**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КАФЕДРА САПР**

**Практическая работа №1**

**по дисциплине**

**«Объектно-ориентированное программирование»**

**Тема: «Консольная программа для поиска в Википедии»**

|  |  |  |
| --- | --- | --- |
| Студентки гр. 3352 |  | Парфенова Ю.А.  Нижейко Е.И. |
| Преподаватель |  | Кулагин М.В. |

Санкт-Петербург

2024

1. **Задание**

Напишите программу, которая с консоли считывает поисковый запрос пользователя, и выводит результат поиска по Википедии. После выбора нужной статьи программа должна открывать ее в браузере. Программа должна реагировать корректно на любой пользовательский ввод.

Задача разбивается на 5 этапов:

1. Считать введенные пользователем данные;
2. Сделать запрос к серверу;
3. Распарсить ответ;
4. Вывести результат поиска;
5. Открыть нужную страницу в браузере.

Использовать готовые библиотеки для работы с Википедией нельзя.

1. **Спецификация программы**

Для выполнения этой задачи был выбран объектно-ориентированный язык программирования Python.

Программа представляет собой консольное приложение, которое позволяет искать информацию в Википедии с помощью API. Пользователь может вводить поисковые запросы, просматривать результаты и открывать выбранные статьи в веб-браузере.

**Первый класс** — WikiSearch:

**Описание:** Класс WikiSearch предназначен для поиска информации на Википедии и обработки результатов поиска. Он использует библиотеку requests для отправки HTTP-запросов к API Википедии, а также библиотеки langdetect и BeautifulSoup для определения языка поискового запроса и очистки HTML-разметки соответственно.

**Атрибуты:**

1. self.query- Этот атрибут хранит строку, содержащую поисковый запрос, который был передан конструктору при создании экземпляра класса.

2. self.results- Это список, который изначально пуст. После успешного выполнения метода search() он заполняется результатами поиска, извлечёнными из ответа API Википедии.

**Методы:**

1. \_\_init\_\_(self, query)

Это конструктор класса, который инициализирует объект WikiSearch. Принимает аргумент query — строку, представляющую поисковый запрос. Внутри конструктора создаются два атрибута:

- self.query: сохраняет поисковый запрос.

-self.results: пустой список, предназначенный для хранения результатов поиска.

2. detect\_language(self)

Этот метод автоматически определяет язык поискового запроса с помощью функции detect из библиотеки langdetect. Если определение языка проходит успешно, возвращается соответствующий код языка (например, 'ru' для русского). В случае ошибки возвращается значение по умолчанию — 'en'.

3. search(self)

Основной метод, отвечающий за выполнение поиска на Википедии. Последовательность действий:

- Определяет язык поискового запроса с помощью метода detect\_language.

- Формирует URL для обращения к API Википедии в зависимости от языка:

- Для русского языка: "https://ru.wikipedia.org/w/api.php"

- Для остальных случаев: "https://en.wikipedia.org/w/api.php"

- Отправляет GET-запрос с необходимыми параметрами:

- "action": "query" — указывает, что выполняется запрос.

- "list": "search" — сообщает, что нужно выполнить поиск.

- "srsearch": self.query — передает поисковую фразу.

- "format": "json" — запрашивает результат в формате JSON.

- "srlimit": 10 — ограничивает количество возвращаемых результатов до 10.

- "utf8": 1 — включает поддержку UTF-8.

- Преобразует полученный ответ в JSON и проверяет наличие ключа "query" и вложенного ключа "search". Если ключи найдены, результаты поиска сохраняются в атрибут self.results.

4. get\_results(self)

Метод возвращает результаты поиска в удобном формате. Он создает список кортежей, где каждый кортеж содержит пару значений:

- Название статьи (result["title"])

-Очищенное от HTML-тегов описание статьи (self.clean\_html(result["snippet"]))

Очистку HTML осуществляет метод clean\_html.

5. clean\_html(self, raw\_html)

Этот вспомогательный метод принимает сырой HTML-код и удаляет из него все теги, оставляя только текст. Используется библиотека BeautifulSoup для парсинга HTML. Очищенный текст возвращается методом soup.get\_text().

Каждый из этих методов играет свою роль в процессе поиска и обработки данных с Википедии, обеспечивая удобство и гибкость использования класса.

