**Języki Skryptowe**

Dokumentacja projektu Plan Metra z Olimpiady Informatycznej XXV

Michał Osiewicz, grupa 1B

Wydział Matematyki Stosowanej

Kierunek Informatyka

Semestr III

Studia Stacjonarne

17 stycznia 2021

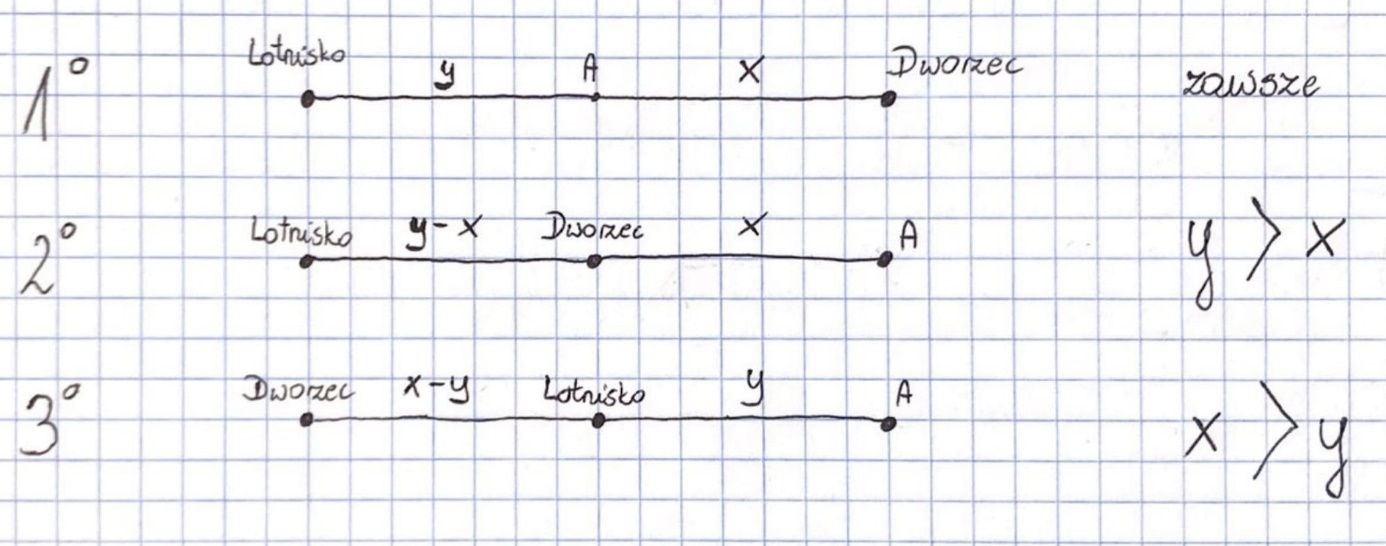
**Część I**

**Opis programu**

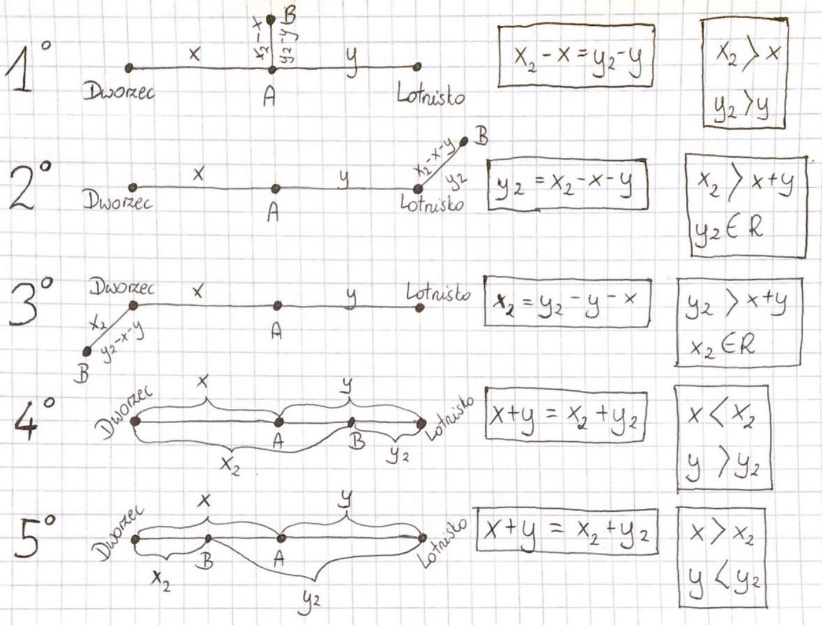
Bajtazar udał się na wycieczkę do Bajtogrodu i postanowił, że podczas całego pobytu w mieście będzie poruszał się metrem. Wysiadłszy na dworcu kolejowym (przy którym znajduje się jedna ze stacji metra), poszedł zakupić bilety w automacie. Z cennika wynikało, że przejazd na trasie prowadzącej od stacji „Dworzec kolejowy” do stacji „Lotnisko” jest darmowy, natomiast dla wszystkich pozostałych tras cena biletu jest równa odległości pomiędzy stacją początkową a stacją docelową. Dla wygody pasażerów przyjeżdżających do Bajtogrodu, przy automacie była wypisana lista cen biletów dla tras od stacji „Dworzec kolejowy” do wszystkich pozostałych stacji oraz dla tras od stacji „Lotnisko” do wszystkich pozostałych stacji. Bajtazar dowiedział się też, że metro posiada n stacji, połączonych oszczędną siecią n − 1 tuneli dodatniej długości (wystarczających jednak do przejechania z dowolnej stacji do dowolnej innej). Na podstawie tych wszystkich informacji nasz bohater chciałby wyznaczyć połączenia pomiędzy stacjami lub stwierdzić, że posiadane przez niego dane są błędne.

**Opis działania**

Program rozwiązuje zadanie wybierając pierwszą stację(A)(oprócz Dworca i Lotniska) i łączy ją ze stacją Dworzec i Lotnisko, na jeden lub dwa sposoby w zależności od odległości Dworca od stacji A(x) oraz Lotniska od stacji A(y). Algorytm jest przeprowadzany dla każdej stacji jako pierwszej(A). Poniżej prezentuje możliwe umiejscowienie stacji ze wzglądu na x i y.

