Politechnika Śląska
Wydział Matematyki Stosowanej
Kierunek Informatyka
Gliwice, 16.01.2025

Języki skryptowe

Dokumentacja projektu zaliczeniowego

"System raportujący w Pythonie i Batchu w oparciu o zadanie „Licznik długu” z XXVIII Olimpiady Informatycznej"

Wojciech Grzywocz gr. lab. 1/2

1. Opis projektu.

Program działa w oparciu o zadanie pt. "Licznik długu" z XXVIII Olimpiady informatycznej. W skrócie polega ono na tym iż otrzymujemy dane o długu wewnętrznym pewnego państwa i jego długu zagranicznym i mamy zbudować oprogramowanie do wyświetlacza długu sumarycznego. Jako dane otrzymujemy nie tylko początkowe kwoty długów ale również dostajemy informacje jak zmieniają się poszczególne pozycje/cyfry obu długów i na podstawie tych zmian wartości mamy aktualizować wyświetlacz wyświetlający dług sumaryczny.

Dokładne polecenie zadania z olimpiady brzmi:

Sytuacja ekonomiczna Bajtocji jest tragiczna – takie jest zdanie profesora Bajterowicza. Postanowił on zwrócić uwagę opinii publicznej na tę kwestię i zamówił w firmie Bajtazara zainstalowanie w centrum stolicy wielkiego wyświetlacza, na którym będzie pokazywany aktualny dług publiczny Bajtocji. Bajtazarowi przypadło w udziale napisanie oprogramowania do wyświetlacza. Urządzenie składa się z n cyfr dziesiętnych. Utrudnieniem jest fakt, że do oprogramowania wyświetlacza podawane są dwie liczby o co najwyżej n − 1 cyfrach: dług wewnętrzny Bajtocji (krajowy) oraz dług zewnętrzny Bajtocji (zagraniczny). Natomiast na wyświetlaczu ma zostać pokazana suma tych dwóch liczb. Wyświetlana liczba ma być aktualizowana w czasie rzeczywistym. Pomóż Bajtazarowi i napisz program, który umożliwi wykonywanie następujących operacji:

- *zmiana i-tej cyfry długu wewnętrznego,*
- *zmiana i-tej cyfry długu zewnętrznego,*
- *podanie i-tej cyfry sumarycznego długu.*

Wejście

W pierwszym wierszu wejścia są dwie liczby całkowite n i z ($2 \leq n \leq 100\,000$, $1 \leq z \leq 100\,000$) oznaczające długość wyświetlacza i liczbę operacji do wykonania. W drugim wierszu znajduje się liczba całkowita oznaczająca

początkową wartość długu wewnętrznego Bajtocji w postaci napisu złożonego z n − 1 cyfr (napis może posiadać zera wiodące). W trzecim wierszu w takim samym formacie znajduje się początkowa wartość długu zewnętrznego. W kolejnych z wierszach znajdują się opisy operacji. Każdy z tych wierszy jest w jednym z trzech formatów: • W i c – operacja zmiany i-tej cyfry długu wewnętrznego na c (1 ≤ i < n, 0 ≤ c ≤ 9), • Z i c – operacja zmiany i-tej cyfry długu zewnętrznego na c (ograniczenia jak wyżej), • S i – zapytanie o i-tą cyfrę sumarycznego długu (1 ≤ i ≤ n). Cyfry numerujemy od strony prawej (od najmniej znaczącej cyfry) do lewej.

Wyjście

Na wyjście należy wypisać po jednym wierszu dla każdej operacji S z wejścia. Wiersz ma zawierać jedną cyfrę c (0 ≤ c ≤ 9) będącą odpowiedzią na zapytanie.

Przykład

Dla danych wejściowych:

5 6

7341

0150

S 3

W 3 0

S 3

Z 1 9

S 1

S 3

poprawnym wynikiem jest:

4

1

0

2

Wykorzystując matematyczną wiedzę i zależności między liczbami oparłem algorytm w głównym pliku Zadanie.py o pisemne działania matematyczne.

Dla efektu wizualnego aby zmiany długu były lepiej widoczne postanowiłem dodać moduł Tkinter który pozwala na wyświetlenie w formie graficznej wskaźnika długu sumarycznego.

2. Budowa projektu i funkcjonalności.

Projekt znajduje się w folderze „Projekt_Grzywocz_Wojciech_inf1_2". Wewnątrz folderu mamy: 3 pliki Pythona .py (Zadanie.py, Raport.py, Backup.py) i jeden plik „BAT EXE wykonujący.bat" ze skryptem Batchowym, który służy do zarządzania projektem i wywoływania kolejnych skryptów Pythona podczas działania programu. Aby uruchomić program wystarczy kliknąć dwukrotnie na ten plik Batch a program uruchomi się automatycznie. Pliki Pythona mają natomiast za zadanie kolejno: przetwarzać dane wejściowe i realizować polecenia zgodne z treścią zadania z olimpiady informatycznej, a następnie zapisać wynik swoich działań do pliku wyniki.txt – za to odpowiedzialny jest plik Zadanie.py; następnie wywoływany jest plik Raport.py który tworzy raport w formacie pdf na podstawie początkowych danych wejściowych i danych wyjściowych z pliku Zadanie.py, tenże raport z danymi zostaje zapisany w folderze Raporty w moim projekcie.; po utworzeniu raportu jako trzeci rozpoczyna działanie plik Backup.py, którego zadaniem jest utworzenie kopi zapasowej wszystkich dotychczasowych raportów z folderu Raporty w jednym pliku. Efektem takiego działania jest utworzenie w folderze projektu pliku backup.pdf, który zawiera w sobie wszystkie dotychczas wykonane raporty. Przechowywanie takiego pliku backup ze wszystkimi raportami razem pozwala nam co jakiś czas opróżniać folder Raporty aby nie „zaśmiecać" miejsca na dysku dużą ilością plików. Dzięki plikowi backup mamy dostęp do wszystkich danych w jednym pliku. Plik takiej kopii zapasowej chroni nas również w przypadku nieumyślnego usunięcia ważnego dla nas raportu z folderu Raporty. Warto również zaznaczyć, że aby zminimalizować złożoność czasową i obliczeniową i tym samym uniknąć scalania wszystkich dotychczasowych raportów podczas jednego wykonywania programu, kod jest tak skonstruowany, że podczas jednego wywołania dopisuje do pliku backup.pdf tylko jeden nowo tworzony, podczas bieżącego obiegu programu, raport pdf.

