Języki skryptowe

dokumentacja projektu:

ZADANIE 2 – ODWRÓCONA PIRAMIDA

Zadanie zaproponowali: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska (zadanie finałowe edycji 2015/16)

"Zadania 2017"

Piotr Dusiński, grupa 3/5

3 stycznia 2025

ZADANIE 2 – ODWRÓCONA PIRAMIDA

Zadanie zaproponowali: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska (zadanie finałowe edycji 2015/16)

Istnieją wyrazy, z których poprzez wykreślanie liter i anagramowanie (przestawianie kolejności bez zmiany ilości wystąpień liter) można dojść (za każdym razem otrzymując istniejący wyraz) aż do słów jednoliterowych, np.:

L A W E T A
W A L E T
L E W A
E W A
W E
W

ALGORYTMION

Zespół „Algorytmion"
Politechnika Śląska
Wydział Matematyki Stosowanej
ul. Kaszubska 23
44-100 Gliwice

Studenckie Koło Naukowo-Informatyczne „Link"
Politechnika Śląska
Wydział Matematyki Stosowanej
ul. Kaszubska 23
44-100 Gliwice

POLITECHNIKA ŚLĄSKA

Wydział Matematyki Stosowanej
Studenckie Koło Naukowo-Informatyczne „Link"
ul. Kaszubska 23, 44-100 Gliwice

Istnieją też wyrazy, dla których jest to niemożliwe, np. *chrząszcz*.

W pliku *słownik.txt* znajduje się słownik, w którym słowa (każde w nowej linii) posortowane są alfabetycznie. Napisz program, który na podstawie tego słownika będzie weryfikował czy dla zadanego słowa procedura ta jest możliwa (jeśli tak, to wypisze ciąg tych wyrazów, jeśli nie napisze, że nie jest to możliwe).

UWAGA: w słowniku nie ma „słów" jednoliterowych!

Spis treści

Wstęp

Projekt został stworzony na potrzeby warunku zaliczeniowego przedmiotu "Języki skryptowe (Informatyka st. I sem. 3)". Problem, którego należało rozwiązać to - tworzenie anagramów z słowa, z którego została wykreślona któraś z jego liter, proces ten należy powtarzać aż, dopóki nie wystąpi pojedyncza litera, a następnie wyświetlanie wyników w formie odwróconej piramidy.

Według wymagań założonych na utworzenie projektu, została utworzona architektura serwera raportującego oraz plik `program.exe`, który wykonuje działanie programu.

Architektura

Program wykorzystuje wersje Pythona 3.12.2. Struktura programu przedstawia się w następujący sposób:

```
│   process.bat
│   program.exe
│   program.py
│   program.spec
│
├───in
│       slownik.txt
│
├───out
│       host.py
│
└───scripts
    │   main.py
    │
    ├───find_anagram
    │   │   find_anagram.py
    │   │
    │   └───__pycache__
    │           find_anagram.cpython-312.pyc
    │
    ├───read
    │   │   read.py
    │   │
    │   └───__pycache__
    │           read.cpython-312.pyc
    │
    └───write
        │   write.py
        │
        └───__pycache__
                write.cpython-312.pyc
```

Opis struktury

```
│    process.bat – skrypt powłoki systemowej batch
│    program.exe – uruchomienie całego projektu
│    program.py – kod program.exe
│    program.spec – specyfikacja program.exe
├───in – zawiera plik tekstowy z słownikiem, który jest potrzebny do szukania anagramów.
├───out – host.py odpowiada za serwer raportujący http
└───scripts – folder zawierający logikę działania całego programu
        │    main.py – skrypt włączający host.py
        │
        ├───find_anagram
        │    │    find_anagram.py – skrypt z funkcją szukającą anagramu
        │    │
        │    └───__pycache__
        │             find_anagram.cpython-312.pyc
        │
        ├───read
        │    │    read.py – skrypt czytający slownik.txt
        │    │
        │    └───__pycache__
        │             read.cpython-312.pyc
        │
        └───write
              │    write.py – wzór zapisu html (nie używany przez inne skrypty)
              │
              └───__pycache__
                    write.cpython-312.pyc
```

Logika programu

1. program.exe uruchamia skrypt process.bat, który następnie uruchamia scripts/main.py
2. W pliku main.py za pomocą host.py, skrypt rozpoczyna działanie od uruchomienia serwera localhost na porcie 8080:

`main.py`

```python
# Uruchomienie serwera rapotującego
print("Uruchamianie serwera...")
server_process = subprocess.Popen(["python", file_path_out])
time.sleep(2)
print("Serwer uruchomiony. Wykonywanie dalszych operacji...")
```

`host.py`

```python
# Uruchomienie serwera
PORT = 8080
server = HTTPServer(("localhost", PORT), ResultHandler)
print(f"Serwer HTTP uruchomiony na porcie {PORT}")
```

3. `main.py` zaczyna wykonywać funkcję `def search_anagram()`, która zaczyna od sczytania wszystkich słów i zapiswania je do listy, z `slownik.txt`

```python
def search_anagram():
    # Czytanie slownik.txt z folderu in
    file_content: list = read_file(file_path_in)
```

4. Następnie funkcja sprawdza poprawność wprowadzonego słowa przez użytkownika

```python
word = input("Podaj słowo: ")
    pattern = re.compile(r"^\w+$", re.UNICODE)
    while not word.strip() or not pattern.match(word):
        print(
            "Słowo nie może być puste i musi zawierać tylko litery. Spróbuj ponownie."
        )
        word = input("Podaj słowo: ")
```

5. Po czym uruchamia funkcje `find_anagram()`

Opis funkcji `find_anagram()`

`find_anagram()` przyjmuje dwa argumenty:

- Słowo – słowo podane przez użytkownika
- Dane – słowa z pliku `slownik.txt`

Funkcja zaczyna od stworzenia tablicy, która przechowuje najlepiej pasujące do słowa anagramy, gdzie architektura jej to – krotki, gdzie pierwszy argument to indeks słowa w Dane a drugi to ilość pasujących liter do Słowo.

Po czym funkcja rozbija Słowo na pojedyncze litery i zapisuje je do listy. Rozpoczyna ona po tym iterowanie po długości Dane i wykonuje podobną operacje co w przypadku Słowo.

```python
def find_anagram(word: str, data: list[str]) -> list:

    words_that_matches: List[Tuple[int, int]] = []

    word = word.lower().strip()
    letters_of_word = [char for char in word if char.isalpha()]

    for i in range(len(data)):

        compare_word = data[i].lower().strip()
        letters_to_compare = [char for char in compare_word if char.isalpha()]

```

By zoptymalizować działanie i szybkość funkcji jest liczony diff – różnica długości Słowo i aktualnego wyrazu z Dane, jeżeli jest ona większa od 1 lub mniejsza od 1, wtedy funkcja nie bada, czy litery pasują do siebie.

Dalej, jeżeli warunek jest spełniony wykonywana jest operacja porównywania liter, jeżeli pasują one do siebie między wyrazami oraz litera na danym indeksie nie wystąpiła już raz w dopasowaniu to do licznika jest dodawana jedynka

Po zakończeniu lub przerwaniu funkcji by dodać do słów, które spełniają oczekiwania jest warunek by diff był równy jeden oraz by liczba pasujących liter dla danego wyrazu z Dane była równowartością długości Słowo-1

Po zakończeniu iterowania po Dane zwracana jest lista słów pasujących do wymagań zadania

```
1
2       diff = len(letters_of_word) - len(letters_to_compare)
3
4       letters_to_ignore = []
5       letter_counter = 0
6       for j in range(len(letters_of_word)):
7           if diff > 1 or diff < 1:
8               break
9           for n in range(len(letters_to_compare)):
10              if (
11                  letters_of_word[j] == letters_to_compare[n]
12                  and n not in letters_to_ignore
13              ):
14                  letter_counter += 1
15                  letters_to_ignore.append(n)
16                  break
17
18      if diff == 1 and letter_counter == len(letters_of_word) - 1:
19          words_that_matches.append((i, letter_counter))
20  return words_that_matches
```

6. Wracając do `main.py` tworzona jest lista do raportu oraz sprawdzana jest zawartość listy zwróconej przez `find_anagram()` po czym jest wyświetlany odpowiedni komunikat.

```
report_list = []

if anagram_list == []:
    print(
        "Dla danego słowa nie występuje możliwość stworzenia odwróconej piramidy dla"
    )
    report_list.append(f"Niemożność stworzenia piramidy dla - {word}")
else:
    print("Dla danego słowa można utworzyć piramidę")
    print("Tworzenie piramidy...\n")
```

