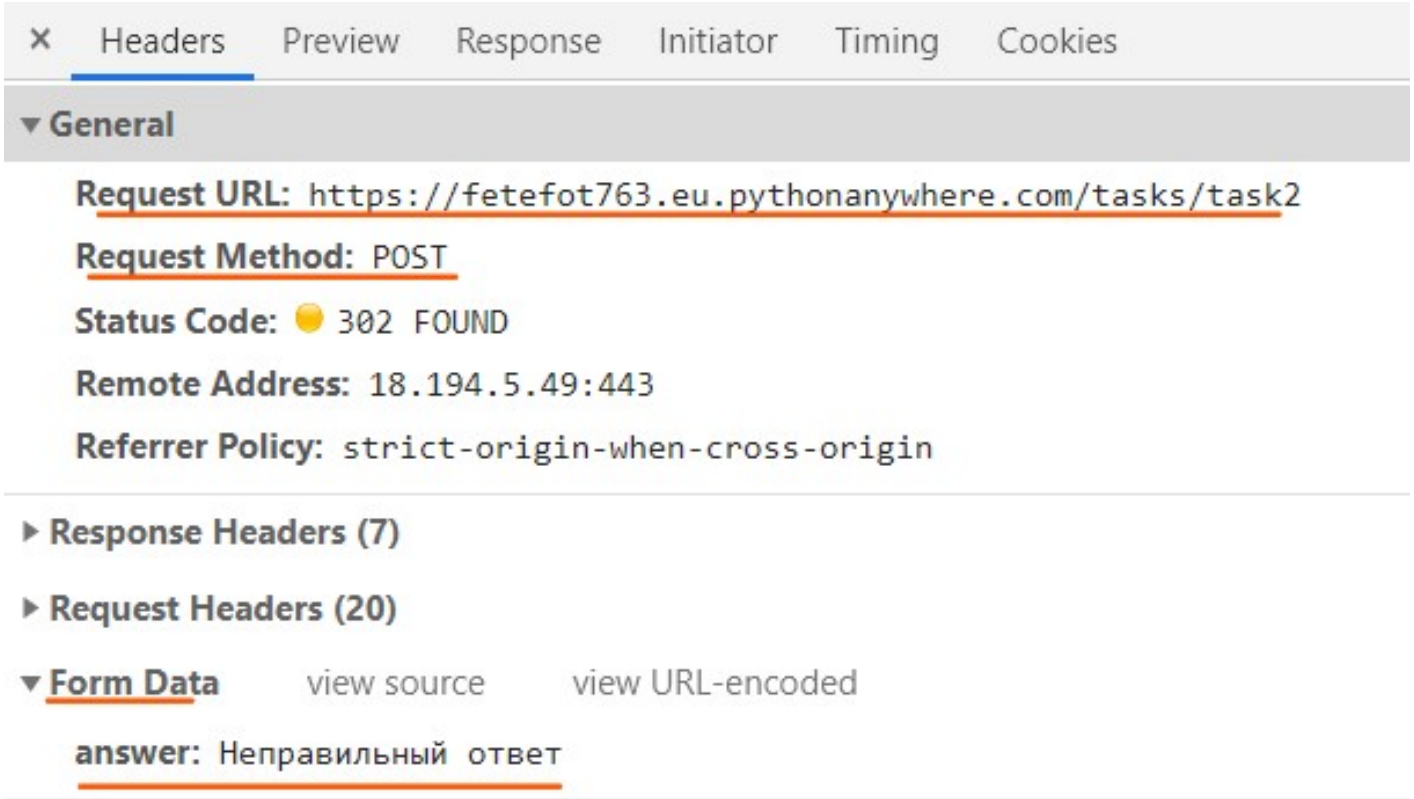

Рис. 3: Изучение отправки браузером ответа на задачу.

Видим, что браузер отправляет запрос всё на ту же страницу, он уже POST. В запрос браузер включает `данные формы`, содержащие поле `answer` со значением, которое мы только что вписали в поле ввода ответа. Ещё замечаем что браузер снова отправляет cookie. Он их вообще всегда отправ-

ляет.
Напишем функцию отправки ответа в данных формы:

```
def send_data(data):                                # Объявляем функцию
    requests.post(                                  # Отправляем запрос
        host + f'tasks/task{task_id}',              # На страницу задачи
        data={'answer': data},                      # С данными формы
        cookies={'login': login, "token": token}    # И cookies авторизации
    )
```

На этом работа с сетью завершается. Остаётся только в правильном порядке вызывать наши функции:

```
# Для текстовых задач
if __name__ == '__main__':                          # Если запущен именно этот файл
    while True:                                     # Вечный цикл
        txt = get_text_from_data(get_data())        # Получаем текст задачи
        res = solve(txt)                            # Решаем задачу и сохраняем ответ
        send_data(res)                              # Отправляем ответ
```

```
# Для файловых задач
if __name__ == '__main__':                          # Если запущен именно этот файл
    while True:                                     # Вечный цикл
        get_data()                                  # Получаем данные
        res = solve()                               # Решаем задачу и сохраняем ответ
        send_data(solve())                          # Отправляем ответ
```

Просто вставить этот код себе и запустить у вас не получится: функция `solve` неопределена. Реализацию этой функции мы рассмортим отдельно для каждой задачи.

2.1 Школьный конкурс компьютерной графики на тему "Программирование"

Здесь ничего решать не нужно, нужно просто отправить `29 апреля`. Тогда фукция `solve` может просто возвращать эту строку, без каких-либо вычислений.

```
def solve(txt):
    return '29 апреля'
```

2.2 Водолей

2.3 WinRar 3000

2.4 Никита Егоров

2.5 Цветовод

2.6 Кадровое агенство

2.7 Вечеринка

2.8 Календарь

2.9 Я - Гуль

2.10 Рукой подать

2.11 Emoji Warrior

2.12 CuSo4