

Języki Skryptowe

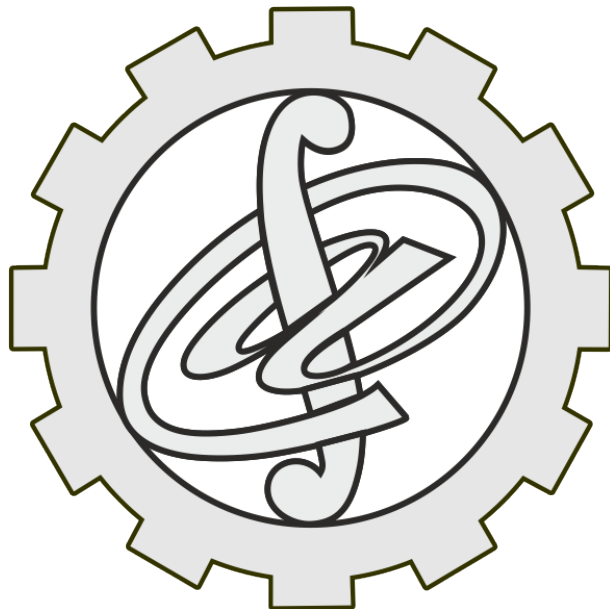
Projekt Zaliczeniowy - Algorytmion 2011

Zadanie "Monety"

Politechnika Śląska
Wydział Matematyki Stosowanej

Radosław Terelak

Gliwice, 2022



Spis treści

1	Treść zadania	2
1.1	Treść i problematyka	2
1.2	Dane	2
2	Algorytm	3
2.1	Opis słowny	3
2.2	Schemat blokowy	3
3	Implementacja	4
3.1	Przepływ danych	4
3.2	ProjektTerelak.py	5
3.3	Backup.py	8
3.4	Menu.bat	11
4	Podsumowanie	12
4.1	Wnioski i wrażenia	12

1 Treść zadania

1.1 Treść i problematyka

Mój projekt zaliczeniowy to rozwiązanie zadania "Monety" z **Algorytmionu 2011**.

Treść zadania:

Na placu targowym umieszczono **X worków**, w których w każdym z nich znajdowało się **Y złotych monet**. Każda złota moneta waży **3 gramy**. Falszerz monet podmienił jeden worek, zamieniając w nim wszystkie monety na podrobione. Falszywa moneta waży **2 gramy**.

Sprzedawca chce znaleźć worek z fałszywymi monetami. Może to zrobić tylko w ten sposób, że wyciąga dowolną ilość monet z każdego worka i waży je złożone razem tylko jeden raz.

Program wczytuje ze zbioru **dane.txt** ilość worków znajdującą się na targu, ilość monet wyciągniętych przez sprzedawcę kolejno ponumerowanych worków oraz wagę wszystkich wyciągniętych monet.

Na podstawie wagi stwierdza, w który worek znajdują się fałszywe monety i zapisuje numer worka z fałszywymi monetami do pliku **wynik.txt** albo w przypadku nie możliwości określenia numeru worka wpisuje komunikat "**dokonaj wyciągnięcia monet jeszcze raz**".

Wejście: Pierwszy wiersz zawiera liczbę całkowitą określającą ilość worków i zakończoną znakiem nowej linii. Druga linia zawiera **n liczb całkowitych** i każda z nich jest ilością wyciągniętych monet przez sprzedawcę z ponumerowanych worków od 1 do X. W ostatnim wierszu znajduje się **waga** wszystkich wyciągniętych monet, prawdziwych i fałszywych.

Wyjście: Liczba typu całkowitego określająca numer worka z fałszywymi monetami albo napis "dokonaj wyciągnięcia monet jeszcze raz".

Problematyka:

Problem ukazany w zadaniu to określenie fałszywości worka na podstawie jego wagi. Aby go rozwiązać należy na początku policzyć wagę, gdyby wszystkie wyciągnięte monety były prawdziwe. Pozwoli to określić czy istnieje fałszywy worek. Następnie należy ważyć monety z każdego kolejnego worka wagą monety fałszywej i dodając wagę pozostałych monet o wadze monety złotej. Jeśli suma tych wag jest równa wartości zapisanej w pliku, worek fałszywy został znaleziony.

