**Содержание урока:**

* Введение
* Пушим репозиторий
* Знакомство с понятие API
* ЦБРФ API
* Библиотека datetime и словари
* Самостоятельная работа
* Заключение

**Введение**

Сегодня познакомимся с понятием API, поработаем с ресурсами центробанка РФ. Узнаем как как работать с датой и временем в Python и как сайт отвечает на различные запросы. Научимся считывать и изменять файлы в python и заставим наш компьютер разговаривать

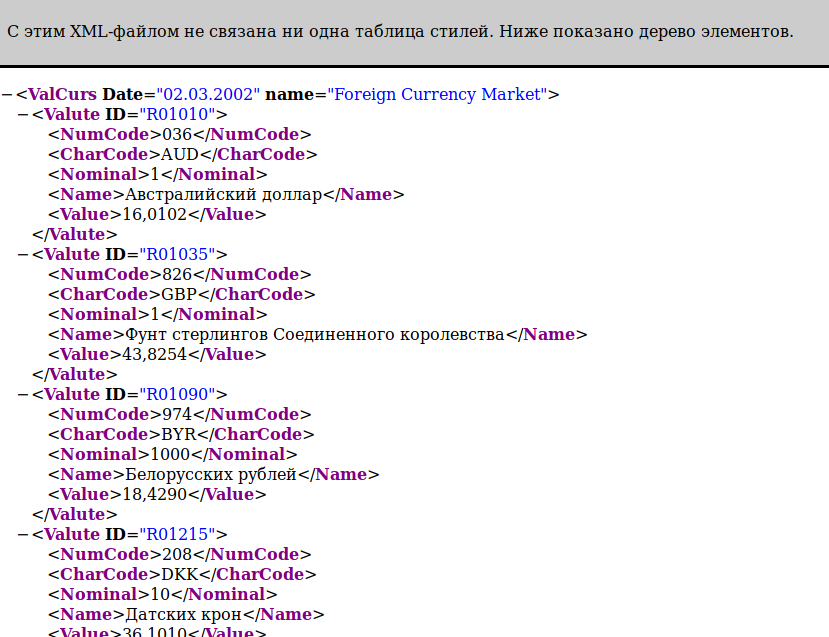
**1-й урок**

На прошлом уроке мы с вами вытягивали данные с интернета при помощи парсинга, сегодня же мы с вами поработает с API. Дело в том, что в случае изменения дизайна сайта который мы парсим, наша программа-парсер перестанет работать. API же не принятно менять и если его меняют, то старую версию ещё некоторое время поддерживают, чтобы программа не сломалась в один момент. Что же такое API, это application programming interface. Это программный интерфейс для управления чем-то. В реальной жизни мы постоянно сталкиваемся с похожими интерфейсами. По своей сути руль и коробка передач, это тоже интерфейс который позволяет нам отправлять запросы машине и заставлять её что-то делать. В нашем случае мы отправляем запросы на сайт и в зависимости от наших запросов получаем какой-то ответ. Конкретно сейчас это простой пример API, в дальнейшем мы будем работать с более сложными. Мы уже сейчас работали с более сложными, дело в том, что библиотеки по своей сути тоже является API.

У сайта цб РФ, есть своё api, мы делает какой-то запрос и получаем ответ в виде xml таблицы. Сейчас мы попробуем вытащить с него курс на 02.03.2002.

Давайте перейдём по следующей ссылке:

[**http://www.cbr.ru/scripts/XML\_daily.asp?date\_req=02/03/2002**](http://www.cbr.ru/scripts/XML_daily.asp?date_req=02/03/2002)



В отличие от того что мы парсили на прошлом уроке, это намного проще, тут всё логично, нет лишнего мусора. Мы видим что за один австралийский доллар, в 2002 году давали 16 рублей.

Давайте обратим внимание на ссылку, мы видим data\_req=02/03/2002, если мы изменим 2002, на 2015 то увидим курс на 28.02.2015, как мы видим курс поменялся. Теперь исходя из этого мы можем узнать курс на сегодня для евро, долларов и китайских юаней. Самостоятельная работа.

