Języki Skryptowe

dokumentacja projektu Tataraki i balonik

Kamil Ptak, grupa 4/8

15 grudnia 2022

Część I

Opis programu

W pliku słownik.txt znajduje się słownik, w którym słowa (każde w nowej linii) posortowane są alfabetycznie.

Napisz program, który dla zadanej początkowej litery słowa, poszukiwał będzie w tym słowniku wyrazów, które da się podzielić na pewnym miejscu w ten sposób, że zarówno do miejsca podziału jak i od miejsca podziału (tę część czytamy wspak), tak powstałe słowa również znajdują się w tym słowniku.

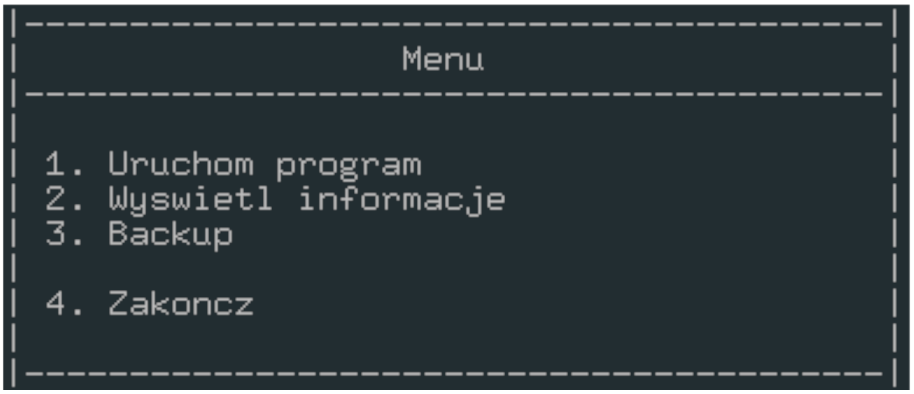
Przykładowo, jeśli podalibyśmy jako argument literę t, to program mógłby znaleźć słowo tataraki, bo dzieląc je po czwartej literze, otrzymamy słowa tata i ikar, a dla litery b, program mógłby zwrócić słowo balonik (podział po trzeciej literze na słowa bal i kino).

Zakładamy dodatkowo, że zarówno poszukiwane słowo, jak i jego składowe, są co najmniej dwuliterowe.

Program ma zwracać wszystkie wyrazy spełniające warunki zadania (na zadaną literę początkową).

Instrukcja obsługi

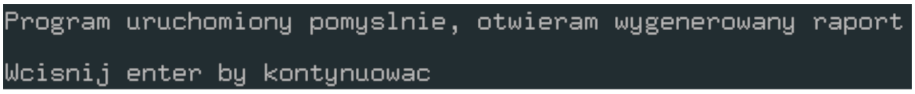
Aby uruchomić program należy włączyć skrypt menu.sh otwierający menu obsługi naszego programu. Po uruchomieniu wyświetli nam się tekst z instrukcją obsługi programu, wymagający podania przez użytkownika liczby w celu wykonania odpowiadającej mu funkcji.



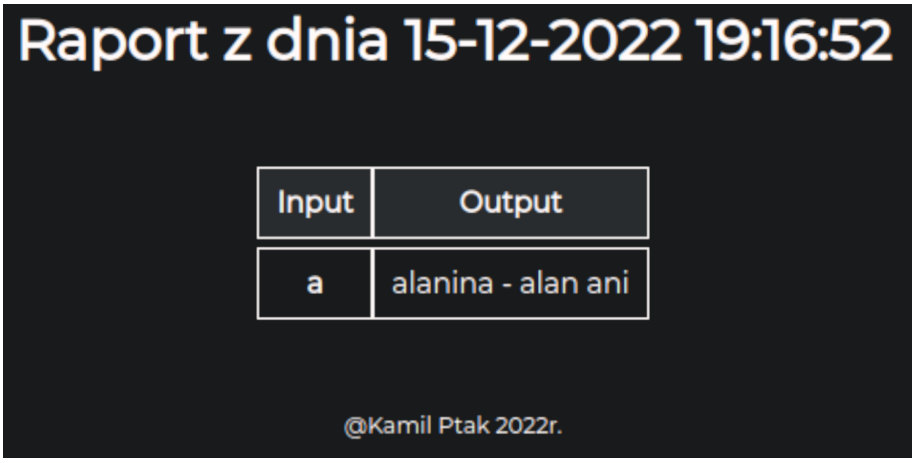
Rysunek 1: Główne menu programu

Możliwe wybory są następujące:

1. Uruchom program - Uruchamia program pobierając przy tym wszystkie dane z katalogu input i tworzy raport.html, który jest następnie wyświetlany w domyślnej przeglądarce systemowej



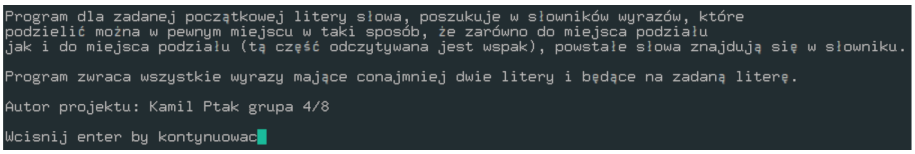
Rysunek 2: Przykładowy komunikat o pomyślnej próbie uruchomienia programu



Rysunek 3: Przykładowy raport programu

W przypadku braku wymaganych plików do uruchomienia programu, bądź błędnych plików input, program wyświetli odpowiedni komunikat z opisem problemu

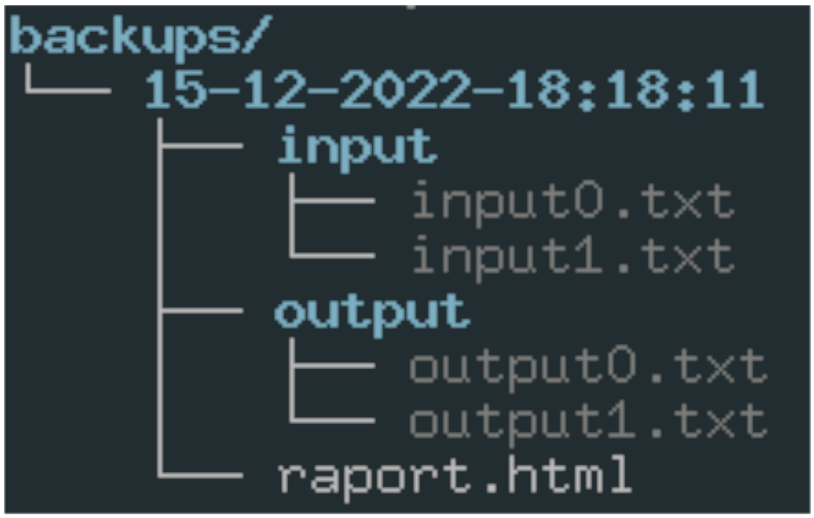
2. Wyswietl informacje - Wypisuje na ekranie konsoli opis założeń programu



Rysunek 4: Informacje o programie

3. Backup - Tworzy kopię zapasową danych w katalogu backups zawierającą raport.html oraz zawartość folderów input i output

Rysunek 5: Komunikat o dokonanym backupie



Rysunek 6: Przykładowa struktura zapisu backupu

4. Zakoncz - Zamyka menu, kończąc tym samym program.

Podanie innej liczby lub znaku, skutkuje powiadomieniem o wprowadzeniu niepoprawnego polecenia

Struktura danych programu

Program składa się z następującej struktury danych, wymaganych do prawidłowego uruchomienia aplikacji:

- menu.sh - Skrypt bash będący menu, którym uruchamia się program, wyświetla informacje o programie jak i tworzy kopie zapasową danych otrzymanych w wyniku wykonania tegoż programu
- tataraki.py - Skrypt python zawierający główny program, pobierający pliki wejściowe zawierające literę początkową słowa i tworzący plik wyjścia zawierające słowa z pliku słownik.txt, które są zgodne z założeniami projektu
- raport.py - Skrypt python pobierający dane z plików wejścia oraz wyjścia i generujący plik raport.html zawierający raport wszystkich danych w postaci tabeli
- Katalog wordbases zawierający bazowo tylko jeden plik:
    - slownik.txt - Plik tekstowy zawierający szeroki zbiór słów ze słownika języka Polskiego
- Katalog resources zawierający:
    - info.txt - Plik tekstowy zawierający opis programu, wyświetlany z poziomu menu.sh po wybraniu odpowiedniej opcji
    - menu.txt - Plik tekstowy zawierający opis korzystania z menu.sh, wyświetlany na każdym etapie korzystania z programu
    - style.css - Plik kaskadowego arkuszu styli wykorzystywany podczas generowania raportu w celu stylizacji tegoż raportu, jak i ukazania danych w sposób właściwy
- Katalog input zawierający pliki wejściowe, nazwane według klucza:

$$input0.txt, input1.txt ... inputX.txt \tag{1}$$

gdzie X to kolejna liczba rzeczywista.

Ponadto program w wyniku działania tworzy dodatkowo katalogi output oraz backups oraz pliku raport.html, które nie są wymagane do prawidłowego uruchomienia aplikacji.

Rysunek 7: Struktura danych programu w formie drzwa

Część II

Opis działania

Skrypt menu.sh pobiera kolejno wszystkie nazwy plików wejściowych z katalogu input i przekazuje je pojedynczo jako argument do tataraki.py. Następnie dla każdej pobranej nazwy program otwiera podany plik oraz zczytuje z niego pojedyńczą literę, która jest przypisywana do zmiennej keyletter. Aplikacja pobiera słowa z pliku słownik.txt do listy wordbank, oraz tworzy drugą listę zawierającą odwrócone wersje tych słów nazwaną reversedWordbank. Następnie tworzona jest kolejna lista zawierająca jedynie te słowa ze słownik.txt, które rozpoczynają się na tą samą literę co ta podana w pliku wejściowym, nazwana keywords, oraz pobiera z niej długość najdłuższego słowa longest_word.

Program następnie łączy ze sobą wszystkie słowa z list keywords i reversedWordbank, których łączna długość nie przekracza długości longest_word, i sprawdza czy znajdują się one w zbiorze keywords. Jeśli tak, to wpisywane są one do pliku temp_wbuffer.txt wraz ze słowami składowymi, przy czym drugie słowo jest ponownie odwracane.

Finalnie program kolejno wypisuje zawartosc pliku temp_wbuffer.txt do pliku output o tym samym numerze co ten znajdujący się w nazwie pliku input.

Otrzymane w ten sposób wyniki są następnie przetwarzane przez raport.py, który tworzy plik raport.html, który w tabeli umieszcza zarówno zawartość plików input oraz output, tylko i wyłącznie jeśli numer w ich nazwie jest taki sam, co sprawia, że raport zawiera jedynie pomyślnie wykonane iteracje programu.

Ostatecznie uruchamiana jest domyślna przeglądarka użytkownika, w której wyświetla się utworzony raport.html

Algorytm

Data: Dane wejściowe plik *input*
Result: Dane wyjściowe plik *output*
Do *keyletter* przypisz wartość *input*
Do *wordbank* wczytaj zawartość pliku *slownik.txt*
Do *reversedWordbank* przypisz odwrócone słowa z *wordbank*
Do *keywords* przypisz słowa z *wordbank* zaczynające się na *keyletter*
Do *longest_word* przypisz długość najdłuższego słowa z *keywords*
Utworz $buforx$, $bufory$

```
for x in keywords do
    for y in reversedWordbank do
        if długość(x+y) <= longest_word and x+y in keywords then
            if Jesli x znajduje sie w buforx i y znajduje sie w bufor then
                continue
            else
                Dopisz do pliku temp_wbuffer: x+y x odwrócone(y)
                Dopisz x do buforx
                Dopisz y do bufory
            end
        end
    end
end

if temp_wbuffer nie jest pusty then
    Przypisz do output zawartość temp_wbuffer
end
```

Algorithm 1: Algorytm wyszukiwania słów zaczynających się na daną literę, składających się ze zwykłego słowa i słowa pisanego wspak.

