Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств
вычислительной техники и автоматизированных систем"

ОТЧЕТ

по учебной практике
на кафедре Прикладной Математики и Кибернетики

Выполнил:

студент гр. ИП-111 Гердележов Даниил Дмитриевич

ФИО студента

«10» мая 2023г.

Руководитель практики

доцент каф. ПМиК _____/Приставка П.А./

«15» мая 2023г. Оценка_____.

Новосибирск 2023 г.

Оглавление

Задание на учебную практику:

Разработать программу, реализующую ввод, хранение и обработку данных о котировках криптовалют на основе данных сайта coinmarketcap.com.

Общие требования к программе:

1. Язык разработки: Python версии не ниже 3.x
2. Операционная система: определяются студентом
3. Набор свойств криптовалют:
   - Name – наименование
   - Symbol
   - Price – стоимость 1 ед. в долларах США (USD)
   - Market_cap – рыночная капитализация

` 4. Ввод данных

Оценки «хорошо» и «удовлетворительно»	Оценка «отлично»
Из файла currencies23.csv. Файл содержит данные о 100 наиболее ценных криптовалютах на 19.02.2023 в формате: Name, Symbol, Price, Market_cap Файл доступен для скачивания в ЭИОС в директории с заданием на практику.	Непосредственно с главной страницы сайта coinmarketcap.com в момент запуска программы. Загрузка и парсинг веб-страницы производится с помощью библиотек Requests и Beautifulsoup или их аналогов Примечание: допускается считывание строчек в количестве менее 100 (Например, 10 строчек с данными о криптовалютах)

5. Хранение

Типы и структуры для хранения данных: определяются студентом

6. Обработка

Реализовать функцию поиска информации о свойствах криптовалюты по ее названию.

Описание алгоритмов.

Общая структура программы:

1. Загрузка с сайта информации о именах классов, которые содержат необходимые нам данные, функция: find_class_names().

   Потребность в получении этих данных возникает из-за того, что имена классов на сайте периодически изменяются.
2. Инициализация интерфейса (создание окна, кнопок, полей ввода и вывода информации).
3. Ожидание нажатия пользователем кнопок.

   На выбор предоставляется два различных способа парсинга данных: на основе библиотеки **requests** (получает 10 строк, работает довольно быстро) и на основе **selenium** (получает всю таблицу, но работает дольше чем request).
4. После выбора пользователем способа получения информации в отдельном потоке (для предотвращения зависания программы) начинается сбор и обработка информации.
5. **req_pars()** – сохранят код страницы, который передает в функцию data_processing(soup) для последующей обработки.

   sel_pars() – создает экземпляр браузера и ждет полной загрузки всех элементов страницы, после чего пошагово пролистывает страницу до самого низа для сбора всей информации из таблицы (это необходимо из-за того, что информация о криптовалютах подгружается на веб-странице по мере пролистывания). После сбора данных функция также передает код страницы в функцию data_processing(soup), которая с помощью библиотеки BeautifulSoup обрабатывает полученные данные.
6. Обработка данных.

   Обработка данных происходит в функции data_processing(soup), с помощью библиотеки BeautifulSoup. Для получения нужных

данных используется информация об именах классов в которых хранятся данные (эту информацию мы получили в п. 1 с помощью функции find_class_names() и библиотек BeautifulSoup и requests). Полученные данные хранятся в глобальной переменной cl_table, которая представляет собой список.

Алгоритмы работы каждой функции:

Алгоритм работы функции find_class_names():

1. Отправляется GET-запрос на веб-страницу и ответ сохраняется во временную переменную r.
2. Используя объект BeautifulSoup, функция разбирает HTML-код веб-страницы, сохраняя результат в переменную soup.
3. Затем, с помощью метода prettify() объекта soup, она получает форматированный HTML-код и сохраняет его в переменную code.
4. Функция задает глобальные переменные name_class, symbol_class и cap_class.
5. С помощью методов find() и slicing, функция ищет и сохраняет в переменную name_class имя CSS-класса для имени криптовалюты.
6. Аналогично, функция ищет и сохраняет в переменную symbol_class имя CSS-класса для символа криптовалюты.
7. Аналогично, функция ищет и сохраняет в переменную cap_class имя CSS-класса для рыночной капитализации криптовалюты.