(Jeśli plik backup nie istnieje początkowo w folderze projektu to pierwsze poprawne wywołanie programu powinno go utworzyć).

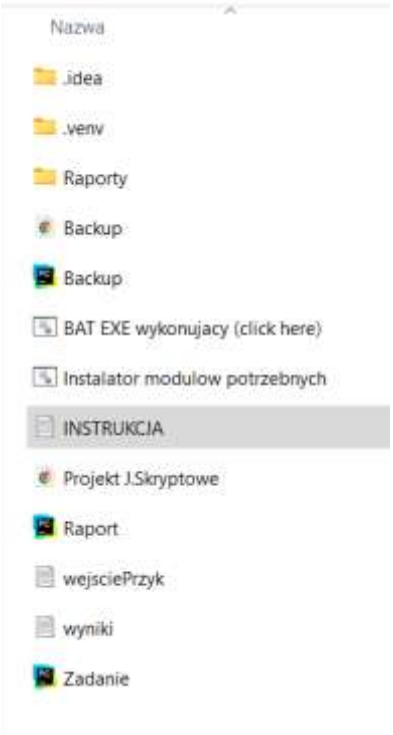
Do folderu projektu dodałem również plik .bat „Instalator modulow potrzebnych", po kliknięciu na który zostaną automatycznie zainstalowane wszystkie moduły potrzebne do prawidłowego działania programu.

W folderze projektu można również znaleźć Instrukcje w formacie txt oraz plik pdf z oficjalnym zadaniem „Licznik długu" z Olimpiady informatycznej.

Ważną uwagą jest też to iż można dodać do folderu projektu swoje dane wejściowe w formacie txt i z nich skorzystać podczas wykonywania programu, jednakże muszą być one koniecznie w formacie, który jest ukazany w pliku przykładowym wejsciePrzyk.txt lub w pliku z treścią zadania z olimpiady, gdyż takie było główne polecenie, żeby trzymać się tego formatu danych w pliku wejsciowym.

Pliki i foldery stałe po otwarciu folderu projektu: Zadanie.py, Raport.py, Backup.py, BAT EXE wykonujący.bat, Instalator modulow potrzebnych.bat, Instrukcja.txt, wejsciePrzyk.txt, Projekt J.Skryptowe.pdf, .venv, .idea

Pliki i foldery, które zostaną utworzone podczas pierwszego obiegu programu (i kolejnych): folder Raporty, backup.pdf, wyniki.txt oraz raporty w formacie pdf które zostaną zapisane do folderu Raporty.



Przykładowy wygląd folderu z projektem

3. Aspekty techniczne: kod, budowa programu i przebieg realizacji.

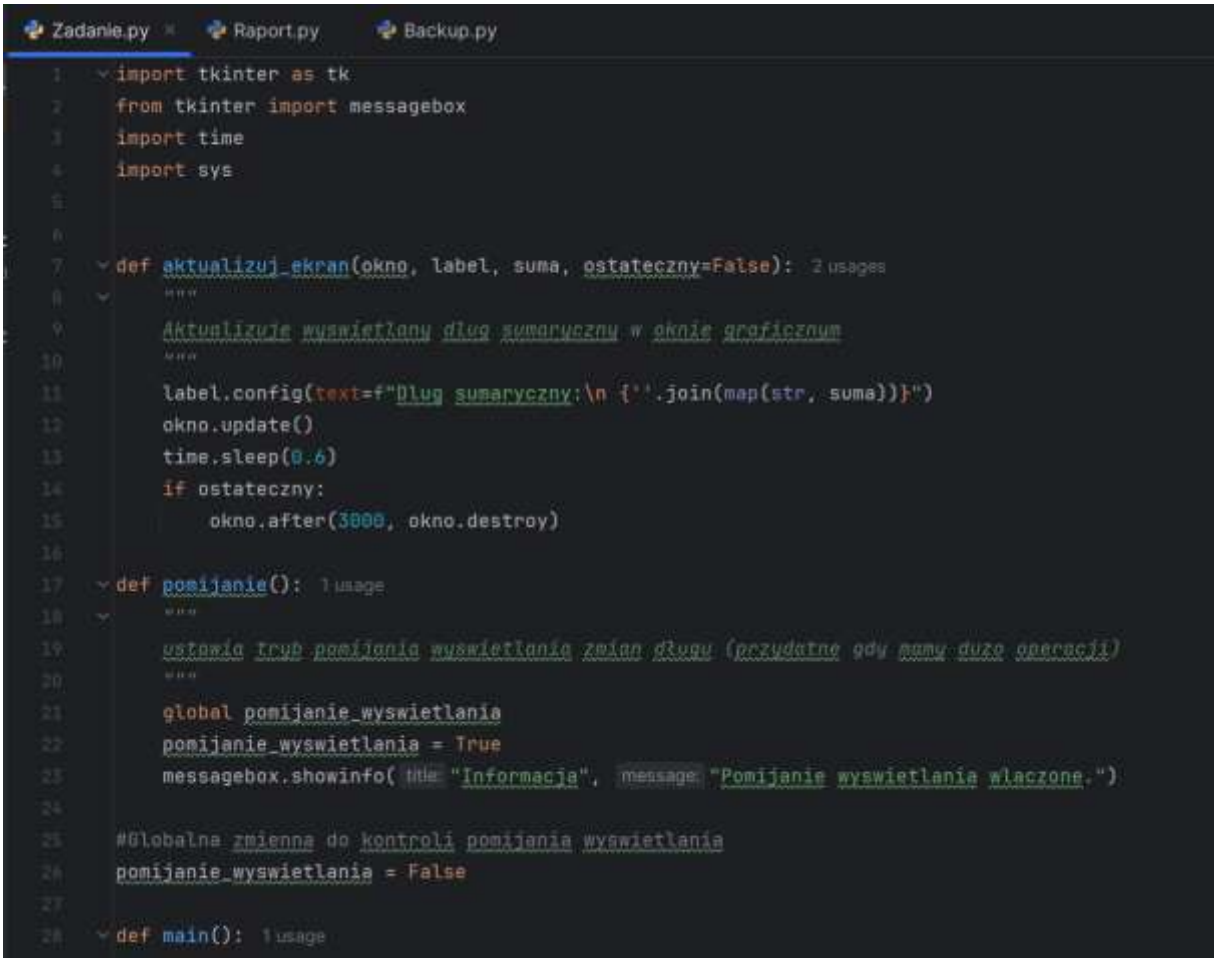
Wykorzystane moduły Pythona: time, sys, tkinter, reportlab, datetime, pathlib, pikepdf