1.2 Dane

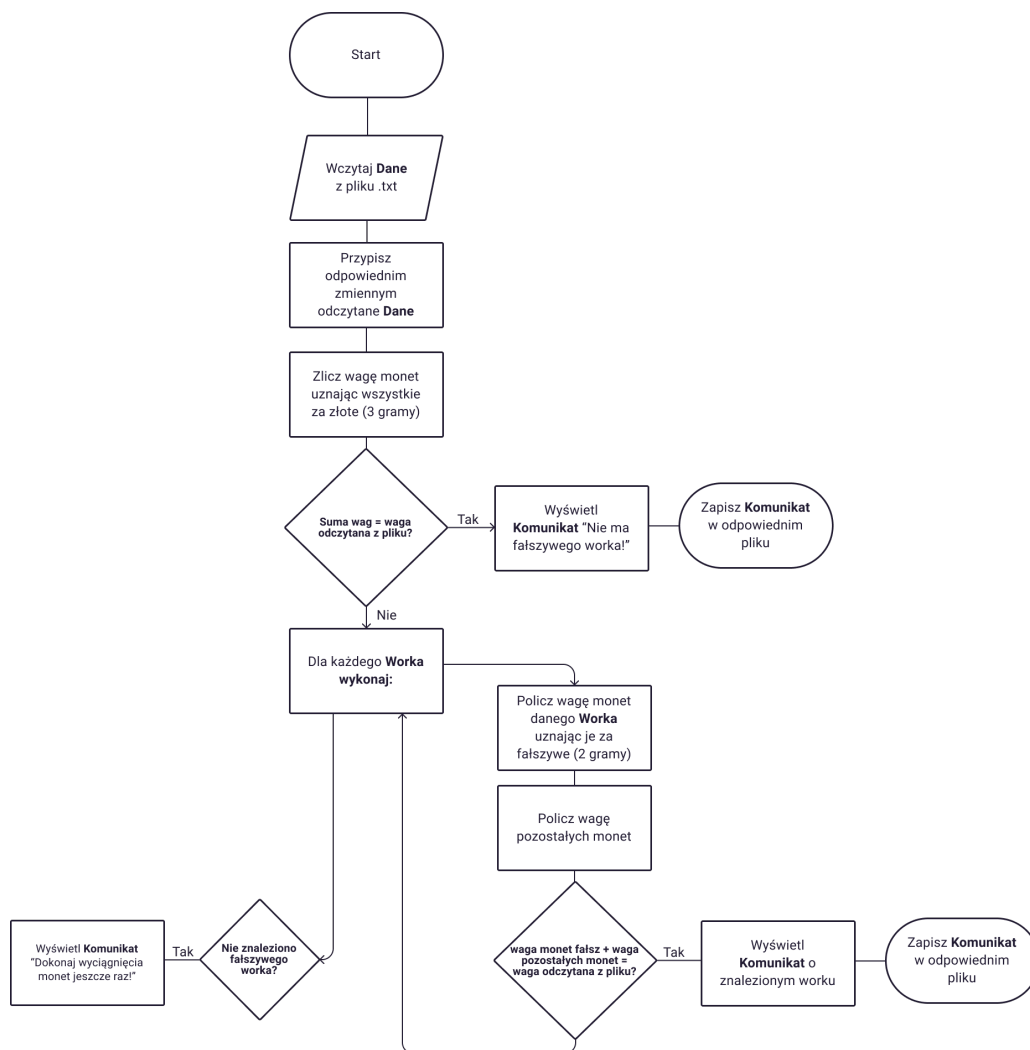
Jako dane wejściowe przygotowałem 9 plików. Pierwsze 3 zawierają fałszywy worek. Kolejny plik zawiera dane, w którym nie istnieje fałszywy worek. Następne zawierają błędy, a ostatni posiada taki zestaw danych, który uniemożliwia określenie worka fałszywego. Jeśli istnieje worek fałszywy lub można określić, że nie istnieje zostaje utworzony plik wyjściowy z odpowiednim komunikatem.