Алгоритм работы функции req_parsing():

1. Отправляется GET-запрос на веб-страницу и ответ сохраняется во временную переменную r.
2. Используя объект BeautifulSoup, она разбирает HTML-код веб-страницы, сохраняя результат в переменную soup.
3. Затем функция вызывает функцию data_processing(), передавая объект soup в качестве аргумента.

Алгоритм работы функции sel_parsing():

1. Создается объект webdriver.EdgeOptions() и добавляются опции для запуска браузера в headless режиме и отключения использования GPU.
2. Создается объект webdriver.Edge с драйвером для Microsoft Edge и опциями, созданными на предыдущем шаге.
3. Вызывается метод get() для загрузки страницы с URL_TEMPLATE в веб-браузере.
4. Прокручиваем страницу постепенно до ее конца, чтобы получить доступ ко всем элементам, используя цикл while и методы execute_script() и scrollTo() объекта драйвера.
5. Создается объект BeautifulSoup, используя текущее содержимое страницы, полученное с помощью метода page_source объекта драйвера.
6. Закрывается веб-браузер, вызывая метод close() объекта драйвера.
7. Вызывается функция data_processing(), передавая объект soup в качестве аргумента.

Алгоритм работы функции sel_parsing():

1. Функция `data_processing` принимает в качестве аргумента объект `soup`, представляющий собой синтаксическое дерево

HTML-кода, полученного при помощи библиотеки BeautifulSoup из ответа на запрос.

2. Внутри функции создается список `data` размерностью 100x4, который будет заполнен данными об имени криптовалюты, ее символьном обозначении, текущей цене и капитализации.
3. Затем функция находит таблицу на странице при помощи метода `find` объекта `soup` и извлекает из нее все строки (`tr`) при помощи метода `find_all`.
4. Для каждой строки функция пытается извлечь информацию об имени криптовалюты, символьном обозначении, цене и капитализации, используя метод `find` для поиска соответствующих элементов (`p` или `span`) с помощью их CSS-классов (`name_class`, `symbol_class` и `cap_class`). Если какие-то из данных отсутствуют или извлечь их не удалось, соответствующие поля в массиве `data` останутся пустыми.
5. После обработки всех строк таблицы массив `data` записывается в глобальную переменную `cl_table`.

Листинг программы.

main.py

```
import tkinter as tk
import threading
from pars import sel_parsing, req_parsing, data_search, find_class_names


class MyApp:
    def __init__(self, master):
        find_class_names()
        self.master = master
        self.master.title("ИП-111 Гердележов Д.Д.")
        self.master.geometry("450x250")

        self.button_frame = tk.Frame(self.master)
        self.button_frame.pack()

        self.button1 = tk.Button(
            self.button_frame, text="Request (10 строк)", command=lambda:
self.run_in_thread(req_parsing))
        self.button1.pack(side="left", padx=10, pady=10)

        self.button2 = tk.Button(
            self.button_frame, text="Selenium (Вся таблица)", command=lambda:
self.run_in_thread(sel_parsing))
        self.button2.pack(side="left", padx=10, pady=10)

        self.entry = tk.Entry(self.master, state="disabled")
        self.entry.pack(pady=10)

        self.search_button = tk.Button(self.master, text="Поиск",
state="disabled",
                                       command=lambda:
self.run_in_thread(self.search, self.entry.get()))
        self.search_button.pack()

        self.result_text = tk.Text(self.master, height=10, state="disabled")
        self.result_text.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

    def run_in_thread(self, func, *args):
        if func == req_parsing:
            self.result_text.configure(state="normal")
            self.result_text.insert(1.0, "Подождите...\n")
            self.result_text.configure(state="disabled")
            self.button2.configure(state='disabled')
        elif func == sel_parsing:
            self.result_text.configure(state="normal")
            self.result_text.insert(1.0, "Подождите...\n")
```

```
            self.result_text.configure(state="disabled")
            self.button1.configure(state='disabled')

        self.search_button.configure(state='disabled')
        self.entry.configure(state='disabled')

        thread = threading.Thread(target=func, args=args)
        thread.start()

        while thread.is_alive():
            self.master.update()
        else:
            if func == req_parsing:
                self.button2.configure(state='normal')
            elif func == sel_parsing:
                self.button1.configure(state='normal')
            self.enable_entry()

    def req(self):
        thread = threading.Thread(target=req_parsing)
        thread.start()
        thread.join()
        self.button2.configure(state='normal')
        self.enable_entry()

    def sel(self):
        thread = threading.Thread(target=sel_parsing)
        thread.start()
        thread.join()
        self.button1.configure(state='normal')
        self.enable_entry()

    def search(self, name):
        result = data_search(name)
        out = ''
        for i in result:
            out += str(i) + " "

        self.result_text.configure(state="normal")
        self.result_text.insert(1.0, out + '\n')
        self.result_text.configure(state="disabled")

        self.entry.delete(0, tk.END)

    def enable_entry(self):
        self.entry.configure(state="normal")
        self.search_button.configure(state="normal")
```

```
root = tk.Tk()
app = MyApp(root)
root.mainloop()
```