Implementacja systemu

Skrypt menu.sh przy uruchomieniu, jak i po każdym odświeżeniu pobiera i wypisuje z pliku "menu.txt"znajdującego się w katalogu "resources"instrukcje obsługi menu aplikacji.

Uruchomienie programu z poziomu skryptu menu.sh powoduje sprawdzenie czy istnieje katalog na pliki wynikowe "output", jeśli tak to usuwa go wraz z zawartością oraz tworzy go na nowo, następnie uruchamia tataraki.py wraz z argumentem, którym jest nazwa pliku wejścia z katalogu "input". Jeśli program znalazł wynik i zakończył się powodzeniem, tworzy on plik wynikowy do katalogu "output". Następnie usuwany jest plik "temp_wbuffer.txt"o ile takowy istnieje oraz sprawdzane jest czy w katalogu "output"znajdują się jakiekolwiek pliki. Jeśli tak to uruchamiany jest skrypt raport.py, który pobiera dane odnośnie stylizowania strony z pliku "style.css"znajdującego sie w katalogu "resources", tworzy plik raport.html w głównym katalogu projektu, a jeśli nie to program kończy się z odpowiednim komunikatem.

Opcja wyświetlająca informacje wypisuje dane z pliku "info.txt"znajdującego się w katalogu "resources".

Wybranie funkcji backup sprawdza czy istnieje w głównym katalogu plik raport.html, jeśli tak to tworzy katalog "backups"o ile takowy juz nie istnieje, następnie tworzy katalog nazwany aktualną datą i godziną wewnątrz folderu "backups", który natomiast zawiera kopię całego katalogu "input", katalogu "output"oraz pliku raport.html.

Wykorzystane biblioteki i przykłady ich użycia

- os

```
1 if os.stat("temp_wbuffer.txt").st_size == 0:
2     errorcodes("nooutput")
3 //Sprawdza czy plik "temp_wbuffer.txt" jest pusty, jesli tak to
      wywoluje funkcje errorcodes("nooutput")
4
5 os.remove("temp_wbuffer.txt")
6 //Usuwa plik "temp_wbuffer.txt"
7
8 self.outputfiles = [file for file in listdir("output") if isfile(
      join("output", file))]
9 //Tworzy liste plikow output
10 //os.listdir zwraca zawartosc katalogu w danej sciezce
11 //os.path.isfile zwraca true jesli podana sciezka jest plikiem
12 //os.path.join zwraca sciezke wraz z kazda mozliwa podsciezka
      katalogu
```

- datetime

```
now = datetime.now()
//Pobiera aktualny czas i przypisuje go do zmiennej now
```

- sys

```
input = str(sys.argv[1])
//Pobiera pierwszy argument podany podczas uruchamiania skryptu
    python

sys.exit("Nie znaleziono pliku slownikowego")
//Wyswietla w konsoli podany komunikat i przerywa dzialanie skryptu
    python
```

- re

```
if not bool(re.match('^[a-zA-Z]*$', keyletter)):
    errorcodes("keyletter")
//Sprawdza czy keyletter jest mala albo duza litera, jesli nie to
    wywoluje funkcje errorcodes("keyletter")
```

Funkcje zawarte w tataraki.py

```
//Funkcja zwracajaca podane slowo "word" napisane wspak
  def reverse(word):
            return word[::-1]

//Funkcja pobierajaca nazwe bledu "errorname", konczaca program
    przedwczesnie  i wypisujaca w konsoli odpowiedni komunikat
  def errorcodes(errorname):
        if errorname == "keyletter":
            sys.exit(f"{input} - Podana wartosc nie jest litera")
        if errorname == "input":
            sys.exit("Nie znaleziono pliku input")
        if errorname == "dictionary":
            sys.exit("Nie znaleziono pliku slownikowego")
        if errorname == "empty":
            sys.exit(f"{input} - Nie znaleziono slowa zaczynajacego sie
                na ta litere")
        if errorname == "nooutput":
            sys.exit(f"{input} - Nie znaleziono wynikow spelniajacych
                warunki")
```