2 Algorytm

2.1 Opis słowny

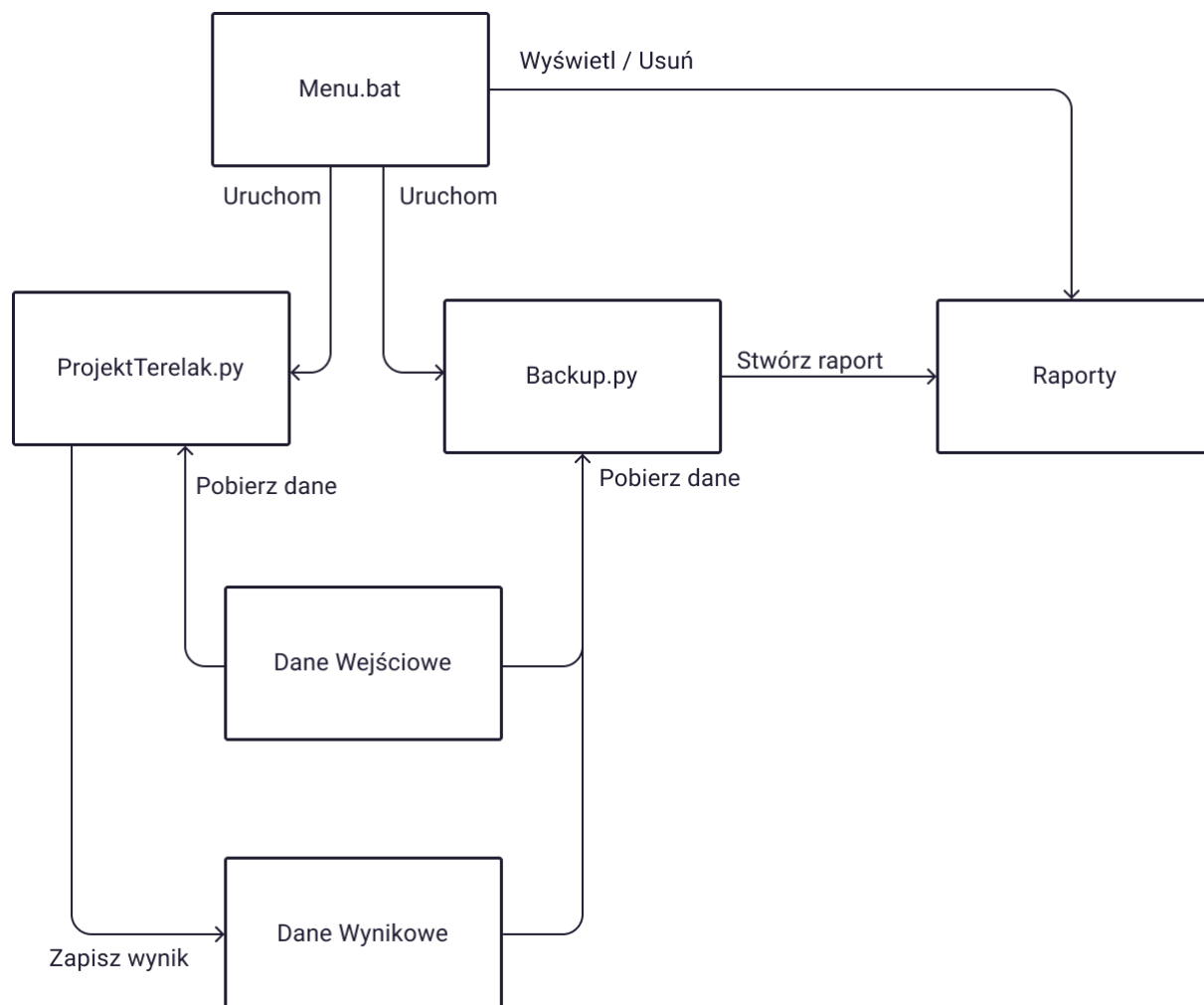
1. Wczytaj dane z pliku .txt.
2. Jeśli dane są zgodne: odpowiednim zmiennym przypisz wartości z pliku.
3. Sprawdź czy w podanych danych zawiera się worek fałszywy. Jeśli nie: wyświetl odpowiedni komunikat
4. Dla każdego worka:
 - Policz wagę monet danego worka uznając je za fałszywe.
 - Policz wagę pozostałych monet.
 - Porównaj sumę tych wartości z wagą odczytaną z pliku.
5. Jeśli znaleziono worek fałszywy: zapisz odpowiedni komunikat do pliku wynikowego.
6. Jeśli nie udało się znaleźć worka fałszywego: wyświetl komunikat aby dokonać wyciągnięcia monet jeszcze raz.