pars.py

```
import requests
from selenium import webdriver
from bs4 import BeautifulSoup as bs
from time import sleep

URL_TEMPLATE = "https://coinmarketcap.com/"

cl_table = []
name_class = ''
symbol_class = ''
cap_class = ''


def data_search(name):
    global cl_table
    for i in cl_table:
        if str(name).lower() == str(i[0]).lower() or str(name).lower() ==
str(i[1]).lower():
            return i

    return (['Введено неверное название'])


def find_class_names():
    r = requests.get(URL_TEMPLATE)
    soup = bs(r.text, "html.parser")
    code = soup.prettify()

    global name_class, symbol_class, cap_class
    code = code[code.find("name-area"):]
    name_class = code[code.find("name-area"):code.find("Bitcoin")]
    name_class = name_class[name_class.find("="):name_class.find(" color")]
    name_class = name_class[2:-1]

    symbol_class = code[code.find(
        "</div>"):code.find("coin-item-symbol") + 1 + len("coin-item-symbol")]
    symbol_class = symbol_class[symbol_class.find("=") + 2:-1]

    cap_class = code[code.find("cmc-link"):]
    cap_class = cap_class[:cap_class.find('data-nosnippet="true"')]
    cap_class = cap_class[-50:]
    cap_class = cap_class[cap_class.find('"') + 1:-2]
```

```
def data_processing(soup):
    global name_class, symbol_class, cap_class
    data = [[0]*4 for i in range(100)]

    table = soup.find('tbody')
    rows = table.find_all("tr")

    count = 0
    for row in rows:
        try:
            data[count][0] = str(
                (row.find("p", class_=name_class)).text)

            data[count][1] = str(
                (row.find("p", class_=symbol_class)).text)

            price_code = row.find_all("a", class_="cmc-link")
            for i in price_code:
                try:
                    price = i.find('span').text
                    data[count][2] = str(price)
                except:
                    pass

            data[count][3] = str(
                (row.find("span", class_=cap_class)).text)
        except:
            pass

        count += 1

    global cl_table
    cl_table = data


# Получает только 10 строк
def req_parsing():
    r = requests.get(URL_TEMPLATE)
    soup = bs(r.text, "html.parser")

    data_processing(soup)


# Получает всю таблицу, однако выполняется дольше
def sel_parsing():
    options = webdriver.EdgeOptions()
    options.add_argument('headless')
    options.add_argument('--disable-gpu')
    driver = webdriver.Edge("./msedgedriver.exe", options=options)
```

```
driver.get(URL_TEMPLATE)

max_height = 0
scroll_height = 1000

# Прокручиваем страницу до конца
while True:
    # Получаем текущую позицию
    current_position = driver.execute_script("return window.pageYOffset;")

    # Прокручиваем страницу на scroll_height пикселей вниз
    driver.execute_script(
        f"window.scrollTo(0, {current_position + scroll_height});")
    sleep(1)

    # Проверяем, достигли ли мы конца страницы
    new_scroll_height = driver.execute_script(
        "return document.body.scrollHeight")
    if new_scroll_height == max_height:
        break
    max_height = new_scroll_height

soup = bs(driver.page_source, 'html.parser')

driver.close()

data_processing(soup)
```

Результаты тестирования.

После запуска программы поле ввода и кнопка “Поиск” заблокированы.

