МИНОБРАЗОВАНИЕ РОСТОВСКОЙ ОБЛАСТИ

государственное бюджетное профессиональное

образовательное учреждение Ростовской области

«Ростовский-на-Дону колледж радиоэлектроники,

информационных и промышленных технологий»

(ГБПОУ РО «РКРИПТ»)

**ИНДИВИДУАЛЬНЫЙ ПРОЕКТ**

|  |  |
| --- | --- |
| **Тема проекта:** | Автоматизация получения данных о расписании занятий колледжа |
| **Работу выполнил** | Забаренко Никита Иванович |
| **Группа** | КСК-11 |
| **Специальность** | 09.02.01 Компьютерные сети и комплексы |
| **Дисциплина** | Технология |
| **Преподаватель** | Костыльга А.Г. |
| **Оценка за проект** |  |

|  |  |
| --- | --- |
|  | /Костыльга А.Г./ |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | |

**Содержание**

Введение…………………………………………………………………………...3

Устройство языка программирования Python…………………………………...4

Алгоритм программы……………………………………………………………..7

Разбор исходного кода программы………………………………………………9

Дистрибуция……………………………………………………………………..12

Вместо заключения: возможные ошибки и перспективы развития проекта...13

Список использованных источников и литературы…………………………...14

**Введение**

Прежде чем начать описывать проект, представим ситуацию. У человека есть своя ферма с кучей растений, которые надо поливать. Но у него нет никакого источника воды, кроме колодца, находящегося в ста метрах от его участка. Человек может всю жизнь ходить туда и обратно с парой ведер, но чего ему стоит напрячься один раз и сделать водопровод, чтобы потом не напрягаться вообще?

Это малосвязанная с темой моего проекта ситуация в плане условий и результата, но в плане оптимизации – связана напрямую. Этот проект зародился по самой абсурдной и в то же время банальной причине – мне стало лень загружать каждую неделю расписание своих занятий.

Особенно неприятна такая ситуация, когда в воскресенье уже выключил компьютер и собираешься ложиться спать, как вдруг вспоминаешь, что не узнал, к какой паре и к какому времени нужно подготовиться. И тогда нужно снова включать компьютер или лезть в телефон…

И тогда я подумал «Почему бы мне не заставить компьютер обновлять и показывать мое расписание самому?»

**Устройство языка программирования Python**

Python – это интерпретируемый язык программирования общего назначения. Имеет приятный глазу, простой и минималистичный синтаксис, несмотря на то, что функционал его стандартной библиотеки очень широк.

Python поддерживает объектно-ориентированное, структурное, функциональное и аспектно-ориентированное программирование. Из особенностей архитектуры стоит выделить динамическую типизацию, автоматическое управление памятью, механизм обработки исключений, многопоточные вычисления и высокоуровневую структуру данных.

Для того, чтобы начать программирование на Python, нужно установить его с официального сайта. Для большего удобства можно установить также интегрированную среду разработки (IDE).

Простейшая программа может выглядеть вот так:

*print(“Hello world!”)*

Эта программа вызывает функцию печати в командную строку и передает ей на вывод строку «Hello world!». После выполнения программы на экране высветится эта надпись без кавычек.

В Python поддерживается работа с переменными с динамической типизацией:

*a = 123*

*b = “123”*

*c = a + b*

Эта программа создаст три переменные, причем в первой (а) будет храниться число 123, а во второй – строка «123», т.е. набор символов из «1», «2» и «3». На третьей строке интерпретатор прекратит работу программы и выдаст ошибку, т.к. сложить число с набором символов невозможно.

А вот как в языке пишется условный оператор:

*if a>b:*

*print(a)*

*else:*

*print(b)*

Эта программа работает так: есть какие-то две переменные (a, b). Если a больше b, то на экран выводится a, в остальных же случаях (a меньше или равно b) выводится b.

В моей программе также используется обработка исключений, реализующаяся в качестве конструкции *try-except*:

*a = 123*

*b = “123”*

*try:*

*c = a + b*

*except:*

*print(“Произошла ошибка”)*

*else:*

*print(c)*

Это доработка той программы, в которой я показывал работу с переменными. Теперь если в момент попытки сложить первую переменную со второй выскочит ошибка, на экране высветится надпись «Произошла ошибка», если же ошибок не будет, выведется содержимое переменной c, в которую записано сложение a и b.

В программу можно импортировать библиотеки. Библиотеки – это программы на Python, которые можно импортировать в другие программы для использования их функций и классов (комплексных типов данных).

Таким образом, для того, чтобы подключить библиотеку *random* (используется для генерации случайных значений, будь то число, строка, один символ или что-то еще), нужно написать такой код:

*import random*

Чтобы импортировать из библиотеки только что-то конкретное, вместо *import* пишется конструкция *from-import:*

*from random import randint*

Тогда из библиотеки *random* импортируется только функция *randint,* которая производит генерацию псевдослучайных чисел в заданном диапазоне:

*randint(100, 200)*

Данная программа выведет на экран какое-то целое число между 100 и 200 включительно.