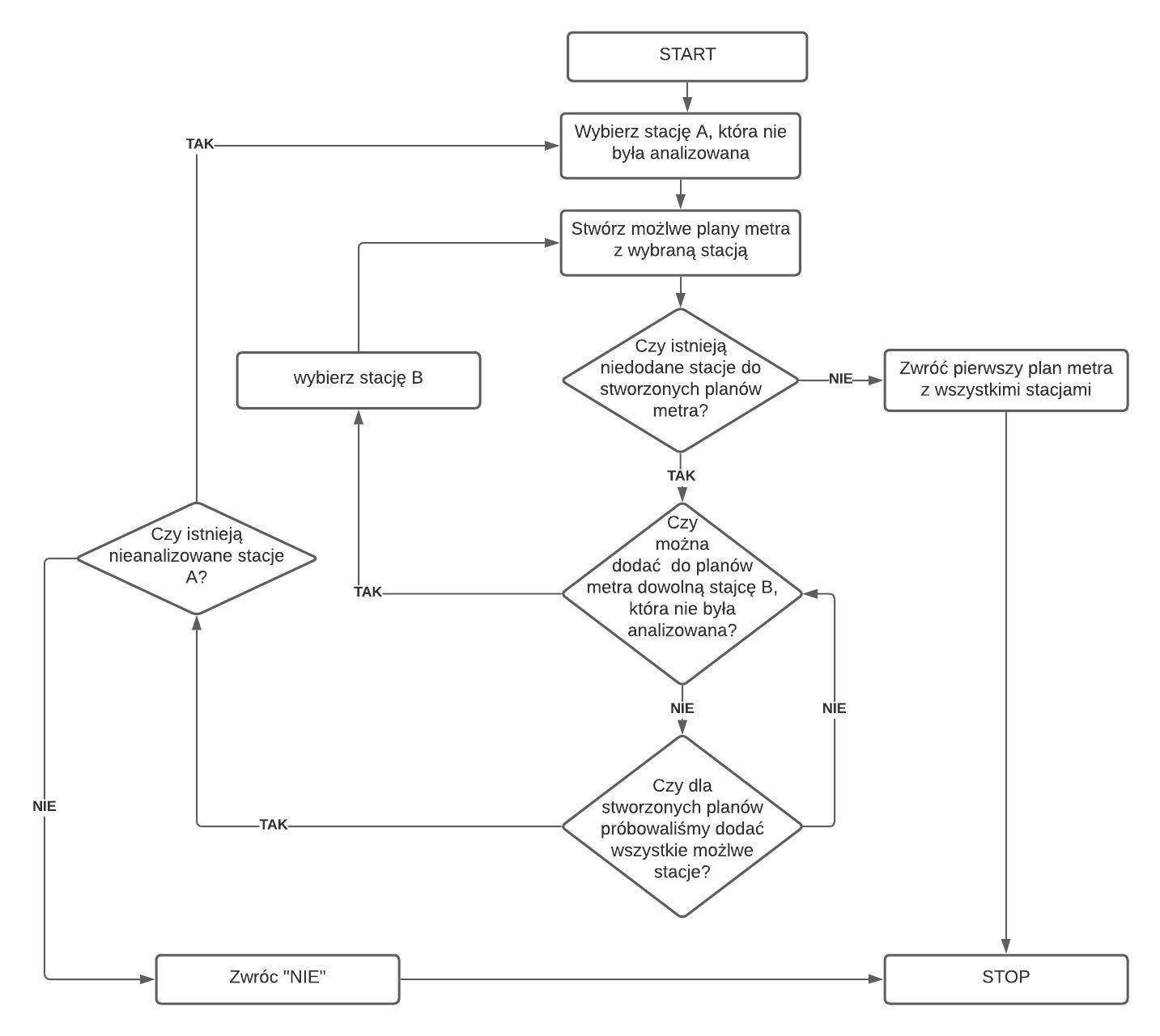


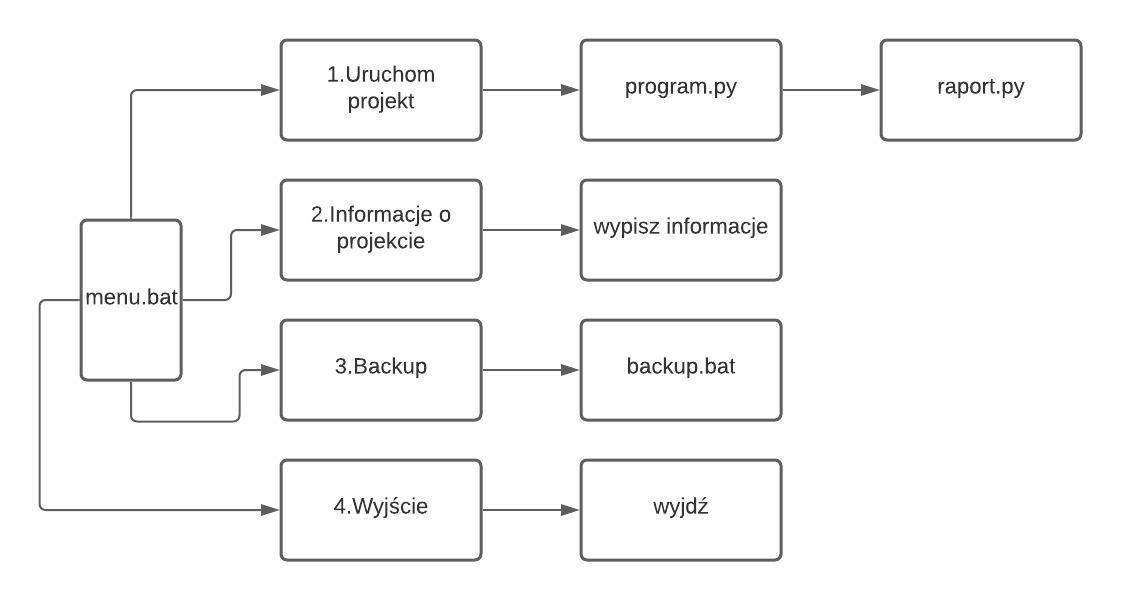
Algorytm dla utworzonego planu metra próbuje dołączyć kolejna stację(B), badając zależność między odległością Dworca od stacji B(x2) oraz Lotniska od stacji B(y2) a x i y z pierwszego kroku algorytmu. Poniżej prezentuje możliwe umiejscowienia stacji B ze względu na x,y,x2,y2.