7. Jeżeli lista zwrócona z anagramami nie jest pusta wykonywana jest piramida. Piramida jest wykonywana przez rekurencje `find_anagram()` dopóki nie zwróci ta funkcja pustej listy. Z listy która jest zwracana przez `find_anagram()`, jest wybierany losowy element i następnie wypisywany.

```
# Tworzenie piramidy
print(word.upper())
report_list.append(word.upper())
while True:
    anagram_list = find_anagram(word, file_content)
    if not anagram_list:
        break
    random_tuple = random.choice(anagram_list)
    index = random_tuple[0]
    word = file_content[index].strip()
    report_list.append(word.upper())
    print(word.upper())
```

8. Następnie dane są przesyłane do serwera raportującego

```
1
2  # Przesyłanie danych do serwera
3  data = ",".join(report_list)
4  url = f"http://localhost:8080/results?data={urllib.parse.quote(data)}"
5  webbrowser.open(url)
6
```

Opis działania

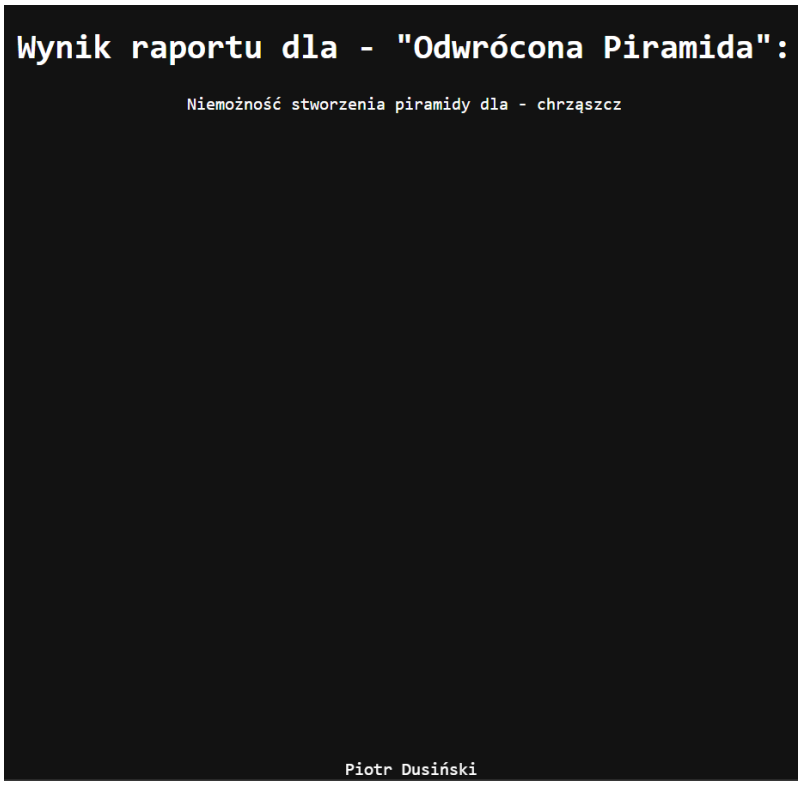
1. Należy uruchomić `program.exe` po czym powinno się wyświetlić

```
[PROGRAM.exe]: Skrypt zostal pomyslnie uruchomiony
[BATCH]: Uruchamiam skrypt Python...
Uruchamianie serwera...
Serwer HTTP uruchomiony na porcie 8080
Serwer uruchomiony. Wykonywanie dalszych operacji...
Szukanie anagramu dla każdego o jeden mniejszy od podanego słowa aż do pozycji: jedna litera
Podaj słowo:
```

2. Po czym należy wpisać słowo, dla którego chcemy wykonać operacje utworzenia piramidy, po czym należy odczekać chwilę by program mógł sprawdzić czy można wykonać zadanie.

```
Podaj słowo: chrząszcz
Chwila...

Dla danego słowa nie występuje możliwość stworzenia odwróconej piramidy dla
Raport http utworzono pomyślnie
Czy chcesz powtórzyć szukanie anagramu? (tak/nie):
```