- *Plik BAT EXE wykonujący.bat*

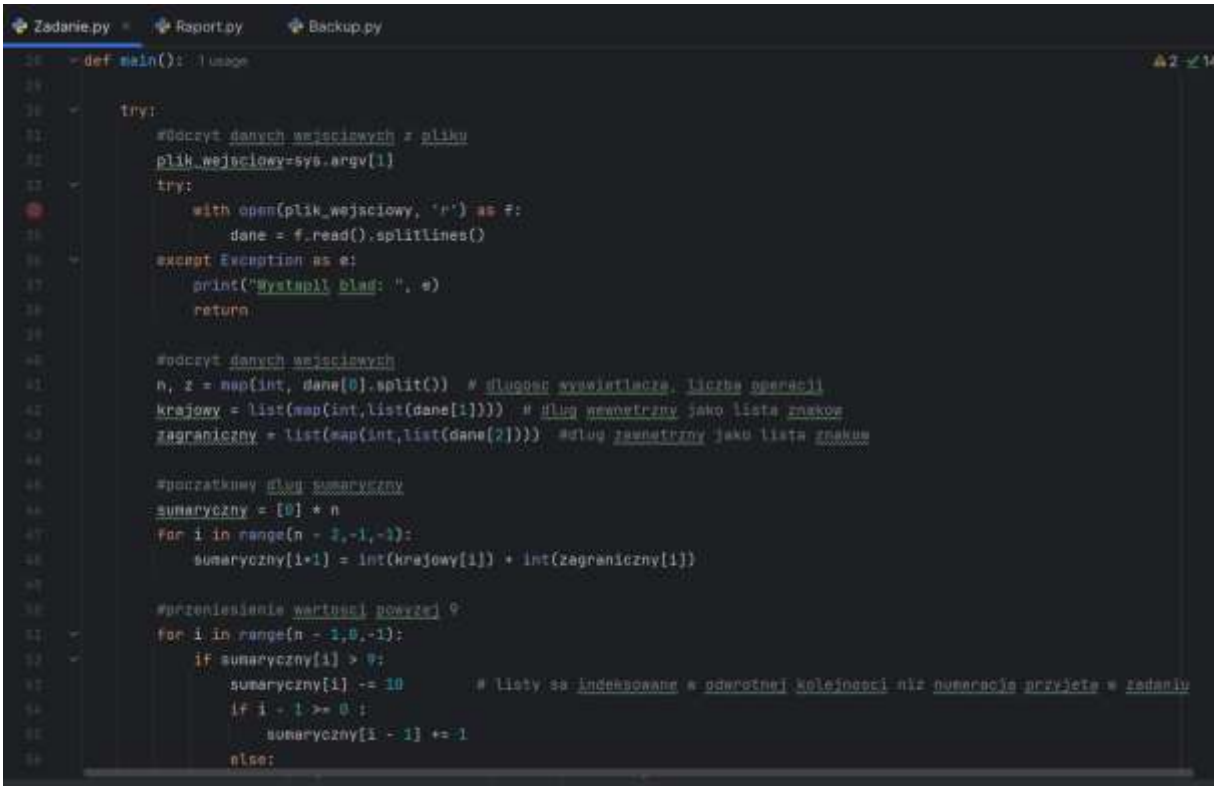
```
@echo off
REM Skrypt do zarządzania zadaniem, raportem i backupem

cd /d %~dp0

set /p "WEJSCIE=Podaj plik w ktorym masz dane wejsciowe
(np.dane.txt lub pelna sciezka): "

echo.

call python Zadanie.py "%WEJSCIE%"

for /f "delims=" %%i in ('python Raport.py "%WEJSCIE%"') do set
"nowy_raport=%%i"
setlocal enabledelayedexpansion
echo Nowy raport zapisano w pliku pdf: !nowy_raport!

call python Backup.py "!nowy_raport!"

echo Wszystkie operacje wykonano pomyslnie.
echo.
echo.
pause
```

Ustawia lokalizacje pracy na folder projektu, pyta użytkownika o podanie nazwy pliku z danymi lub pełnej ścieżki do pliku z danymi, a następnie w sposób zorganizowany wywołuje kolejne Pythony do realizacji głównych zadań.

- *Plik Zadanie.py*



W górnej części pliku zaimportowane są biblioteki oraz opisane są funkcje związane z interfejsem graficznym. Jedna funkcja która aktualizuje okno po każdej zmianie wartości długu a druga funkcja jest zastosowana jako funkcja akcji dla naciśnięcia przycisku pomijania wyświetlania na interfejsie graficznym.
Niżej znajduje się funkcja główna main opatrzona obsługą wyjątków try except.



Na początku program pobiera dane z zadanego pliku. Następnie przypisane zostają podstawowe zmienne i obliczony zostaje początkowy dług sumaryczny w oparciu o pisemne działania matematyczne.

```
def main():  1 usage

        #funkcja do  aktualizacji sumarycznego długu
    def aktualizuj_sumaryczny(index, delta_krajowy=0, delta_zagraniczny=0):
        przeniesienie = 0
        if delta_krajowy:
            sumaryczny[index] += delta_krajowy
        if delta_zagraniczny:
            sumaryczny[index] += delta_zagraniczny

        # zmiana cyfr i zachowanie przeniesienia
        while index>=0 and index < n:
            if sumaryczny[index] > 9:
                sumaryczny[index] -= 10
                przeniesienie = 1
                if index - 1 >= 0:
                    sumaryczny[index - 1] += przeniesienie
                index -= 1
                continue
            elif sumaryczny[index] < 0:
                sumaryczny[index] += 10
                przeniesienie = -1
                if index - 1 >= 0:
                    sumaryczny[index - 1] -= przeniesienie
                index -= 1
                continue
            else:
                break
```

Niżej znajduje się funkcja do aktualizowania długu w oparciu o pisemne działania matematyczne.

```
#tworzenie okna Tkinter do wyswietlania
global pomijanie_wyswietlania
okno = tk.Tk()
okno.title("Dług Bajtocji")
okno.config(bg="black")
okno_width = 700
okno_height = 500
screen_width = okno.winfo_screenwidth()       #pobieranie wymiarow ekranu
screen_height = okno.winfo_screenheight()
# obliczanie pozycji do wysrodkowania
x_cord = (screen_width // 2) - (okno_width // 2)
y_cord = (screen_height // 2) - (okno_height // 2)
okno.geometry(f"{okno_width}x{okno_height}+{x_cord}+{y_cord}")     #ustawianie wymiarow okna
label = tk.Label(okno, text=f"Dług sumaryczny:\n {''.join(map(str, sumaryczny))}", font=("Arial", 40), bg="black", fg="lime")
label.pack(pady=100)
przycisk = tk.Button(okno, text="Pomiń wyświetlanie", command=pomijanie)
przycisk.pack(pady=10)
```