2.2 Schemat blokowy



3 Implementacja

3.1 Przepływ danych



3.2 ProjektTerelak.py

Plik **ProjektTerelak.py** zawiera implementację algorytmu. Znajdują się w nim 4 funkcje.

Funkcja START:

Jest to funkcja inicjująca program. Na początku sprawdza czy istnieje odpowiedni katalog wynikowy. Jeśli nie istnieje to go tworzy. Następnie przechodzi do folderu z danymi wejściowymi i inicjuje algorytm dla wszystkich plików zawierające dane. Jeśli plik z danymi wejściowymi nie istnieje na ekranie wyświetli się adekwatny komunikat.

Funkcja wczytajDane:

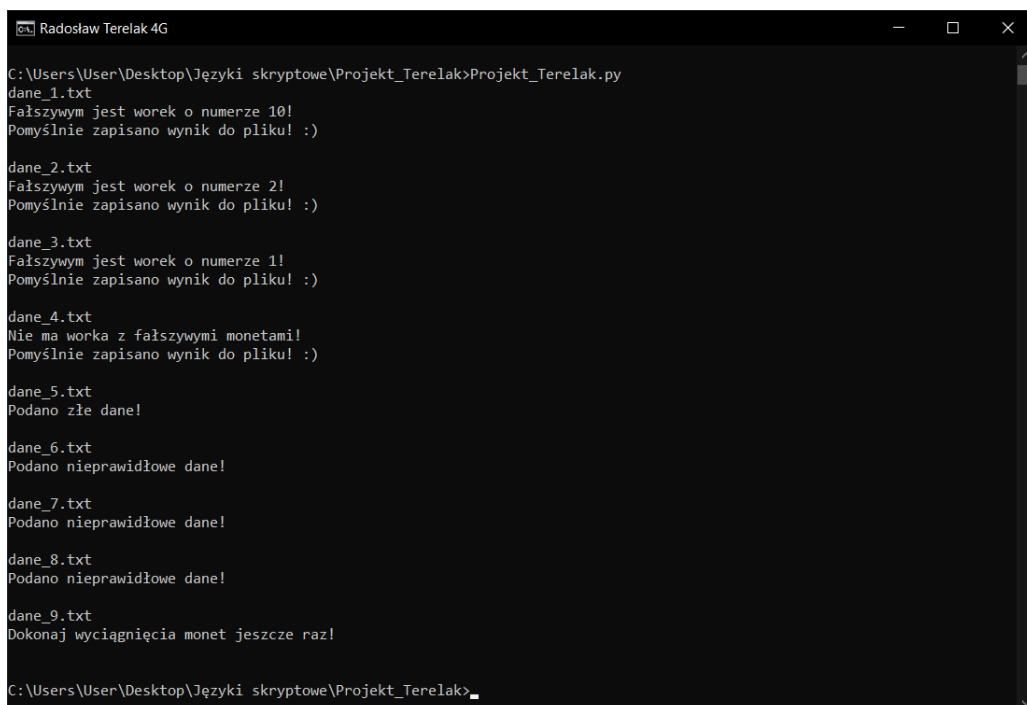
Zadaniem tej funkcji jest odczytanie danych z pliku wejściowego. Następnie tworzy odpowiednie zmienne, w których odczytane dane będą przekazywane. Sprawdza poprawność danych np. poprzez weryfikację czy ilość worków jest równa ilości podanych wyciągnięć monet.

Funkcja sprawdzDane:

Ta funkcja analizuje wczytane dane i szuka worka fałszywego. Na początku sprawdza, czy w ogóle worek fałszywy występuje porównując wczytaną wagę łączną monet z rzeczywistą wagą monet przyjmując, że każda jest złota. Jeśli może wystąpić sprawdza każdy kolejny worek i porównuje wartości wagowe. Po analizie danych wyświetla odpowiedni komunikat. Jeśli znaleziono worek fałszywy, komunikat zostaje zapisany w odpowiednim pliku wynikowym.