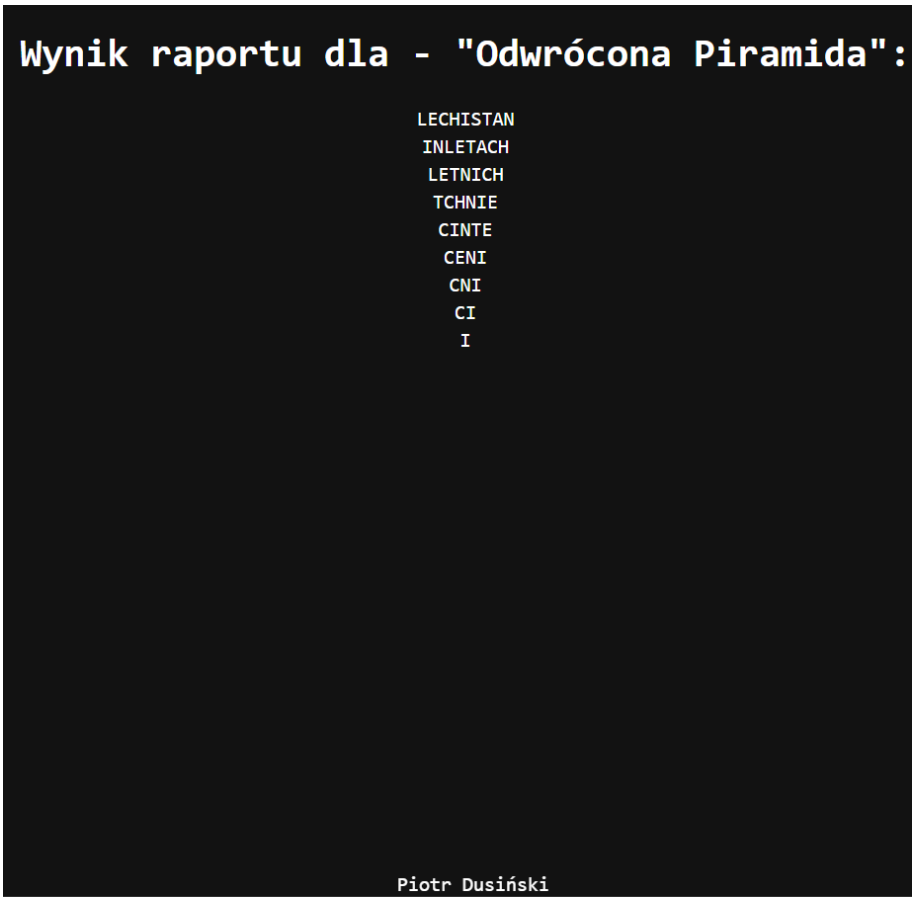
- Niestety w tym scenariuszu nie można utworzyć piramidy. Powinien się wyświetlić w przeglądarce wynik raportu. W konsoli można wykonać operacje ponownie wypisując "tak"



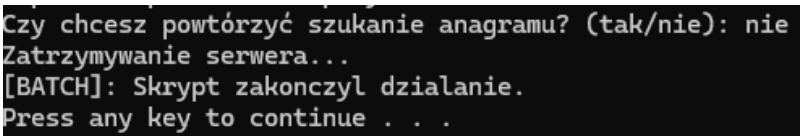
- Próba innego scenariusza

```
Czy chcesz powtórzyć szukanie anagramu? (tak/nie): tak
Szukanie anagramu dla każdego o jeden mniejszy od podanego słowa aż do pozycji: jedna litera
Podaj słowo: lechistan
Chwila...

Dla danego słowa można utworzyć piramidę
Tworzenie piramidy...

LECHISTAN
INLETACH
LETNICH
TCHNIE
CINTE
CENI
CNI
CI
I
Raport http utworzono pomyślnie
Czy chcesz powtórzyć szukanie anagramu? (tak/nie):
```

- Sukces



3. W każdym z scenariuszy automatycznie w przeglądarce wyświetli się wynik raportu, następnie program pozwala na ponowne wykonanie zadania lub gdy wpisać "nie" wyłączy on serwer raportujący i zamknie program.



Bibliografia

- https://www.w3schools.com/
- https://www.geeksforgeeks.org/
- https://stackoverflow.com/