Algorytm po udanym dodaniu stacji próbuje dołączyć kolejną stacje w taki sam sposób jak stację B, algorytm próbuje dołączyć każdą niedodaną stację na każdym kroku. Po udanym dołączeniu ostatniej stacji algorytm zwraca plan metra i kończy swoją pracę. W przypadku przeanalizowania wszystkich możliwych przypadków i braku planu metra ze wszystkimi stacjami algorytm zwraca informację o braku możliwości utworzenia planu metra z podanych danych wejściowych.

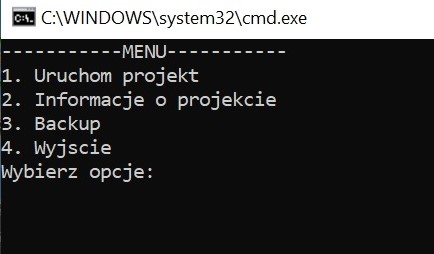
**Algorytm**

****

**Schemat działania**

**Instrukcja obsługi**

Program uruchamiamy włączając plik menu.bat. Po uruchomieniu pliku ukazuje nam się menu programu. Wybieramy jedną z czterech opcji wpisując odpowiadający im numer(1,2,3,4).

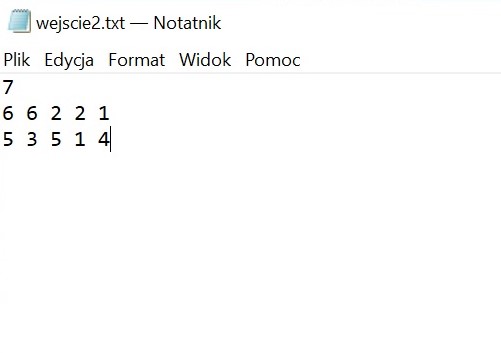


Do poprawnego działania programu należy utworzyć w katalogu „in” pliki tekstowe, nazywając je „wejsciei.txt”, gdzie i to cyfra. Dla każdego takiego pliku zostanie utworzony plik „wyjsciei.txt” w katalogu „out” z rozwiązaniem danego przykładu. Poniżej zostały opisane wytyczne jak powinien wyglądać plik wejścia oraz jak będzie wyglądał plik wyjścia.

**Wejście**

Pierwszy wiersz standardowego wejścia zawiera jedną dodatnią liczbę całkowitą n oznaczającą liczbę stacji metra w Bajtogrodzie. Stacje są numerowane liczbami od 1 do n, przy czym stacja „Dworzec kolejowy” ma numer 1, a stacja „Lotnisko” ma numer n. W drugim wierszu znajduje się ciąg n − 2 liczb całkowitych d2, d3, . . . , dn−1 z przedziału [1, 1 000 000], pooddzielanych pojedynczymi odstępami; i-ta liczba w ciągu oznacza cenę biletu na trasie od stacji „Dworzec kolejowy” do stacji o numerze i. Cena ta jest równa długości trasy. W trzecim wierszu znajduje się ciąg l2, l3, . . . , ln−1 w takim samym formacie, opisujący ceny biletów na trasach od stacji „Lotnisko”.