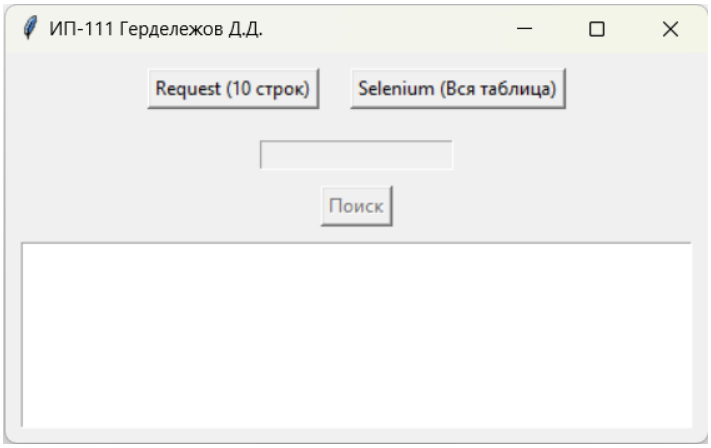

Рис. 1 – Окно программы после запуска.

На время выполнения парсинга данных деактивируются все кнопки кроме нажатой, в окно вывода информации выводится сообщение с просьбой подождать окончания.

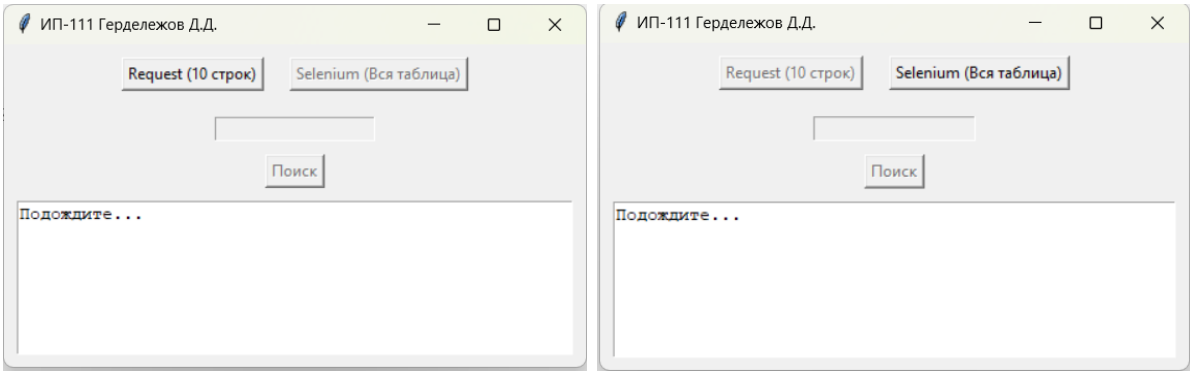

Рис. 2 – Окно программы во время выполнения функций сбора и обработки данных.

После завершения парсинга данных поле ввода и кнопка “Поиск” становятся активными. Пользователь может ввести название криптовалюты или её символьное обозначение для получения информации. Информация выводится ниже, в поле вывода.

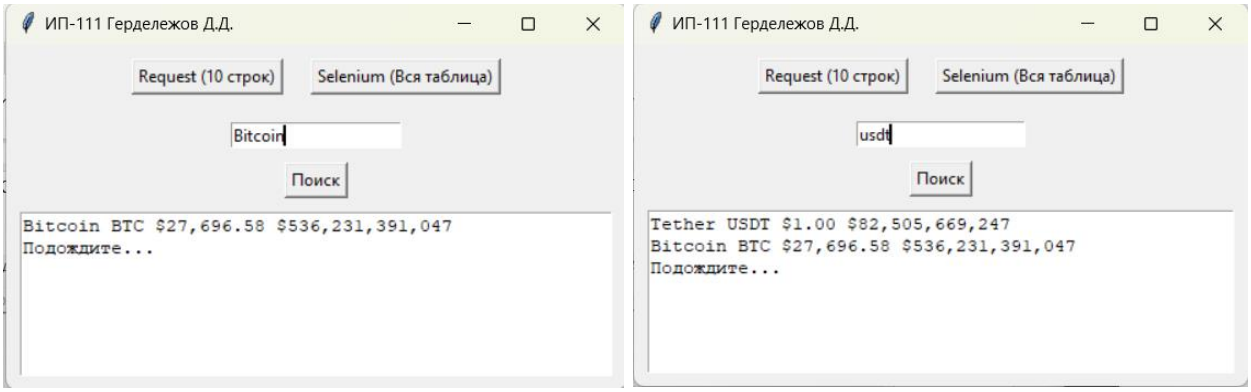

Рис. 3 – пример поиска информации о криптовалюте.