**Второй класс** — GetInfo:

**Описание:** Класс GetInfoслужит для организации взаимодействия между пользователем и поиском информации на Википедии. Он объединяет функционал классов ConsoleWorkи WikiSearch, предоставляя пользователю возможность вводить запросы, просматривать результаты и открывать выбранную статью в браузере.

**Атрибуты:**

- self.searcher— хранит объект класса WikiSearch, используемый для поиска статей на Википедии.

- self.console— объект класса ConsoleWork, обеспечивающий взаимодействие с консолью.

**Методы:**

1. \_\_init\_\_(self)

Конструктор класса, который инициализирует два атрибута:

- self.searcher устанавливается в None.

- self.console создается как объект класса ConsoleWork.

2.process(self)

Основной метод, управляющий процессом поиска и вывода информации. Последовательность шагов:

1. Запрашивается ввод поискового запроса у пользователя с помощью метода get\_query() объекта self.console.

2. Создается объект класса WikiSearch с переданным запросом и сохраняется в self.searcher.

3. Выполняется поиск с помощью метода search() объекта self.searcher.

4. Результаты поиска извлекаются методом get\_results() и сохраняются в переменную results.

5. Если результаты найдены:

- Они выводятся на экран с помощью метода show\_results() объекта self.console.

- Пользователь выбирает одну из найденных статей с помощью метода get\_selection().

- Из выбранного результата извлекаются название статьи (title) и краткое описание (snippet).

- Формируется URL статьи на Википедии в зависимости от языка (ru или en), заменяя пробелы на \_.

- Статья открывается в браузере с помощью функции webbrowser.open().

6. Если результаты не найдены, выводится сообщение об этом.

Класс GetInfo обеспечивает полный цикл работы с Википедией через консольное приложение:

1. Получение ввода от пользователя.

2. Поиск информации на Википедии.

3. Отображение результатов поиска.

4. Открытие выбранной статьи в браузере.

Такой подход делает использование Википедии удобным и интуитивным даже в командной строке.

**Третий класс** — ConsoleWork:

**Описание:** Класс ConsoleWork предназначен для управления взаимодействием с пользователем через консоль. Он включает методы для получения пользовательского ввода, отображения результатов поиска и выбора нужной статьи.

**Методы:**

1.get\_query(self)

Этот метод запрашивает у пользователя ввод поискового запроса для Википедии. Выводится сообщение "Введите поисковый запрос в Wikipedia:", после чего ожидается ввод строки от пользователя. Введенная строка возвращается как результат метода.

2.show\_results(self, results)

Метод выводит на экран список найденных статей. Параметр results— это список кортежей, где каждый кортеж содержит название статьи и ее краткий текстовый фрагмент (сниппет). Каждая статья нумеруется начиная с 1, и рядом с номером указывается название статьи и ее описание.

3. get\_selection(self, max\_selection)

Метод запрашивает у пользователя выбор одной из предложенных статей. Пользователю предлагается ввести номер статьи в диапазоне от 1 до max\_selection. Если введен недопустимый номер (не целое число или выходит за пределы допустимого диапазона), выводится соответствующее сообщение об ошибке, и запрос повторяется. Когда вводится правильный номер, метод возвращает индекс выбранной статьи (индексация начинается с нуля).

Класс ConsoleWork используется для создания простого интерфейса командной строки, позволяющего пользователю взаимодействовать с программой поиска на Википедии. Благодаря этому классу, программа может принимать ввод от пользователя, выводить результаты поиска и позволять выбрать нужную статью для дальнейшей обработки.

**3. Описание интерфейса пользователя программы**

Программа запрашивает у пользователя наименование статьи в Wikipedia. Затем, если такие статьи найдены, то выводит 10 статей и запрашивает у пользователя выбрать номер статьи, которую нужно открыть в браузере. Эти действия иллюстрирует рисунок 1.