* Создайте папку third\_lesson c файлом course.py
* При помощи requests сделайте запрос на текущую дату
* При помощи bs4 распарсите xml таблицу с текущим курсом (Название всех тегов указываются с маленькой буквы) Вспомните как мы парсили время восхода и заката, оно тоже было вложенным
* Выведите курс

Ответ

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  url = "http://www.cbr.ru/scripts/XML\_daily.asp?"  date = "date\_req=28/08/2018"  responce = requests.get(url + date)  xml = BeautifulSoup(responce.content, "lxml")  def getCourse (id):  return xml.find("valute", {'id': id}).value.text  print(getCourse("R01235"), "рублей за 1 доллар")  print(getCourse("R01239"), "рублей за 1 евро")  print(getCourse("R01375"), "рублей за 10 юаней") |

Вроде неплохо, но надо доработать

Задание 6

|  |
| --- |
| from datetime import datetime |

Подключим библиотеку, для того чтобы узнавать какое сейчас число и не вбивать его каждый раз по новой.

Задание 7

|  |
| --- |
| today = datetime.today()  today = today.strftime("%d/%m/%Y") |

Объявим переменную today, в которой будем хранить сегодняшнее число и следом перезапишем её, отформатировав число так, как нам нужно. В формате дата/месяц/год.

Обратите внимание, на записи в фигурных скобках, который мы передаём функции find. Это называется словарём, словарь это ещё один тип данных в python, нам нужен будет словарь для того чтоб передать аргументом в функцию get

Задание 8

|  |
| --- |
| payload = {"date\_req" : today} |

Немного про словари. Словарь — неупорядоченная структура данных, которая позволяет хранить пары «ключ — значение». Вот пример словаря на Python:

Вывести на экран

|  |
| --- |
| dictionary = {'персона': 'человек',  'марафон': 'гонка бегунов длиной около 26 миль',  'противостоять': 'оставаться сильным, несмотря на давление',  'бежать': 'двигаться со скоростью'} |

Данный словарь использует строки в качестве ключей, однако ключом может являться в принципе любой неизменяемый тип данных. Значением же конкретного ключа может быть что угодно. Вот ещё один пример словаря, где ключами являются числа, а значениями — строки:

Вывести на экран

|  |
| --- |
| ender\_dict = {0: 'муж',  1: 'жен'} |

Вывести на экран

|  |
| --- |
| print(dictionary["марафон"]) |

Тут мы получили значение по ключу марафон. Мы будем использовать словарь, потому что это один из аргументов функции get, то что мы раньше складывали строки, это не совсем круто.

Передаём наш словарь в функцию get

Задание 9

|  |
| --- |
| responce = requests.get(url, params=payload) |

Давайте посмотрим, по какой ссылке мы переходим

Задание 10

|  |
| --- |
| print(responce.url) |

**2-й урок**

**Введение**

Мы научились вытягивать данные из XML таблиц центробанка, давайте теперь научимся что-то записывать. Дело в том что как только вы закрываете программу, все данные с которыми она работала удаляются. Есть много способом их сохранить, давайте разберём один из таких. Это запись в текстовый документ.

Перед тем как прочесть и записать что-либо в файл, его следует открыть. Чтобы открыть файл в Python используется встроенная функция open(). При вызове, эта функция создает объект типа файл, с которым в дальнейшем можно работать.

Давайте с вами попробуем считать все доступные курсы валют и структурировано их вывести а после этого записать куда-то

Задание 11

|  |
| --- |
| my\_file = open("file.txt", "a") |

В функцию мы передаём 2 параметра, название файла в который мы хотим, что-то записать и флаг. Если с название файла всё понятно, то с флагом придётся разобраться. Государство, количество, революция, что общего у этих слов? Нет, это не связано с историей, эти слова удобно сокращать гос-во, кол-во, рев-ция. Тоже самое и с флагами, флаг — это сокращённая запись слова, в данном случае команды. Флаг — это некоторая контрольная отметка, с помощью которой программист управляет логикой работы программы. Может быть переменной, или даже отдельным битом - смотря сколько требуется вариантов значений. Классический пример флагов — это переменные типа boolean (TRUE/FALSE). Программа проверяет значение переменной, и действует в зависимости от того, какое значение было установлено - либо пользователем, либо самой программой в процессе выполнения.