Funkcja zapiszWynik:

Ta funkcja tworzy odpowiedni plik wynikowy, którego numer odpowiada numerowi pliku wejściowego. Zapisuje w nim komunikat przekazany przez funkcję **sprawdzDane**. Wyświetla informację o pomyślnym, bądź nieudanym zapisie danych.



```
C:\Users\User\Desktop\Języki skryptowe\Projekt_Terelak>Projekt_Terelak.py
dane_1.txt
Fałszywym jest worek o numerze 10!
Pomyślnie zapisano wynik do pliku! :)

dane_2.txt
Fałszywym jest worek o numerze 2!
Pomyślnie zapisano wynik do pliku! :)

dane_3.txt
Fałszywym jest worek o numerze 1!
Pomyślnie zapisano wynik do pliku! :)

dane_4.txt
Nie ma worka z fałszywymi monetami!
Pomyślnie zapisano wynik do pliku! :)

dane_5.txt
Podano złe dane!

dane_6.txt
Podano nieprawidłowe dane!

dane_7.txt
Podano nieprawidłowe dane!

dane_8.txt
Podano nieprawidłowe dane!

dane_9.txt
Dokonaj wyciągnięcia monet jeszcze raz!

C:\Users\User\Desktop\Języki skryptowe\Projekt_Terelak>
```