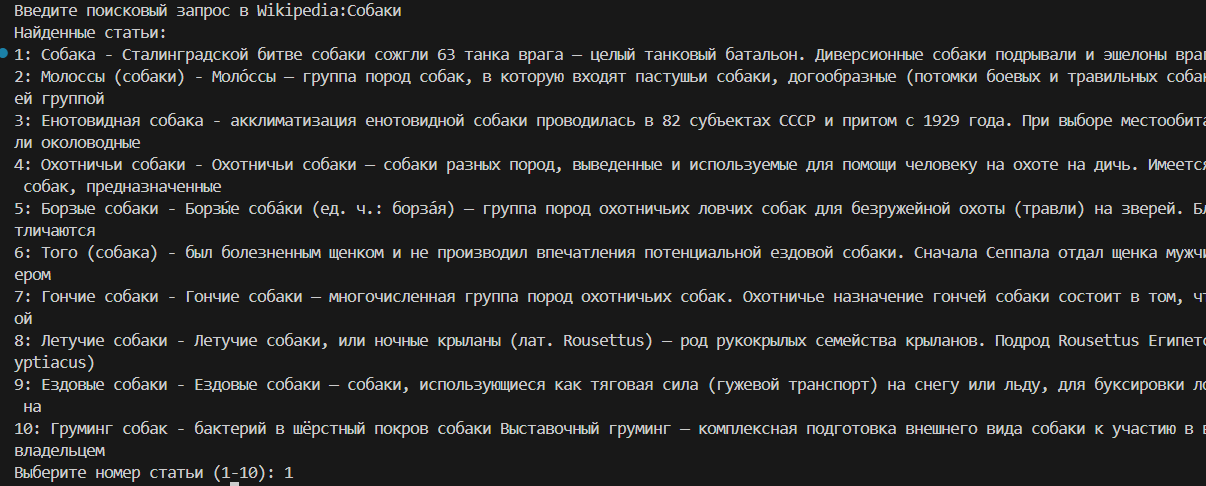


Рисунок 1 – интерфейс пользователя в консоли

После ввода номера статьи, она открывается в браузере. Это показано на рисунке 2.