Funkcje zawarte w raport.py

Raport.py zawiera klasę HTMLCreator, posiadającą następujące metody w celu sprawniejszego wygenerowania raportu.html

```
    def __init__(self): //Przy inicjacji klasy usuwa stary raport.html
        if exists("raport.html"):
            os.remove("raport.html")

        //Tworzy listy plikow input i output
        self.outputfiles = [file for file in listdir("output") if isfile
            (join("output", file))]
        self.inputfiles = [file for file in listdir("input") if isfile(
            join("input", file))]

        //Tworzy podstawowe pliki html
        now = datetime.now()
        self.fulldate = now.strftime("%d-%m-%Y %H:%M:%S")
        self.css = open("resources/style.css", "r")
        self.html = open("raport.html", "w")
        self.html.write(f"""<!DOCTYPE html>
            <html>
            <head>
                <title>Raport z dnia {self.fulldate}</title>
                <style>
                {self.css.read()}
                </style>
            </head>
            <body>
                <div class="container">
                <h1>Raport z dnia {self.fulldate}</h1>
                <table>
                <tr>
                    <th>Input</th>
                    <th>Output</th>
                </tr>\n""")

    //Tworzy znacznik html, ktory jest automatycznie domykany
    def doubletag(self, tag, content=""):
        self.html.write(f"<{tag}>{content}</{tag}>\n")

    //Tworzy znacznik html bez domkniecia
    def singletag(self, tag):
        self.html.write(f"<{tag}>\n")
```

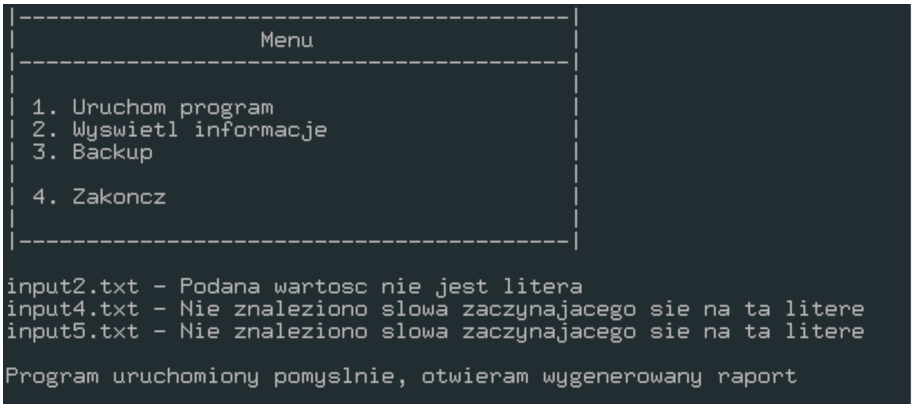
Testy

Dane wejściowe:

input0.txt = b
input1.txt = t
input2.txt = !
input3.txt = k
input4.txt = X
input5.txt = B

Dane wyjściowe:

output0.txt = balonik - bal kino
output1.txt = tataraki - tata ikar
output3.txt = kartonik - kart kino



Rysunek 8: Wynik programu w konsoli

Rysunek 9: Utworzony raport

Eksperymenty

Podczas realizacji z uwagi na duży początkowy rozmiar pliku słownik.txt, próbowałem zastosować kilka różnych algorytmów i sposobów przechowywania danych tymczasowych wymaganych do wykonania programu, w celu zoptymalizowania czasu jaki aplikacja wymagała, aby się wykonać.

Pierwotnie program przechowywał na osobnych listach pierwsze słowo (firstword), drugie słowo (secondword), oraz całe słowo(fullword), a algorytm wyglądał następująco:

```
    for x in fullword:
        for y in firstword:
            if len(y) > 1 and len(y) <= len(x)-2:
                temp_word = fullword.replace(firstword, "")
                for z in secondword:
                    if len(z) > 1 and len(z) == len(temp_word):
                    print(f"{x} - {y} {z}")
```

Skutkowało to długim czasem wykonywania z powodu zagnieżdżonych pętli i dużej złożoności czasowej.