Poniżej umieściłem podstawowe instrukcje dla interfejsu graficznego w module Tkinter.

```
def main():  1 usage
        wyniki = []
        for linia in dane[3:]:
            operacja = linia.split()
            if operacja[0] == 'W':  #zmiana cyfry dlugu krajowego
                i = n-1 - int(operacja[1])
                c = int(operacja[2])
                delta = c - krajowy[i]
                krajowy[i] = c
                aktualizuj_sumaryczny(i+1, delta_krajowy=delta)
            elif operacja[0] == 'Z':  #zmiana cyfry dlugu zagranicznego
                i = n-1 - int(operacja[1])
                c = int(operacja[2])
                delta = c - zagraniczny[i]
                zagraniczny[i] = c
                aktualizuj_sumaryczny(i+1, delta_zagraniczny=delta)
            elif operacja[0] == 'S':  #zapytanie o cyfre dlugu sumarycznego
                i = n - int(operacja[1])
                wyniki.append(str(sumaryczny[i]))
                #print(wyniki)

            #aktualizacja ekranu po kazdej operacji, jesli tryb wyswietlania jest wlaczony
            if not pomijanie_wyswietlania:
                aktualizuj_ekran(okno, label, sumaryczny)

        #wyświetlenie końcowego wyniku
        aktualizuj_ekran(okno, label, sumaryczny, ostateczny=True)
        okno.mainloop()
```

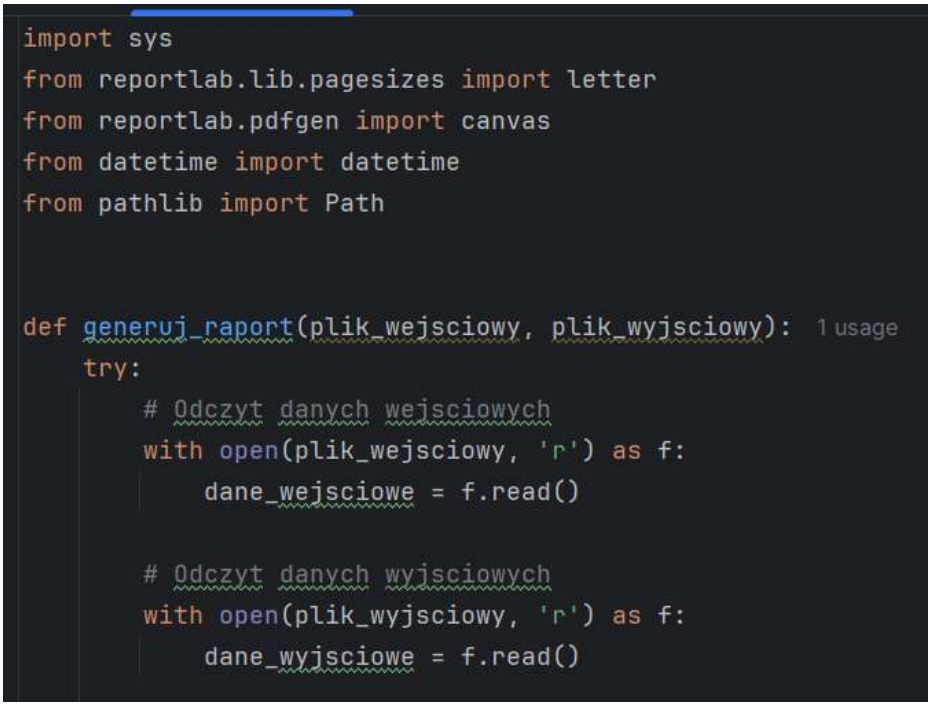
Następnie mamy główną pętle programu odpowiedzialną za przetwarzanie danych wejściowych w wyjściowe.

```
        # Zapis wyników do pliku wyjściowego
        try:
            with open('wyniki.txt', 'w') as f:
                f.write("\n".join(wyniki))
            print("Wyniki zapisane do pliku 'wyniki.txt'.")
        except Exception as e:
            print("Wystąpił błąd: ", e)




    except Exception as e:
        print("Wystąpił błąd: ",e)



if __name__ == "__main__":
    main()
```

Na koniec znajduje się zapis do pliku wyniki.txt danych wyjściowych i obsługa potencjalnych wyjątków.