ProjektTerelak.py

```

1 # Języki Skryptowe - Projekt zaliczeniowy Radosław Terelak 4G
2 # Algorytmion - 2011 - Zadanie 1 "Monety"
3
4 import os, glob
5
6 folder_dane = "IN"
7 folder_wynik = "OUT"
8
9 def wczytajDane(nazwa_pliku): # Wczytaj dane z pliku wejściowego i sprawdź
                                # ich poprawność
10     with open(nazwa_pliku, "r") as plik:
11         try:
12             numer_pliku = nazwa_pliku[5:6]
13             ile_workow = int(plik.readline())
14             monety = plik.readline().split(";")
15             waga = int(plik.readline())
16
17             if(len(monety) != ile_workow):
18                 raise ValueError
19             sprawdzDane(ile_workow, monety, waga, numer_pliku)
20         except ValueError:
21             print("Podano złe dane!\n")
22
23 def sprawdzDane(ile_workow, monety, waga, numer_pliku): # Przeanalizuj dane
                                                            # , znajdź fałszywy worek
24     suma_wag = 0
25     znaleziono = False
26     try:
27         for i in range(ile_workow):
28             if int(monety[i]) <= 0:
29                 raise ValueError
30             suma_wag += int(monety[i]) * 3
31
32     if(suma_wag == waga):
33         print("Nie ma worka z fałszywymi monetami!")
34         return zapiszWynik("Nie ma worka z fałszywymi monetami!",
                             numer_pliku)
35
36     for i in range(ile_workow):
37         waga_falsz = int(monety[i]) * 2
38         waga_pozostale = 0
39         nr_worka = i + 1
40
41         for j in range(ile_workow):
42             if(j != i):
43                 waga_pozostale += int(monety[j]) * 3
44
45         if(waga_falsz + waga_pozostale == waga):
46             znaleziono = True
47             print("Fałszywym jest worek o numerze {}".format(nr_worka)
                    )
48             return zapiszWynik("Fałszywym jest worek o numerze: " + str
                                (nr_worka),

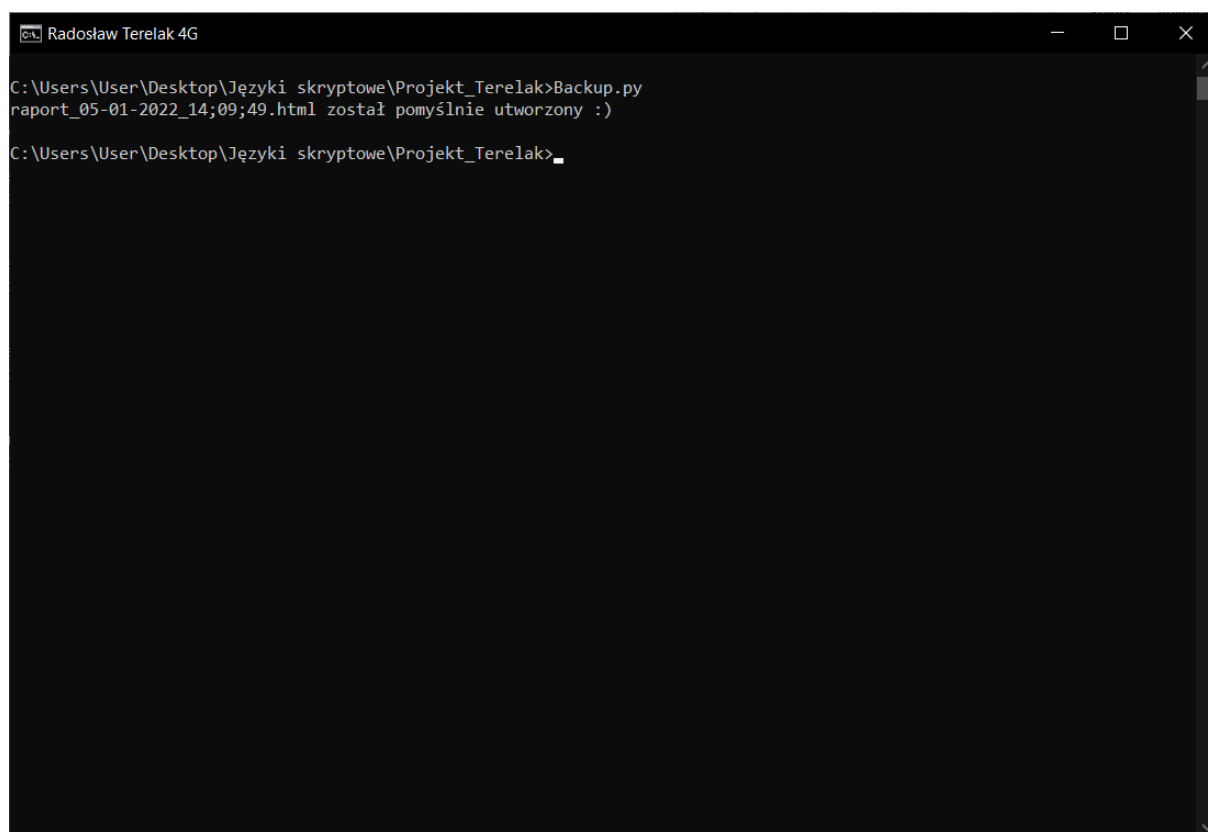
```

```

49                                     numer_pliku)
50         if znaleziono == False:
51             print("Dokonaj wyciagniecia monet jeszcze raz!\n")
52     except ValueError:
53         print("Podano nieprawidlowe dane!\n")
54
55 def zapiszWynik(komunikat, numer_pliku): # Zapis wynikow do odpowiednich
56                                         plikow w folderze wynikowym
57     plik_wynikowy = "\wynik_" + numer_pliku + ".txt"
58     with open("../" + folder_wynik + plik_wynikowy, "w") as plik:
59         try:
60             plik.write(komunikat)
61             print("Pomyslnie zapisano wynik do pliku! :)\n")
62         except:
63             print("Nie udalo sie zapisac wyniku do pliku :(\n")
64
65 def START(): # Tworzenie folderu wynikowego oraz szukanie plikow z danymi
66     if os.path.exists(folder_dane):
67         if not os.path.exists(folder_wynik):
68             os.mkdir(folder_wynik)
69
70     os.chdir("../" + folder_dane)
71     for plik in glob.glob("dane_?.txt"):
72         print(plik)
73         wczytajDane(plik)
74     else:
75         print("Folder z danymi nie istnieje!")
76
77 START()
```


3.3 Backup.py

Plik **Backup.py** służy do tworzenia raportów HTML z utworzonych plików wynikowych. Pobiera aktualny czas i datę, która zostanie zawarta w nazwie raportu. Zawiera wstępny szablon strony HTML. Otwiera istniejące pliki wynikowe oraz pliki wejściowe i odczytuje dane i dopisuje do kodu HTML. Po przejrzaniu wszystkich plików tworzy plik raportu w folderze **BACKUP**.



```
Radosław Terelak 4G
C:\Users\User\Desktop\Języki skryptowe\Projekt_Terelak>Backup.py
raport_05-01-2022_14;09;49.html został pomyślnie utworzony :)
C:\Users\User\Desktop\Języki skryptowe\Projekt_Terelak>
```