Następną próbą realizacji było przechowywanie wszystkich możliwych kombinacji pierwszego słowa i drugiego slowa w pliku, a następnie sprawdzanie czy znajduje się on w zbiorze wordlist, co jednak skutkowało dużym zużyciem przestrzeni dyskowej powodując zatrzymanie programu z powodu braku miejsca na dysku. Przechowanie tych samych danych w postaci list również nie odniosło skutku, gdyż zużywało to za dużo pamięci RAM, również skutkując niepowodzeniem programu.

Szukając alternatywnych sposobów na przechowanie danych, próbowałem zastosować set oraz dictionary zamiast list, jednakże mimo szybszych operacji nie pozwalają one na przechowywanie duplikatów, co sprawia, że program nie wyświetlał wszystkich możliwych rozwiązań.

Pełen kod aplikacji

tataraki.py

```
# -*- coding: utf-8 -*-
import os
import sys
import re

input = str(sys.argv[1])


def reverse(word):
    return word[::-1]


def errorcodes(errorname):
    if errorname == "keyletter":
        sys.exit(f"{input} - Podana wartosc nie jest litera")
    if errorname == "input":
        sys.exit("Nie znaleziono pliku input")
    if errorname == "dictionary":
        sys.exit("Nie znaleziono pliku slownikowego")
    if errorname == "empty":
        sys.exit(f"{input} - Nie znaleziono slowa zaczynajacego sie na
            ta litere")
    if errorname == "nooutput":
        sys.exit(f"{input} - Nie znaleziono wynikow spelniajacych
            warunki")


database = "wordbases/slownik.txt"
wordbank = []  # zawiera wszystkie slowa
reversedWordbank = []  # zawiera wszystkie slowa odwrocone

file_number = ""
for letters in input:
    if letters.isdigit():
        file_number += letters

try:
    keyletter = open(f"input/{input}", "r")
    keyletter = keyletter.read(1)
except:
    errorcodes("input")

if not bool(re.match('^[a-zA-Z]*$', keyletter)):
    errorcodes("keyletter")

try:
    read = open(database, "r")
except:
    errorcodes("dictionary")

```

```
for x in read:
    x = x.strip()
    if len(x) > 1:
        wordbank.append(x)

read.close()

for x in wordbank:
    reversedWordbank.append(reverse(x))

keywords = []  # zawiera same slowa zaczynajace sie na keyletter
for x in wordbank:
    if x.startswith(keyletter):
        keywords.append(x)

try:
    longest_word = len(max(keywords, key=len))
except:
    errorcodes("empty")

wordsbuffer = open(f'temp_wbuffer.txt', 'w')
buforx=[]
bufory=[]

for x in keywords:
    for y in reversedWordbank:
        if len(x + y) <= longest_word and x + y in keywords:
            if x in buforx and y in bufory:
                pass
            else:
                wordsbuffer.write(x + y + " " + x + " " + reverse(y) + "
                    \n")
                buforx.append(x)
                bufory.append(y)

wordsbuffer.close()

if os.stat("temp_wbuffer.txt").st_size == 0:
    errorcodes("nooutput")

count = 0
with open(r"temp_wbuffer.txt", 'r') as fp:
    for count, line in enumerate(fp):
        pass

wordsbuffer_len = count + 1
wordsbuffer = open(f'temp_wbuffer.txt', 'r')
outputfile = open(f"output/output{file_number}.txt", "w")

for x in range(0, wordsbuffer_len):
    line = wordsbuffer.readline().strip()
    word = line.split(" ")[0]
    firstw = line.split(" ")[1]
    secondw = line.split(" ")[2]
    outputfile.write(f"{word} - {firstw} {secondw}\n")
```

```
103
104 wordsbuffer.close()
105 os.remove("temp_wbuffer.txt")
106 outputfile.close()
```