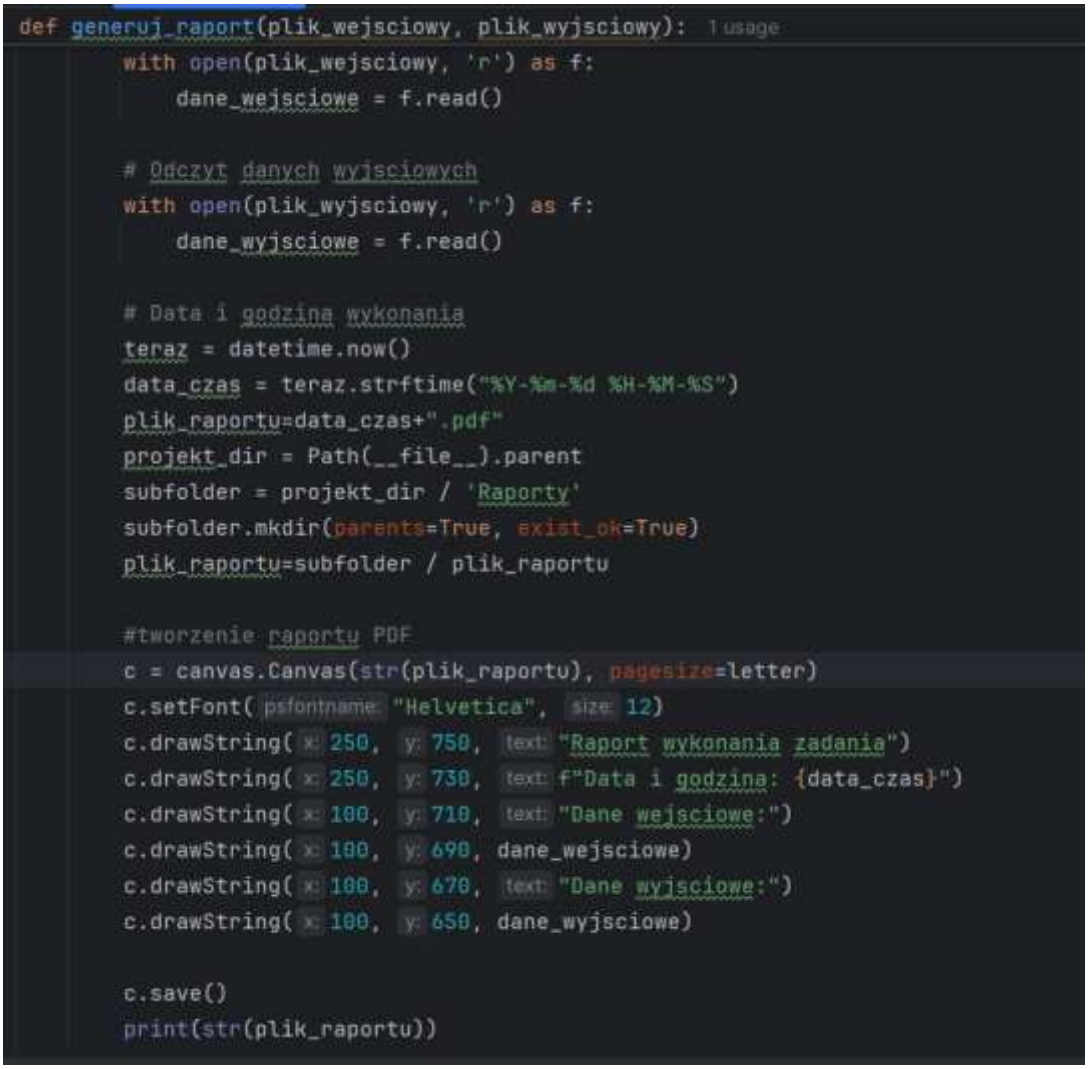
- *Plik Raport.py*

  Służy on do utworzenia raportu na podstawie danych wejściowych i danych wyjściowych z pliku Zadanie.py. Raport zostaje zapisany w pliku pdf w folderze Raporty.



```python
import sys
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
from datetime import datetime
from pathlib import Path


def generuj_raport(plik_wejsciowy, plik_wyjsciowy):  1 usage
    try:
        # Odczyt danych wejsciowych
        with open(plik_wejsciowy, 'r') as f:
            dane_wejsciowe = f.read()

        # Odczyt danych wyjsciowych
        with open(plik_wyjsciowy, 'r') as f:
            dane_wyjsciowe = f.read()
```

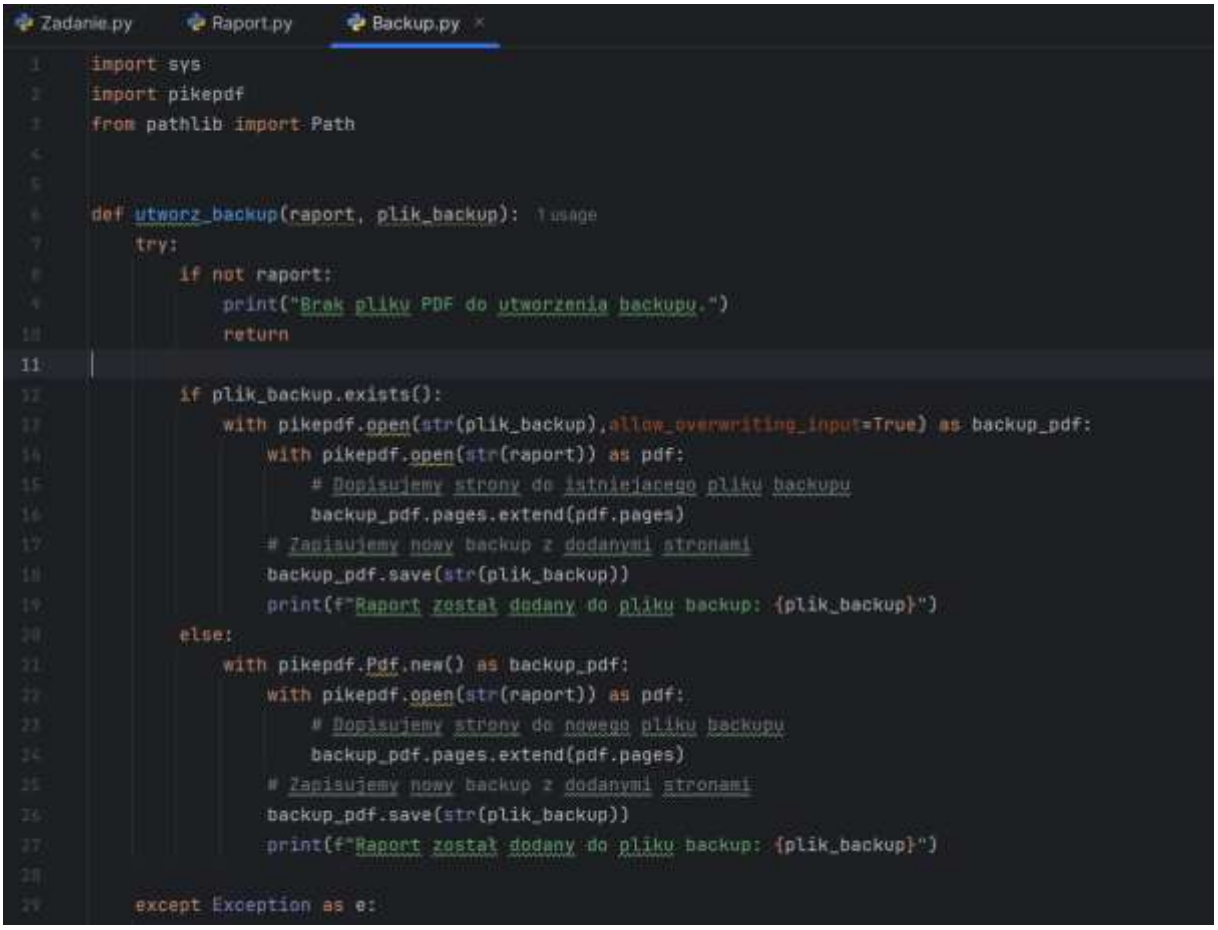
Plik wykorzystuje moduł reportlab oraz sys, datetime i pathlib.
Główna część kodu znajduje się w funkcji generuj raport, a cały ten blok kodu jest opatrzony obsługą wyjątków try except.



```
def generuj_raport(plik_wejsciowy, plik_wyjsciowy):  1 usage
    with open(plik_wejsciowy, 'r') as f:
        dane_wejsciowe = f.read()

    # Odczyt danych wyjsciowych
    with open(plik_wyjsciowy, 'r') as f:
        dane_wyjsciowe = f.read()

    # Data i godzina wykonania
    teraz = datetime.now()
    data_czas = teraz.strftime("%Y-%m-%d %H-%M-%S")
    plik_raportu=data_czas+".pdf"
    projekt_dir = Path(__file__).parent
    subfolder = projekt_dir / 'Raporty'
    subfolder.mkdir(parents=True, exist_ok=True)
    plik_raportu=subfolder / plik_raportu

    #tworzenie raportu PDF
    c = canvas.Canvas(str(plik_raportu), pagesize=letter)
    c.setFont( psfontname: "Helvetica",  size: 12)
    c.drawString( x: 250,  y: 750,  text: "Raport wykonania zadania")
    c.drawString( x: 250,  y: 730,  text: f"Data i godzina: {data_czas}")
    c.drawString( x: 100,  y: 710,  text: "Dane wejsciowe:")
    c.drawString( x: 100,  y: 690, dane_wejsciowe)
    c.drawString( x: 100,  y: 670,  text: "Dane wyjsciowe:")
    c.drawString( x: 100,  y: 650, dane_wyjsciowe)

    c.save()
    print(str(plik_raportu))
```