Backup.py

```

1  from datetime import datetime
2  import os, glob
3
4  czas = datetime.now()
5  nazwa_pliku = "raport_" + czas.strftime("%d-%m-%Y_%H;%M;%S") + ".html"
6  data_godzina = czas.strftime("%d/%m/%Y %H:%M;%S")
7
8  folder_dane = "IN"
9  folder_wynik = "OUT"
10 folder_backup = "BACKUP"
11
12 wyniki = ["-"] * 10
13
14 html_start = f'''
15 <html>
16     <head>
17         <title>Raport {data_godzina}</title>
18         <link rel="preconnect" href="https://fonts.googleapis.com">
19         <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin
20             >
21         <link href="https://fonts.googleapis.com/css2?family=Oswald:
22             wght@200;400;600&display=swap
23             " rel="stylesheet">
24         <link rel="stylesheet" href="style.css">
25     </head>
26     <body>
27         <h1>Algorytmion 2011 - "Monety" Radoslaw Terelak 4G</h1>
28         <h1>Raport {data_godzina}</h1>
29         <table>
30             <tr>
31                 <th>Nr pliku</th>
32                 <th>Dane wejsciowe</th>
33                 <th>Wynik</th>
34             </tr>
35             '''
36
37 html_end = f'''
38     </table>
39     </body>
40 </html>
41 '''
42
43 html = html_start
44
45 os.chdir("..\\" + folder_wynik)
46 for plik in glob.glob("wynik_?.txt"):
47     with open(plik, 'r', encoding="utf-8", errors='ignore') as f:
48         wyniki[int(plik[6:7])] = f.readline()
49
50 os.chdir("..\\" + folder_dane)
51 for plik in glob.glob("dane_?.txt"):
52     nr_pliku = plik[5:6]
53     wynik = wyniki[int(nr_pliku)]