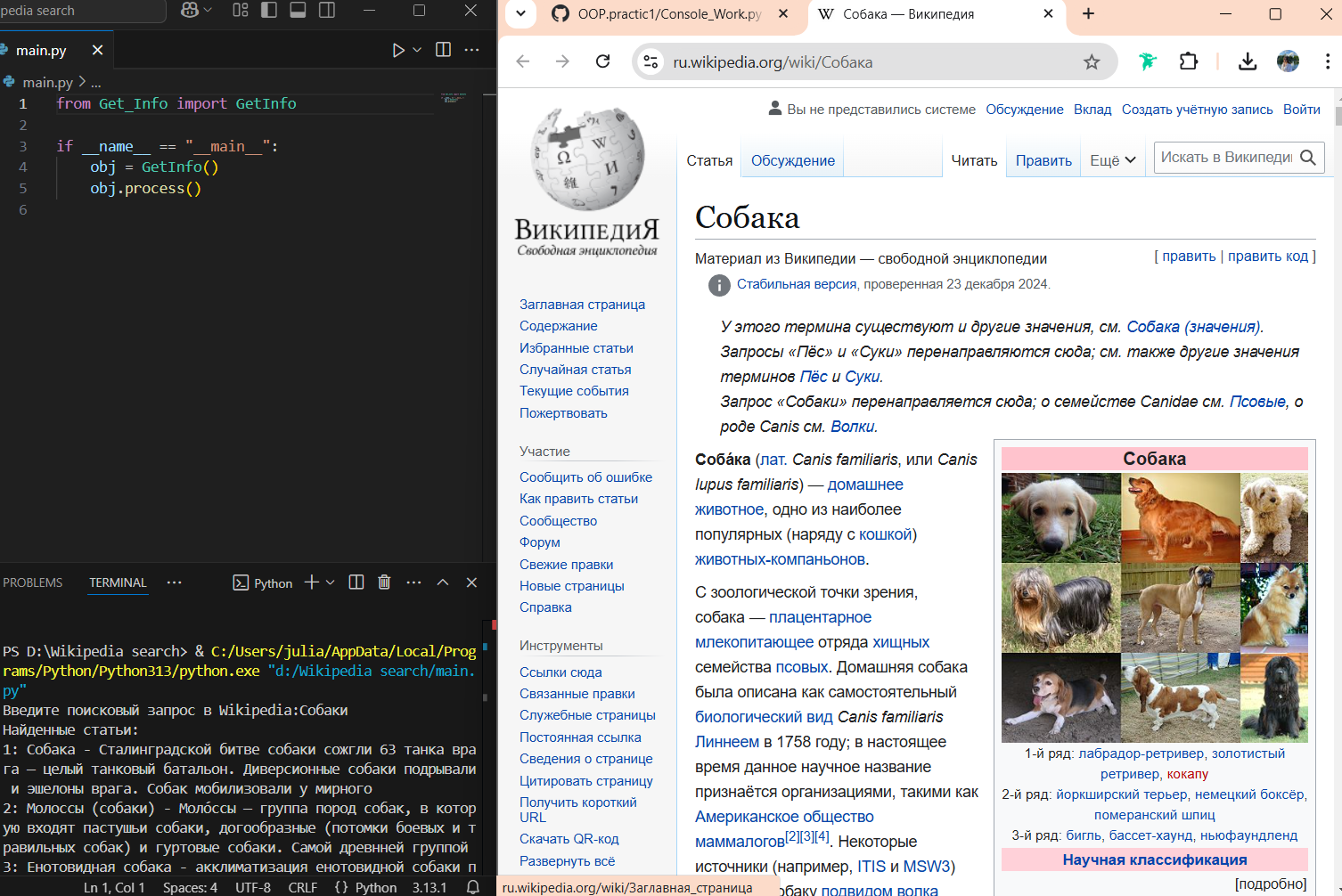


Рисунок 2 – открытие статьи в браузере

**4. Текст программы**

class ConsoleWork:

def get\_query(self):

return input("Введите поисковый запрос в Wikipedia:")

def show\_results(self, results):

print("Найденные статьи:")

for i, (title, snippet) in enumerate(results):

print(f"{i + 1}: {title} - {snippet}")

def get\_selection(self, max\_selection):

while True:

try:

selection = int(input(f"Выберите номер статьи (1-{max\_selection}): "))

if 1 <= selection <= max\_selection:

return selection - 1

else:

print(f"Пожалуйста, выберите номер от 1 до {max\_selection}.")

except ValueError:

print("Пожалуйста, введите действительный номер.")

import webbrowser

from Console\_Work import ConsoleWork

from Search import WikiSearch

class GetInfo:

def \_\_init\_\_(self):

self.searcher = None

self.console = ConsoleWork()

def process(self):

query = self.console.get\_query()

self.searcher = WikiSearch(query)

self.searcher.search()

results = self.searcher.get\_results()

if results:

self.console.show\_results(results)

selection = self.console.get\_selection(len(results))

title, snippet = results[selection]

# Обработка URL для открываемой статьи

base\_url = "https://ru.wikipedia.org/wiki/" if self.searcher.detect\_language() == 'ru' else "https://en.wikipedia.org/wiki/"

article\_url = f"{base\_url}{title.replace(' ', '\_')}"

webbrowser.open(article\_url)

else:

print("Статьи по вашему запросу не найдены.")

import requests

from langdetect import detect

from bs4 import BeautifulSoup

class WikiSearch:

def \_\_init\_\_(self, query):

self.query = query

self.results = []

def detect\_language(self):

try:

lang = detect(self.query)

return lang

except:

return 'en'

def search(self):

lang = self.detect\_language()

if lang == 'ru':

URL = "https://ru.wikipedia.org/w/api.php"

else:

URL = "https://en.wikipedia.org/w/api.php"

params = {

"action": "query",

"list": "search",

"srsearch": self.query,

"format": "json",

"srlimit": 10,

"utf8": 1

}

answer = requests.get(URL, params=params)

json\_answer = answer.json()

if "query" in json\_answer and "search" in json\_answer["query"]:

self.results = json\_answer["query"]["search"]

def get\_results(self):

results = [

(result["title"], self.clean\_html(result["snippet"])) for result in self.results

]

return results

def clean\_html(self, raw\_html):

soup = BeautifulSoup(raw\_html, "html.parser")

return soup.get\_text()

from Get\_Info import GetInfo

if \_\_name\_\_ == "\_\_main\_\_":

obj = GetInfo()

obj.process()

**5.** **Выводы**

В процессе работы была создана программа с графическим интерфейсом для поиска информации на сайте Википедия.

Программа позволяет пользователю ввести поисковый запрос и выбрать результат. Затем она обращается к API Википедии, передавая необходимые параметры. Полученные данные преобразуются в формат JSON и обрабатываются для извлечения заголовков и идентификаторов страниц (парсинг).

После этого программа выводит результаты поиска и открывает страницы в браузере, если пользователь выбирает нужный результат.

В итоге программа успешно выполняет свою задачу и работает без ошибок.

Результаты были выложены на Github: <https://github.com/Julia3489/OOP.practic1.git>