W funkcji następuje odczyt danych wejściowych i wyjściowych do zadania, a następnie jest utworzony pdf z raportem i zapisany do pliku raportu z nazwą jako odpowiednią datą i czasem wykonania raportu, a plik ten zostaje umieszczony w folderze Raporty.

- *Plik Backup.py*

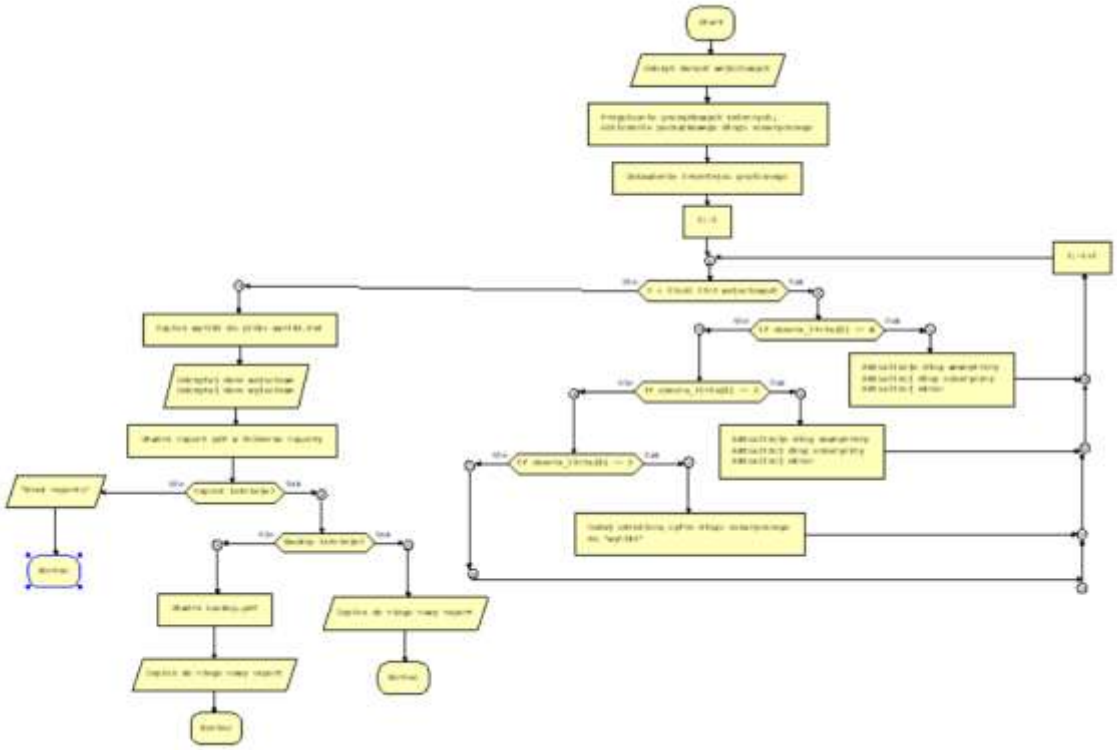


Plik Backup.py wykorzystuje moduły sys, pikepdf oraz pathlib. Główna część programu to funkcja utworz_backup która jest opatrzona instrukcją obsługi wyjątków try except.

Funkcja na początek sprawdza czy raport pdf do zapisu w ogóle istnieje, a jeśli tak to następnie sprawdza czy plik backup.pdf istnieje. Jeśli istnieje dopisuje nowy raport na końcu pliku backup, natomiast jeśli plik backup nie istnieje to program go tworzy i dodaje do niego pierwszy raport pdf.

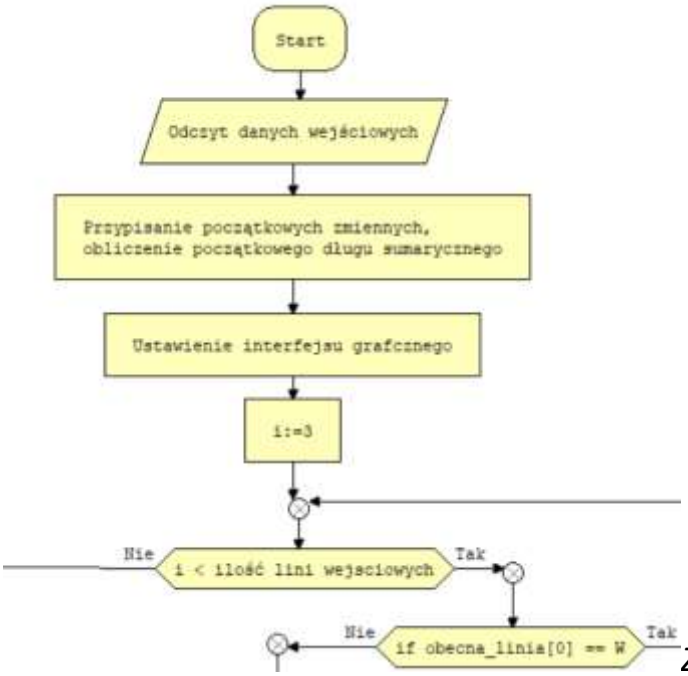
Na koniec mamy funkcje wywołującą utworzenie backupu.