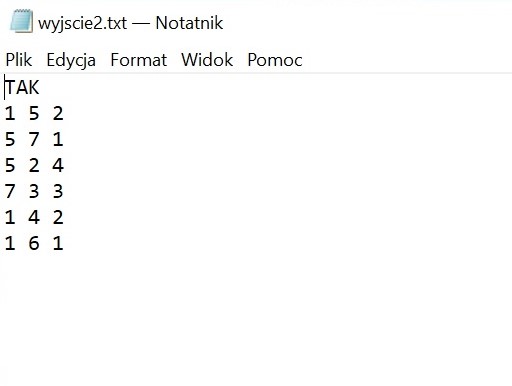
Przykład:



**Wyjście**

Jeżeli nie istnieje żaden plan połączeń pomiędzy stacjami zgodny z informacjami uzyskanymi przez Bajtazara, na standardowe wyjście należy wypisać jeden wiersz ze słowem NIE. W przeciwnym wypadku w pierwszym wierszu należy wypisać jedno słowo TAK, a w kolejnych n−1 wierszach połączenia pomiędzy stacjami. Każdy z tych wierszy powinien zawierać trzy liczby całkowite a, b i c oddzielone pojedynczymi odstępami, co oznacza, że stacje o numerach a i b są połączone tunelem o długości c. Jeżeli jest więcej niż jedna poprawna odpowiedź, Twój program powinien wypisać dowolną z nich.

Przykład:



Źródło: https://szkopul.edu.pl/problemset/problem/vvd6w7n7EXFVEg3nkqGxEirV/site/?key=statement

**Dodatkowe informacje**

Program po wykonaniu swojego zadania tworzy raport w postaci strony www. Po wybraniu opcji „Backup” w menu skrypt tworzy kopię zapasową całego folderu w którym znajduje się projekt.

System operacyjny: Windows

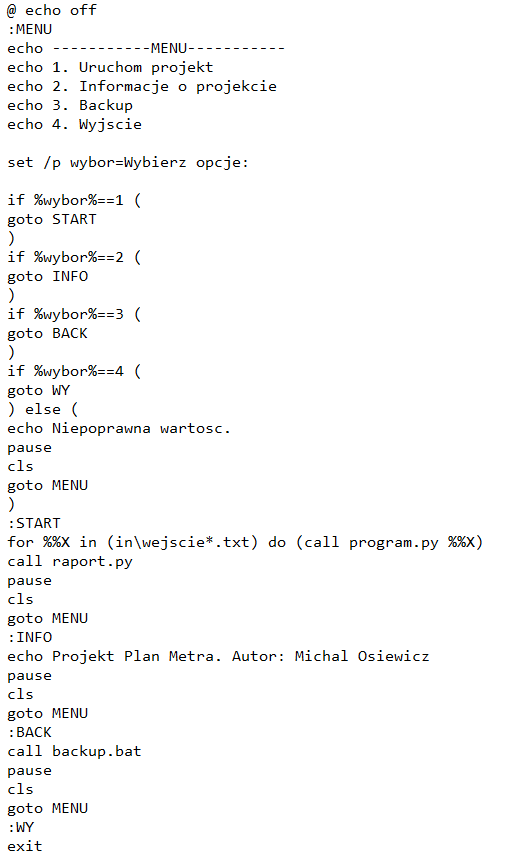
Interpreter języka python: wersja 3.9 lub nowsza

**Część II**

**Implementacja**

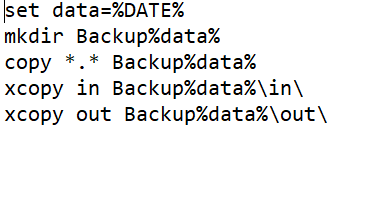
**Menu.bat**

Menu sprawdza która opcje wybrał użytkownik i wykonuje odpowiednie zadanie (schemat projektu).

****

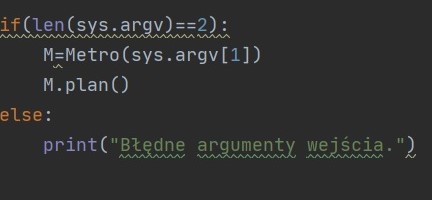
**Backup.bat**

Tworzy kopię zapasową całego folderu w którym znajduje się projekt.



**Program.py**

Wykonuje główny algorytm dla danych wejściowych i zwraca wynik. Program sprawdza czy został podany plik wejścia i wywołuje klasę Metro i jej metodę.