**Алгоритм программы**

1. Зайти на веб-сайт колледжа.
2. Зайти на страницу с расписанием на сайте колледжа.
3. Создать рабочую папку, если ее не существует.
4. Загрузить файл расписания в папку.
5. Открыть расписание.

Выглядит довольно просто, особенно если делать это вручную. Все дело в том, что современные ОС и сайт колледжа рассчитаны на использование именно реальными людьми, а не программами. Казалось бы, чего стоит скачать расписание? Всего лишь нажать на кнопку. Но программа не может нажать на кнопку, с точки зрения компьютера сайт – это файл, а не красивая страница. Это и была главная проблема, с которой я столкнулся, - невозможность получить ссылку на расписание стандартным для пользователя путем.

Сначала я изучил закономерности при обновлении расписания и узнал, что ссылка на файл расписания всегда имеет вид «http://rgkript.ru/wp-content/uploads//<год>/<месяц>/<день>.<месяц>.<год>.xls» и обновляется каждую неделю в воскресение. Тогда я подключил библиотеку *datetime*, которая работает с датой и временем. С помощью нее можно хранить данные типа «дата», «время» (с точностью до микросекунд), «промежуток времени» и взаимодействовать с ними.

Алгоритм вычисления ссылки на файл расписания был максимально прост и, как я понял позднее, абсурден: если в настоящий момент воскресение и уже два часа дня (позже двух часов дня, как правило, расписание не обновляется), создается ссылка с сегодняшней датой, в противном случае – ссылка с датой воскресения прошлой недели.

Однако появилось целых два «но»: во первых, если ссылка была сформирована неправильно, сайт никаких сообщений типа «не найдено» не выдает. Хотя если быть точнее, выдает, но это тоже целая страница, которую пришлось бы обрабатывать, причем это не имеет смысла. Во-вторых, в один прекрасный момент к названию файла расписания прибавилось «\_2». Что этот «хвостик» значит, мне неизвестно, и винить тоже некого, т.к. никто стандартов названия этого файла не прописывал. Но как факт – программа не работала. И тогда пришлось прибегнуть к другому варианту – парсингу веб-страницы.

Парсинг – это процесс обработки и анализа какого-то массивного набора данных. В частном случае парсинг веб-сайта – обработка и анализ содержимого этого веб-сайта.

Структура почти каждой веб-страницы среди всех, которые вы видели, выражается через язык разметки HTML. Вдаваться в подробности не будем, но из того, что нужно в нашей программе, важно сказать, что «код» на HTML выглядит как переплетающиеся друг с другом маркеры типа

*<div>*

*<table width = 600>*

*</table>*

*</div>*

(Этот «код» может быть неправильным, т.к. я плохо разбираюсь в устройстве HTML документов)

**Разбор исходного кода программы**

Ниже представлен полный исходный код написанной мной программы:

**import** **requests**

**import** **os**

**import** **time**

**from** **bs4** **import** BeautifulSoup

# Парсинг страницы колледжа, получение ссылки на файл

htmlDoc = requests.get("http://rgkript.ru/raspisanie-zanyatiy/")

soup = BeautifulSoup(htmlDoc.text, "lxml")

url = str(soup.find("table", width=**600**).find("a").get('href'))

# Создание рабочей папки, если она не существует

**try**:

os.mkdir("C:**\\**RaspWorkDir**\\**")

**except** FileExistsError:

time.sleep(**1**)

# Загрузка файла в папку

f = open("C:**\\**RaspWorkDir**\\**raspisanie.xls", "wb")

ufr = requests.get(url)

f.write(ufr.content)

f.close()

# Открытие

os.startfile("C:**\\**RaspWorkDir**\\**raspisanie.xls")

Разбор кода построчно:

**import** **requests**

Импортирует библиотеку *requests*, используемую для получения веб-страниц и загрузки файлов из Интернета;

**import** **os**

Импортирует библиотеку *os*, используемую для работы с ОС и файловой системой компьютера;

**import** **time**

Импортирует библиотеку *time,* используемую для работы с временем (в данном случае лишь для вызова функции ожидания);

**from** **bs4** **import** BeautifulSoup

Импортирует из библиотеки *bs4* метод *BeautifulSoup,* используемый для парсинга HTML документов;

Строки, начинающиеся с «#», называются комментариями. Они ни на что не влияют и несут только поясняющий смысл, и их игнорирует интерпретатор, поэтому их я рассматривать не буду, к тому же, их содержимое говорит само за себя.

htmlDoc = requests.get("http://rgkript.ru/raspisanie-zanyatiy/")

В этой строке в переменную *htmlDoc* записывается содержимое HTML документа, расположенного по адресу «*http://rgkript.ru/raspisanie-zanyatiy/*»;

soup = BeautifulSoup(htmlDoc.text, "lxml")

Здесь в переменную *soup* поступает уже обработанный и готовый к парсингу документ *htmlDoc;*

url = str(soup.find("table", width=**600**).find("a").get('href'))