- *Schemat blokowy programu:*

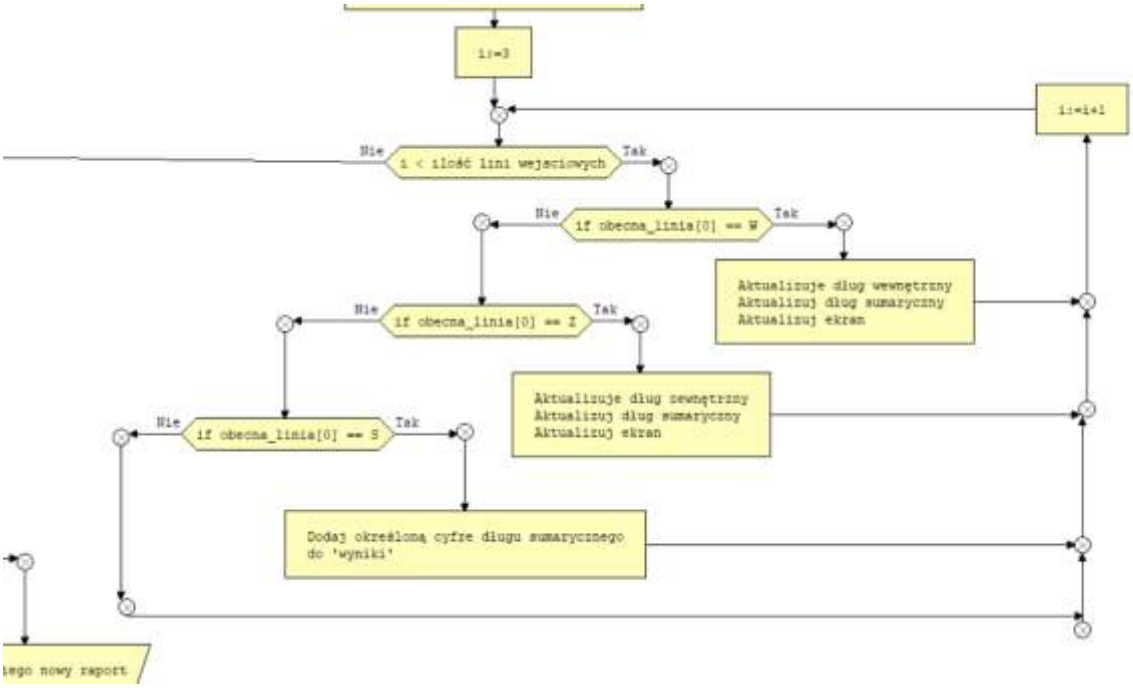


Cały schemat

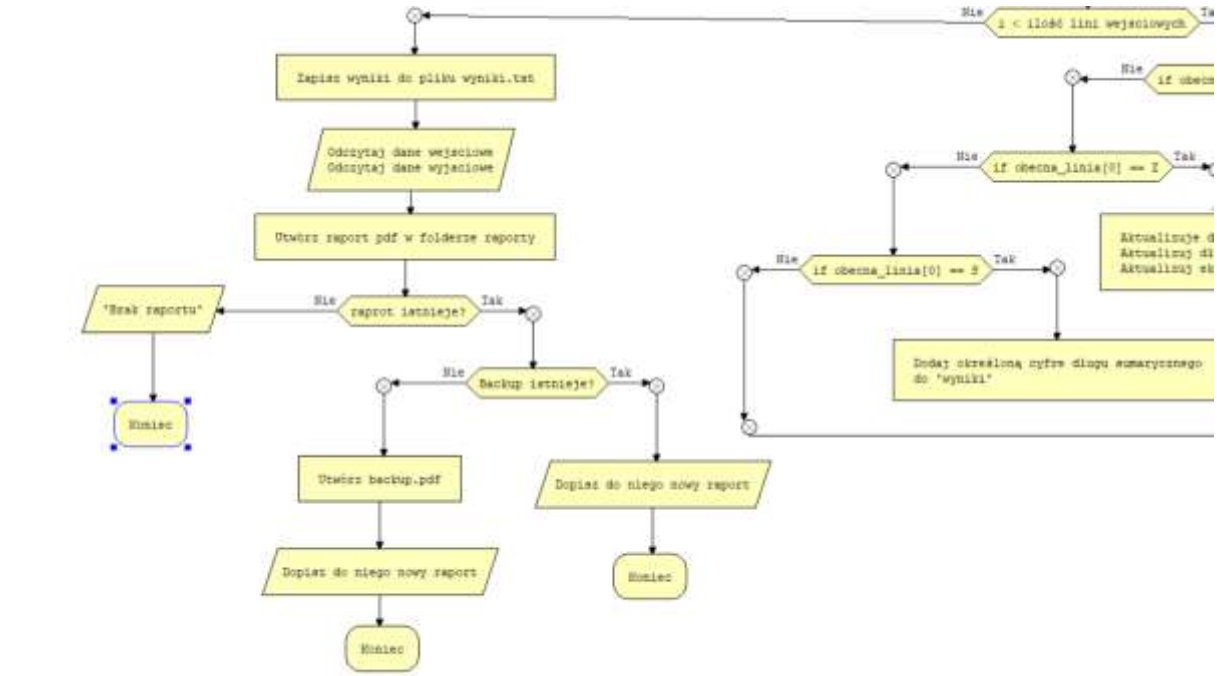
W powiększeniu:



ZDJ.1


i:=3
i:=i+1
Nie
i < ilość lini wejsciowych
Tak
Nie
if obecna_linia[0] == W
Tak
Aktualizuje dlug wewnętrzny
Aktualizuj dlug sumaryczny
Aktualizuj ekran
Nie
if obecna_linia[0] == Z
Tak
Aktualizuje dlug zewnętrzny
Aktualizuj dlug sumaryczny
Aktualizuj ekran
Nie
if obecna_linia[0] == S
Tak
Dodaj określoną cyfre długu sumarycznego
do 'wyniki'
iego nowy raport

ZDJ.2


Zapisz wyniki do pliku wyniki.txt
Odczytaj dane wejsciowe
Odczytaj dane wyjsciowe
Utwórz raport pdf w folderze raporty
'Brak raportu'
Nie
raport istnieje?
Tak
Koniec
Nie
Backup istnieje?
Tak
Utwórz backup.pdf
Dopisz do niego nowy raport
Koniec
Dopisz do niego nowy raport
Nie
i < ilość lini wejsciowych
Nie
if obecna_linia[0] == I
Tak
Nie
if obecna_linia[0] == S
Tak
Dodaj określoną cyfre długu sumarycznego
do 'wyniki'

ZDJ.3

4. Instrukcja użytkownika.

INSTRUKCJA:

Program składa się z 3 plików Python (Zadanie, Raport i Backup) i jednego pliku Batch, który steruje całym projektem.