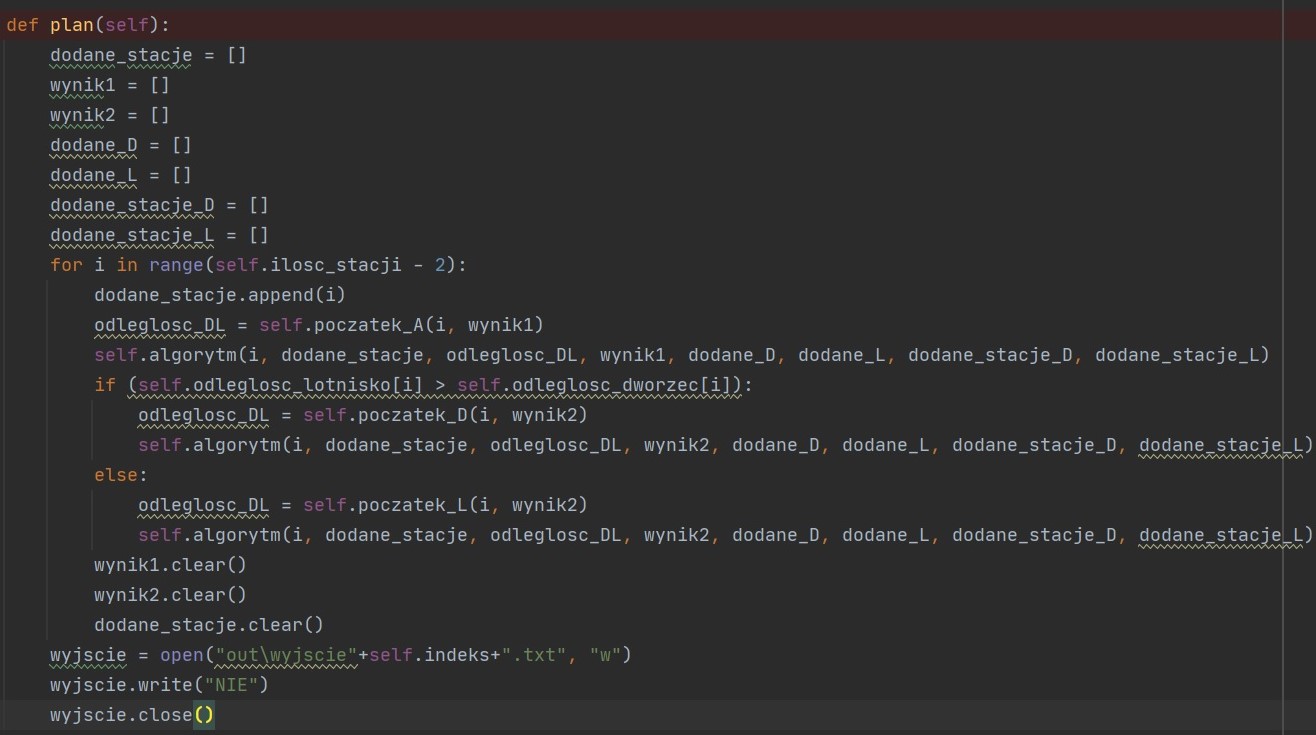


Metody klasy Metro:

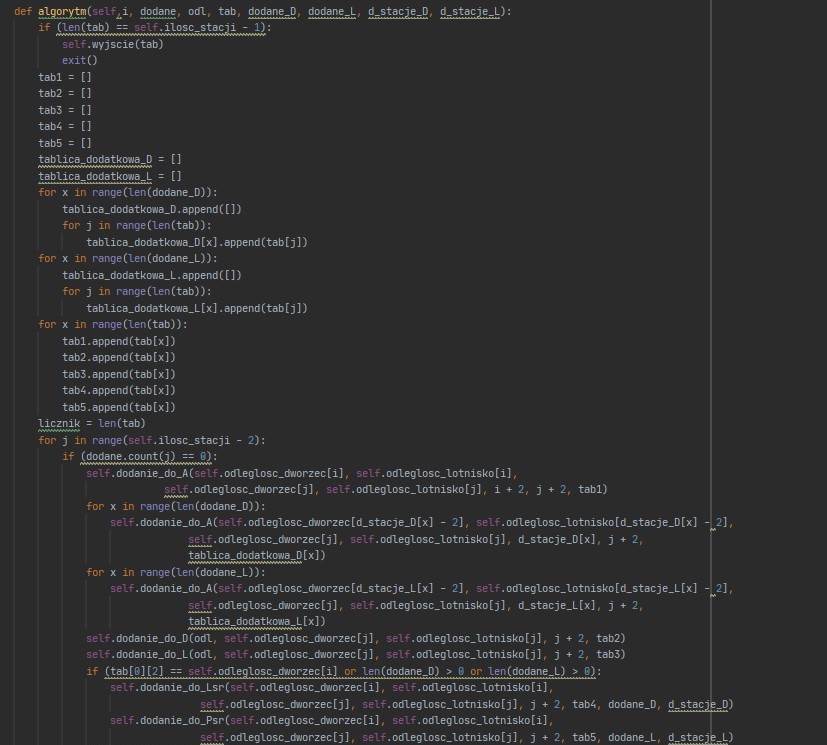
Metoda init rozpoczyna prace programu, sprawdza czy plik wejściowy istnieje, wczytuje dane z pliku wejściowego, sprawdza czy dane są poprawnie zapisane.