При вводе названия криптовалюты информация о которой отсутствует в таблице в поле вывода появится информация об ошибке. (в данном примере я получал только 10 строк таблицы, поэтому информации о криптовалюте EOS нет в таблице с данными (на момент написания отчета она находится на 44 месте)).

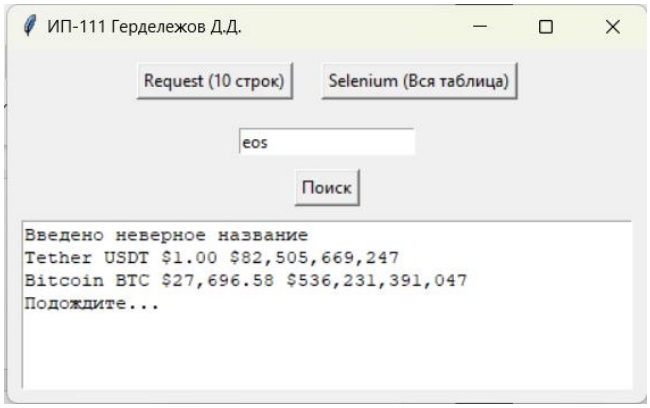

Рис. 4 – Результат работы программы в случае ввода неизвестного названия криптовалюты.

В любой момент можно обновить сохранненные программой данные и/или получит их другим способом. Для этого достаточно нажать на соответсвующие кнопки (как при старте прогаммы). Для демонстрации получу полные данные и запрошу информацию о криптовалюте EOS (аналогично рисунку 4).

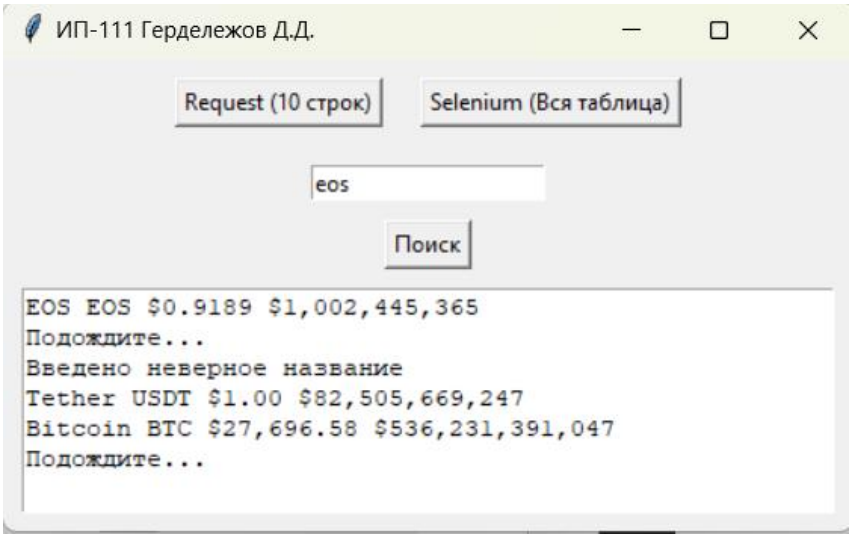

Рис. 5 – результат работы программы после обновления (или получения новой) информации.

Список источников.

1. Электронный учебник *«Самоучетель python»* [электронный ресурс] // https://pythonworld.ru/samouchitel-python
2. *Сузи Р.А.* ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON. УЧЕБНОЕ ПОСОБИЕ // учебное пособие Издательство: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020.
3. Документация библиотеки «Requests» [электронный ресурс] // https://requests.readthedocs.io/en/latest/
4. Документация библиотеки «Selenium» [электронный ресурс] // https://selenium-python.readthedocs.io/
5. Документация библиотеки «BeautifulSoup» [электронный ресурс] // https://www.crummy.com/software/BeautifulSoup/bs4/doc/
6. Документация библиотеки «tkinter» [электронный ресурс] // https://docs.python.org/3/library/tk.html
7. РАБОТА С ВЕБ-ДАННЫМИ С ПОМОЩЬЮ REQUESTS И BEAUTIFUL SOUP В PYTHON [электронный ресурс] // https://www.8host.com/blog/rabota-s-veb-dannymi-s-pomoshhyu-requests-i-beautiful-soup-v-python-3/