Есть флаг r, флаг r говорит, что мы собираемся читать содержимое файла. Есть флаг а, флаг a говорит, что мы собираемся изменять содержимое файла.

Задание 12

|  |
| --- |
| my\_file.write("Текст") my\_file.close() |

Мы можем записать, что-то в файл если используем функцию write и передадим ей аргументом строку. Только, чтобы изменения сохранились, мы должны будем закрыть файл при помощи функции close().

Помимо того, что мы можем туда записать, мы можем прочитать всё что мы туда запишем.

Задание 13

|  |
| --- |
| my\_file = open("file.txt", "r")  my\_text = my\_file.read()  print(my\_text)  my\_file.close() |

Мы делаем это при помощи функции read() а результат записываем в отдельную переменную.

**3-й урок**

Давайте теперь воспользуемся мощными инструментами на python, чтобы прочитать и воспроизвести текст из текстового документа робоголосом.

Нам пригодятся для этого 3 библиотеки:

1. OS — нужна для работы с операционной системой, работы с файловыми директориями. Удаления файлов и прочих разных штук.
2. Pygame — библиотека для создания игр на python, мы не будем создавать игры. Но мы будем использовать её инструменты для записи данных в mp3 файл
3. gtts — инструменты Google для озвучивания текста

Задание 14

|  |
| --- |
| import os  from pygame import mixer  from gtts import gTTS |

Из библиотеки pygame нам пригодится только модуль mixer, из библиотеки gtts только gTTS. Давайте теперь запишем название и расширение файла в который мы будем записывать голос и откроем текстовый документ, из которого мы будем читать текст, и озвучивать его.

Задание 15

|  |
| --- |
| my\_file = open("some.txt","r")  my\_string = my\_file.read()  my\_file.close() |

Ну а теперь озвучим, мы отправляем в google текст который записан у нас в переменную и укажем язык текста. То, что нам вернёт google, записываем в переменную и инициализируем модуль mixer

Задание 16

|  |
| --- |
| mixer.init()  tts=gTTS(text=my\_string, lang='ru') |

Затем сохраняем в наш файл то, что google передал нам в переменную tts. Загружаем этот файл и запускаем. После этого вызываем команду remove и удаляем аудио файл чтоб не загрязнять директорию. Запускаем

Задание 17

|  |
| --- |
| tts.save(mp3\_name)  mixer.music.load(mp3\_name)  mixer.music.play()  os.remove(mp3\_name) |

Мы озвучили текст из файла, можно немного допилить и сделать из неё процедуру (функция которая не возвращает значение), которая принимает текст, который надо озвучить.

**Домашнее задание**

Давайте теперь попробуем самостоятельно создать скрипт, который будет конвертировать валюту.

* У нас есть главная функция, она принимает 3 аргумента
* valute\_from и её требуется перевести в валюту valute\_to через рубль (код: RUR)
* amount, сумма денег, которую мы хотим конвертировать
* Для начала мы находим значения valute\_to, обратите внимание, что номинал может быль больше 1
* Затем условие, если валюта, из которой мы конвертируем это RUR, мы просто парсим курс
* Если эта валюта !RUR, то мы узнаём её курс исходя из курса рубля и делим amount на курс валюты, в которую переводим

Задание 18

|  |
| --- |
| valute\_from = "EUR"  valute\_to = "USD"  amount = 800  def course (valute\_from, valute\_to, amount):  … |

**Заключение:**

Мы с вами познакомились с тем что такое api, при помощи средств центробанка, узнали про новый тип данных, познакомились с новой библиотекой и решили довольно сложную логическую задачу на перевод валют. В следующем раз, мы с вами сделаем графическую оболочку для того что мы написали на этом уроке.