```

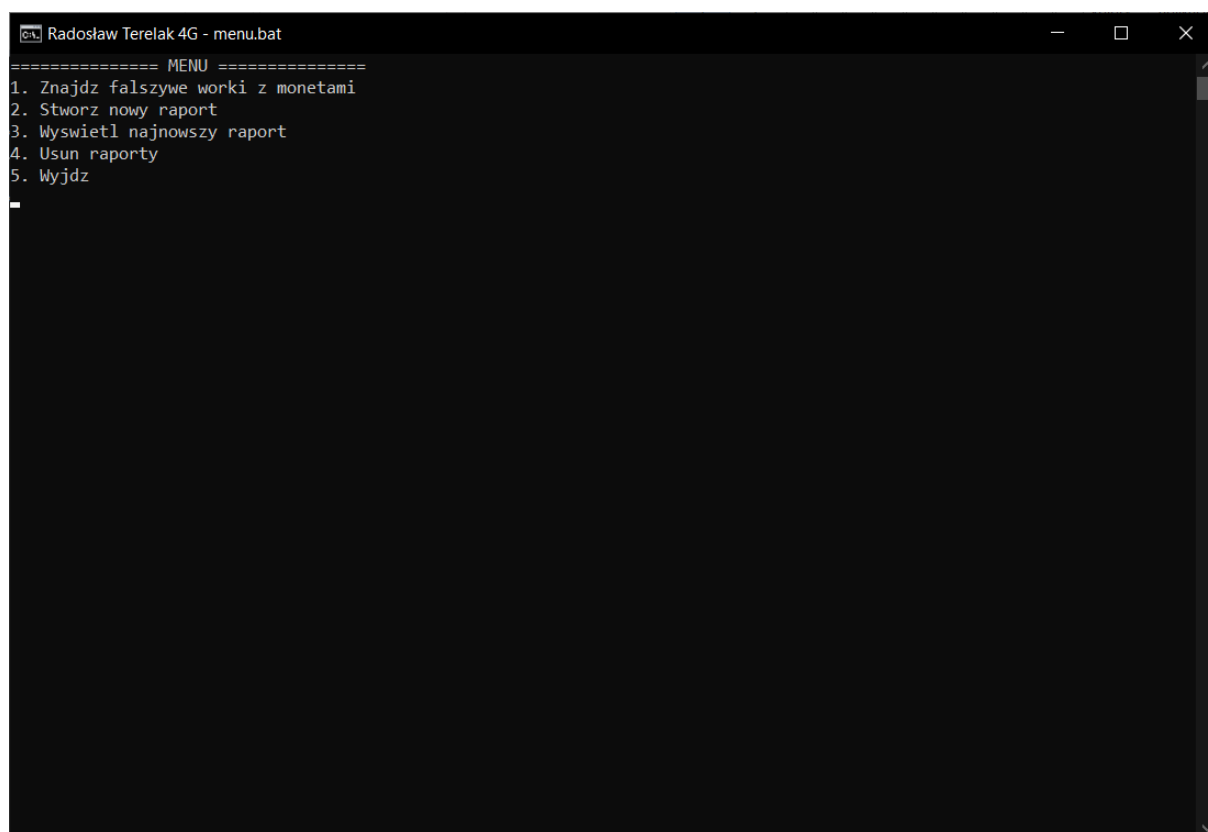
```
51 with open(plik, 'r', encoding="utf-8", errors='ignore') as f:
52     ile_workow = f.readline().strip()
53     liczba_wyciagnietych_monet = f.readline().strip()
54     waga_monet = f.readline().strip()
55     linia_4 = f.readline().strip()
56
57     kontent = f'''
58         <tr>
59             <td>{nr_pliku}</td>
60             <td>
61                 {ile_workow}<br>
62                 {liczba_wyciagnietych_monet}<br>
63                 {waga_monet}<br>
64                 {linia_4}
65             </td>
66             <td>{wynik}</td>
67         </tr>
68     '''
69     html += kontent
70
71 html += html_end
72 os.chdir("../" + folder_backup)
73 with open(nazwa_pliku, 'w', encoding="utf-8") as plik:
74     try:
75         plik.write(html)
76         print("{} został pomyslnie utworzony :)".format(nazwa_pliku))
77     except:
78         print("Nie udalo sie utworzyc raportu :(")
```

3.4 Menu.bat

Plik **Menu.bat** zawiera proste menu łączące funkcjonalności wszystkich skryptów.

Umożliwia:

- uruchomienie algorytmu wyszukującego fałszywe worki,
- utworzenie nowego raportu,
- wyświetlenie najnowszego raportu,
- usunięcie wszystkich istniejących raportów.



```
Radosław Terelak 4G - menu.bat
===== MENU =====
1. Znajdz fałszywe worki z monetami
2. Stworz nowy raport
3. Wyświetl najnowszy raport
4. Usun raporty
5. Wyjdz
-
```

4 Podsumowanie

4.1 Wnioski i wrażenia

Samo zadanie od strony logicznej nie stworzyło mi wielu trudności. Dość szybko zrozumiałem problem i przeszedłem do implementacji. Stopniowo eliminowałem błędy i dodawałem dokładniejsze zabezpieczenia. Od początku kod podzieliłem na kilka funkcji, co zwiększyło przejrzystość i ułatwiło stworzenie działającego programu. Do stworzenia raportów użyłem f-stringów, w których przygotowałem szablony kodu HTML. Uważam to za najlepsze rozwiązanie, bo nie wymaga dodatkowych zewnętrznych bibliotek i jest bardzo czytelne.

Z pewnością dzięki stworzeniu tego projektu utrwaliłem wiedzę z języka Python oraz nauczyłem się nowych rzeczy. Po wykonaniu projektu oraz wszystkich zadań czuję się w tym języku całkiem komfortowo.

Jeśli chodzi o pomysły na rozszerzenie funkcjonalności to można by zwiększyć ilość obsługiwanych plików, bo obecnie kod napisałem w sposób umożliwiający pracę na 10 plikach numerując od 0. Kolejnym pomysłem, który przychodzi mi do głowy jest dokładniejsza walidacja danych i zapisywanie w raporcie odpowiednich komunikatów o błędach.