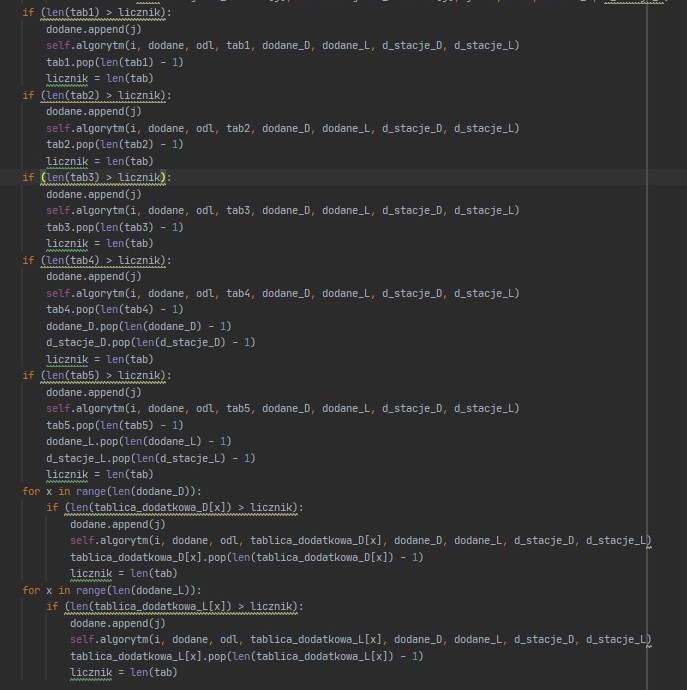


Następnie wykonuje się metoda plan która dodaje pierwsze stacje(A) na różne sposoby i tworzy plany metra. Dla każdej stacji A wywołuje metodę algorytm. W przypadku przejścia wszystkich przypadków zwraca „NIE” na plik wyjściowy.

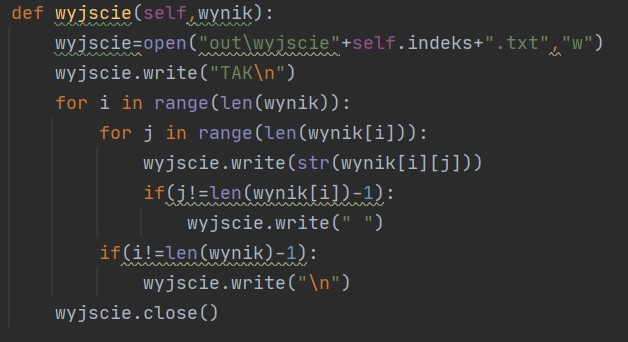


Metoda algorytm dodaje do stworzonego planu stację(B) sprawdza wszystkie możliwości oraz każda stację następnie po dodaniu stacji tworzy nowy plan metra i wywołuje sama siebie dopóki nie przeanalizuje wszystkich możliwości, jeśli znajdzie rozwiązanie wywołuje metodę wyjscie i kończy prace programu.