Эта строка – полная замена первого варианта вычисления ссылки на расписание, который был реализован на *datetime* (кстати, он состоял из 5 очень длинных строчек). Здесь с помощью функции *find* класса *BeautifulSoup* происходит поиск таблицы, в которой расположены на сайте фотографии якобы «расписания», а затем получение из этой таблицы элемента *href*, содержащего искомую ссылку на расписание, и его запись в форме строки (набора символов) в переменную *url*. Таким образом, когда бы пользователь ни запускал эту программу, загрузится именно то расписание, которое в тот момент находилось на сайте, т.е. самое свежее;

**try**:

os.mkdir("C:**\\**RaspWorkDir**\\**")

**except** FileExistsError:

time.sleep(**1**)

Эту конструкцию сложно объяснять построчно. Здесь происходит попытка создать директорию *C:\RaspWorkDir\.*Казалось бы, почему нельзя проверить наличие этой папки с помощью условного оператора, но, как оказалось, в библиотеке *os* отсутствует нужная функция, с помощью которой можно проверить ее наличие, а вместо нее при попытке создать директорию, которая уже существует, возникает необработанное исключение *FileExistsError*, которое можно отловить. В результате работы этой конструкции создается папка *C:\RaspWorkDir\,*независимо от того, была она или нет;

f = open("C:**\\**RaspWorkDir**\\**raspisanie.xls", "wb")

Происходит создание/открытие файла в той самой папке с режимом перезаписи с привязкой к переменной *f;*

ufr = requests.get(url)

По ссылке из переменной *url* с помощью функции загрузки файлов из интернета *requests.get* в переменную *ufr* загружается расписание;

f.write(ufr.content)

Через переменную *f* из переменной *ufr* расписание наконец записывается в файловую систему;

f.close()

Закрытие файла;

os.startfile("C:**\\**RaspWorkDir**\\**raspisanie.xls")

Последний штрих: программа посылает операционной системе сигнал запустить файл так, будто пользователь его включил сам.

В результате работы программы на диске C появляется директория «RaspWorkDir», в которой есть файл «raspisanie.xls», содержащий последнее расписание, который к тому же сразу запускается программой, установленной в системе по умолчанию для работы с xls файлами.

**Дистрибуция**

Еще одна проблема, с которой я столкнулся, заключается в том, что программа в исходном виде представлена как файл с расширением «.py», исполняемый только самим интерпретатором.

Это значит, что, если бы я хотел с кем-нибудь поделиться, этому кому-нибудь пришлось бы тоже устанавливать Python себе на компьютер.

Тогда на помощь пришла утилита PyInstaller. Это отдельная программа, которая устанавливается в пространство имен командной строки Windows, т.е. ее можно вызывать как команду. Она собирает ядро Python, указанный файл программы и используемые в ней библиотеки в исполняемый файл с расширением «.exe» (и, возможно, несколько системных файлов), который будет работать на любом компьютере с ОС Windows.

Синопсис команды PyInstaller:

*PyInstaller [теги] <полный\_адрес\_до\_файла\_программы>*

*Теги* – это приставки к команде, характеризующие и уточнающие ее поведение. Например, тег «-F» настраивает программу так, чтобы все собиралось в ТОЛЬКО ОДИН исполняемый файл, т.е. в него будут вложены и системные файлы, и библиотеки; тег «-w» выключает консоль программы (для их тихой работы или для удобства, если в них есть графический интерфейс) и т.д.

Мне оставалось только собрать «.exe» файл.

**Вместо заключения: возможные ошибки и перспективы развития проекта**

Программа может не работать, это нормально.

Если она не работает, вот рекомендации к исправлению:

- Проверьте интернет-соединение;

- Добавьте программу в список исключений Microsoft Firewall;

- Добавьте программу в список исключений вашего антивируса;

- Проверьте, есть ли у программы разрешение на запись в файловой системе.

Возможно, у вас не сможет открыться файл, тогда вам нужно установить программу, которая способна читать электронные таблицы, например, Microsoft Excel.

Что в принципе стоит добавить:

- Полную автономность, при которой программа функционировала бы все время работы компьютера, и постоянно проверяла бы изменения в расписании, причем, если расписание изменилось, программа послала бы уведомление;

- Графический интерфейс, в котором будет отображаться расписание нужной группы в нужную неделю, причем и то, и другое можно будет выбрать;

- Содействие с системными администраторами колледжа, ведь расписание – это не все, что удобно было бы иметь при себе, есть перспектива разработки целого приложения, которое можно было бы использовать в целях информатизации студентов, быстрой коммуникации, облегчения передачи объявлений и т.д.

**Список использованных источников и литературы**

1. Доусон Майкл «Программируем на Python» - перевод 3-его изд. 2018 г.
2. Пол Бэрри «Изучаем программирование на Python» - 2-е изд. 2017 г.
3. Официальный сайт Python <https://www.python.org/>
4. Официальный сайт ГБПОУ РО «РКРИПТ» <http://rgkript.ru/>
5. Репозиторий проекта на GitHub (оттуда можно загрузить исходный код и исполняемый файл для повседневного использования) <https://github.com/HeadlessNick1352/RKRIPTraspisanie>