Aby uruchomić program wystarczy nacisnąć dwukrotnie na plik Batch "BAT EXE wykonujący". Następnie powinno się postępować zgodnie z wyświetlanymi instrukcjami.

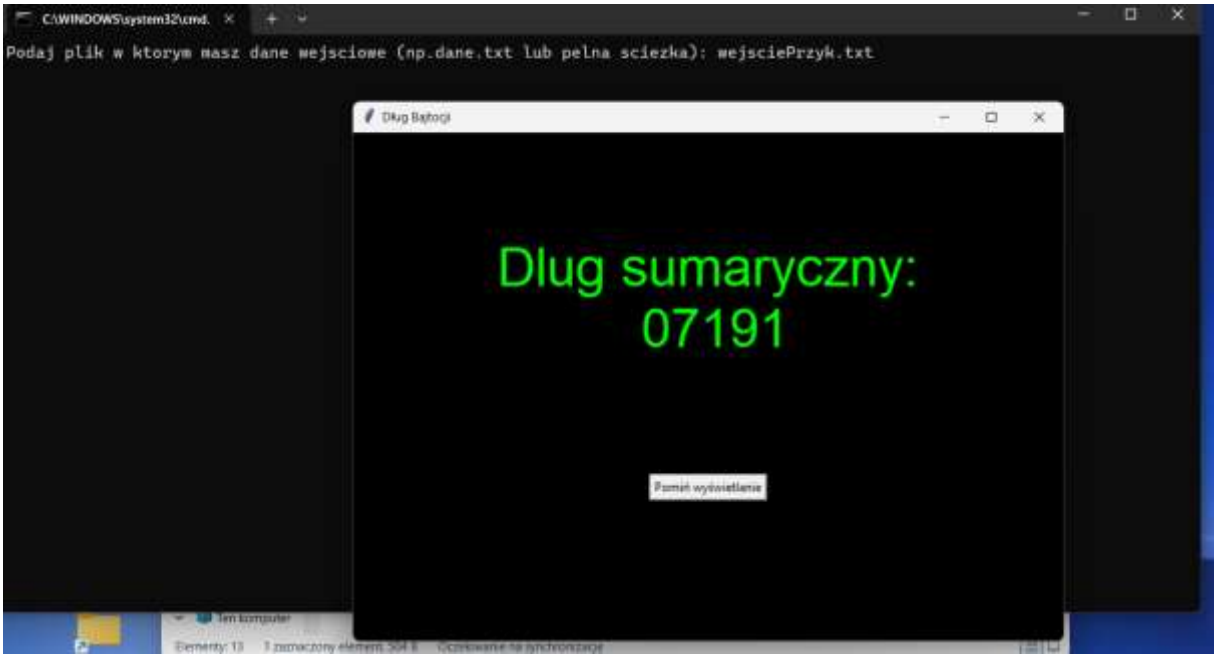
Aby dodać nowe dane wejściowe dla programu należy zapisać je w odpowiedniej formie (zgodnej z tą która jest w pliku wejsciePrzyk.txt) w pliku txt i najwygodniej jest ten plik umieścić w folderze projektu.

Jeśli pojawiłby się problem z działaniem programu dodałem na wszelki wypadek również w folderze projektu plik Batch "Instalator modulow potrzebnych". Jest to plik, który automatycznie zainstaluje potrzebne do działania programu biblioteki. Wystarczy kliknąć dwukrotnie na niego.

Program działa w oparciu o zadanie z XXVIII Olimpiady informatycznej "Licznik długu". W skrócie polega ono na tym iż otrzymujemy dane o długu wewnętrznym pewnego państwa i jego długu zagranicznym i mamy zbudować oprogramowanie do wyświetlacza długu sumarycznego. Jako dane otrzymujemy nie tylko początkowe kwoty długów ale również dostajemy informacje jak zmieniają się poszczególne pozycje/cyfry obu długów i na podstawie tych zmian wartości mamy aktualizować wyświetlacz wyświetlający dług sumaryczny.

Wykorzystując matematyczną wiedzę i zależności między liczbami oparłem algorytm w głównym pliku Zadanie.py o pisemne działania matematyczne.

Dla efektu wizualnego aby zmiany długu były lepiej widoczne postanowiłem dodać moduł Tkinter który pozwala na wyświetlenie w formie graficznej wskaźnika długu sumarycznego.



Przykład poprawnego wyświetlania programu.

5. Podsumowanie i wnioski.

Podsumowując udało się zrealizować wszystko co było zamierzone i dodać interfejs graficzny z modułu Tkinter. Projekt mimo iż początkowo wydawał się skomplikowany tak naprawdę okazał się przyjemny w tworzeniu, a zasługą tego z pewnością była przyjazna dla programisty skladnia pythona która jest po prostu bardzo zbliżona do języka angielskiego. Wyznanie stanowiły indeksacje ponieważ indeksacja przyjęta w zadanie (1 od cyfry jedności) i indeksacja w Pythonie (0 od początku) mocno różnią się o siebie, jednak ostatecznie udało się to pogodzić i zapewnić poprawne działanie programu. Drugą monotonną kwestią było zorganizowanie całej pracy w pliku batch tak żeby cały projekt działał bezbłędnie według mojego zamysłu i to na szczęście również udało się zrealizować.

Program w przyszłości może być podatny na takie rozszerzenia jak na przykład: zmiana całej treści pliku zadanie.py aby wykonać system raportujący dla innego powierzonego zadania lub rozbudowa interfejsu graficznego, gdzie tak naprawdę ogranicza nas tylko wyobraźnia.

Autor: Wojciech Grzywocz