---

raport.py

```
1  import os
2  from datetime import date, datetime
3  from os import listdir
4  from os.path import isfile, join, exists
5
6
7  class HTMLCreator:
8      def __init__(self):
9          if exists("raport.html"):
10             os.remove("raport.html")
11
12         self.outputfiles = [file for file in listdir("output") if isfile
               (join("output", file))]
13         self.inputfiles = [file for file in listdir("input") if isfile(
               join("input", file))]
14
15         now = datetime.now()
16         self.fulldate = now.strftime("%d-%m-%Y %H:%M:%S")
17         self.css = open("resources/style.css", "r")
18         self.html = open("raport.html", "w")
19         self.html.write(f"""<!DOCTYPE html>
20             <html>
21             <head>
22                 <title>Raport z dnia {self.fulldate}</title>
23                 <style>
24                 {self.css.read()}
25                 </style>
26             </head>
27             <body>
28                 <div class="container">
29                 <h1>Raport z dnia {self.fulldate}</h1>
30                 <table>
31                 <tr>
32                     <th>Input</th>
33                     <th>Output</th>
34                 </tr>\n""")
35
36     def doubletag(self, tag, content=""):
37         self.html.write(f"<{tag}>{content}</{tag}>\n")
38
39     def singletag(self, tag):
40         self.html.write(f"<{tag}>\n")
41
42
43 raport = HTMLCreator()
44
```

```python
for x in range(len(raport.outputfiles)):
    output_number = ""
    for letters in raport.outputfiles[x]:
        if letters.isdigit():
            output_number += letters

    for y in range(len(raport.inputfiles)):
        input_number = ""
        for letters in raport.inputfiles[y]:
            if letters.isdigit():
                input_number += letters

        if input_number == output_number:
            raport.singletag("tr")
            inputfile = open(f"input/{raport.inputfiles[y]}", "r")
            raport.doubletag("td", f"{inputfile.read()}")

            raport.singletag("td")
            outputfile = open(f"output/{raport.outputfiles[x]}", "r")
            lines = outputfile.readlines()
            for index, singleline in enumerate(lines):
                raport.doubletag("div", f"{singleline.strip()}")
            raport.singletag("/td")

            raport.singletag("/tr")

raport.singletag("/table")
raport.doubletag("footer", "@Kamil Ptak 2022r.")
raport.singletag("/div")
raport.singletag("/body")
raport.singletag("/html")
raport.html.close()
```

---

menu.sh

```bash
#!/bin/bash
stop(){
  echo
  read -p "Wcisnij enter by kontynuowac" enter
  menu
}

menu(){
  clear
  while read -r line
  do
    echo "$line"
  done < resources/menu.txt

  read -n 1  -s wybor;

  case $wybor in
  "1")#Uruchom program
```

```
19      dir="output"
20      if [ -d $dir ]
21      then
22        rm -r $dir
23      fi
24      mkdir $dir
25      echo
26      for file in "input"/ *
27      do
28        file="${file:6}"
29        python tataraki.py "$file"
30      done
31      if [ -f "temp_wbuffer.txt" ];
32        then
33          rm temp_wbuffer.txt
34      fi
35      if [ -d $dir ]
36      then
37        if [ "$(ls -A $dir)" ]; then
38          python raport.py
39          echo
40          echo "Program uruchomiony pomyslnie, otwieram wygenerowany
                raport"
41          xdg-open raport.html </dev/null >/dev/null 2>&1 & disown
42          #not empty
43        else
44          echo "Brak wynikow dzialania programu"
45          #empty
46        fi
47      fi
48      stop
49      ;;
50    "2")#Wyswietl informacje
51      echo
52      while read -r line
53        do
54        echo "$line"
55        done < resources/info.txt
56
57      stop
58      ;;
59    "3")#Backup
60      if [ -f "raport.html" ]; then
61        if [ ! -d "backups" ];
62        then
63          mkdir backups
64        fi
65
66      printf -v date '%(%d-%m-%Y-%H:%M:%S)T\n' -1
67      mkdir backups/$date
68
69      cp -R input backups/$date
70      cp -R output backups/$date
71      cp raport.html backups/$date
72
```

```
73      echo "Backup zostal utworzony pomyslnie"
74      stop
75      else
76        echo "Nie znaleziono pliku raport.html"
77      fi
78      stop
79      ;;
80    "4")#Wyjdz
81      clear
82      exit
83      ;;
84    *)#Niepoprawne polecenie
85      echo "Wprowadzono niepoprawne polecenie"
86      stop
87      ;;
88    esac
89  }
90
91  menu
```