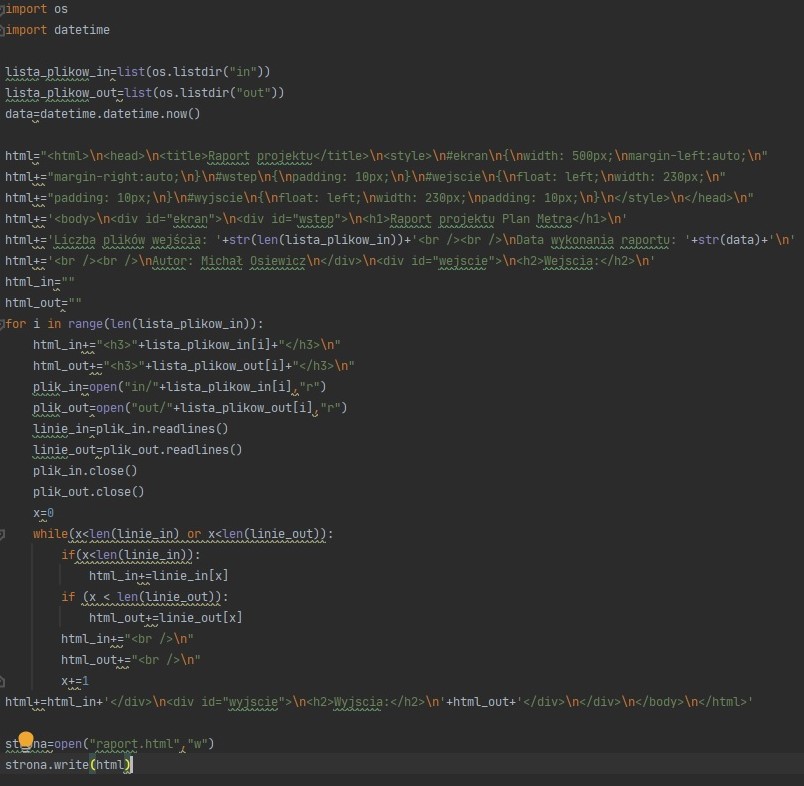


Metoda wynik zapisuje plan metra do pliku wyjściowego.



**Raport.py**

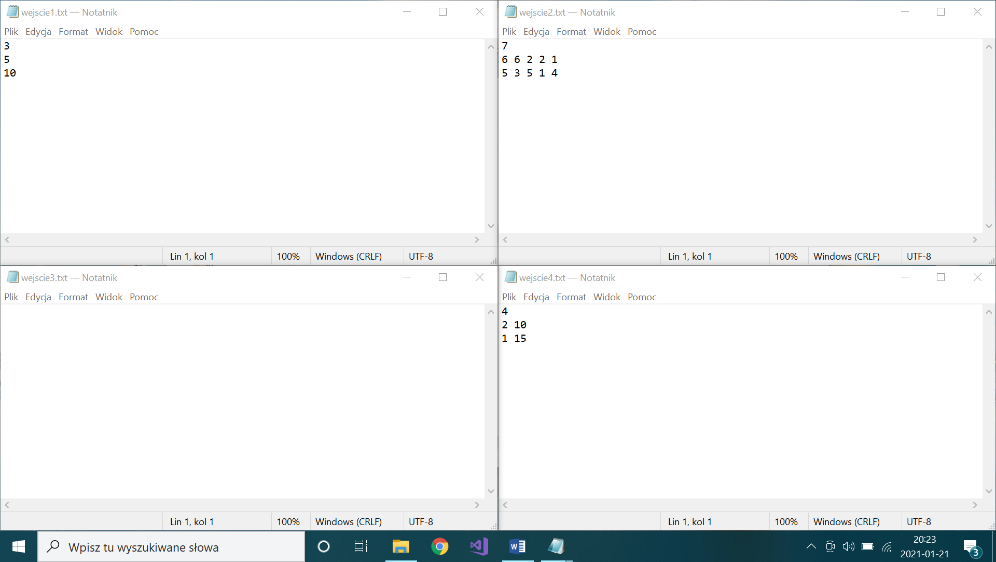
Skrypt dla danych wejściowych i uzyskanych danych wyjściowych tworzy raport w postaci prostej strony html.

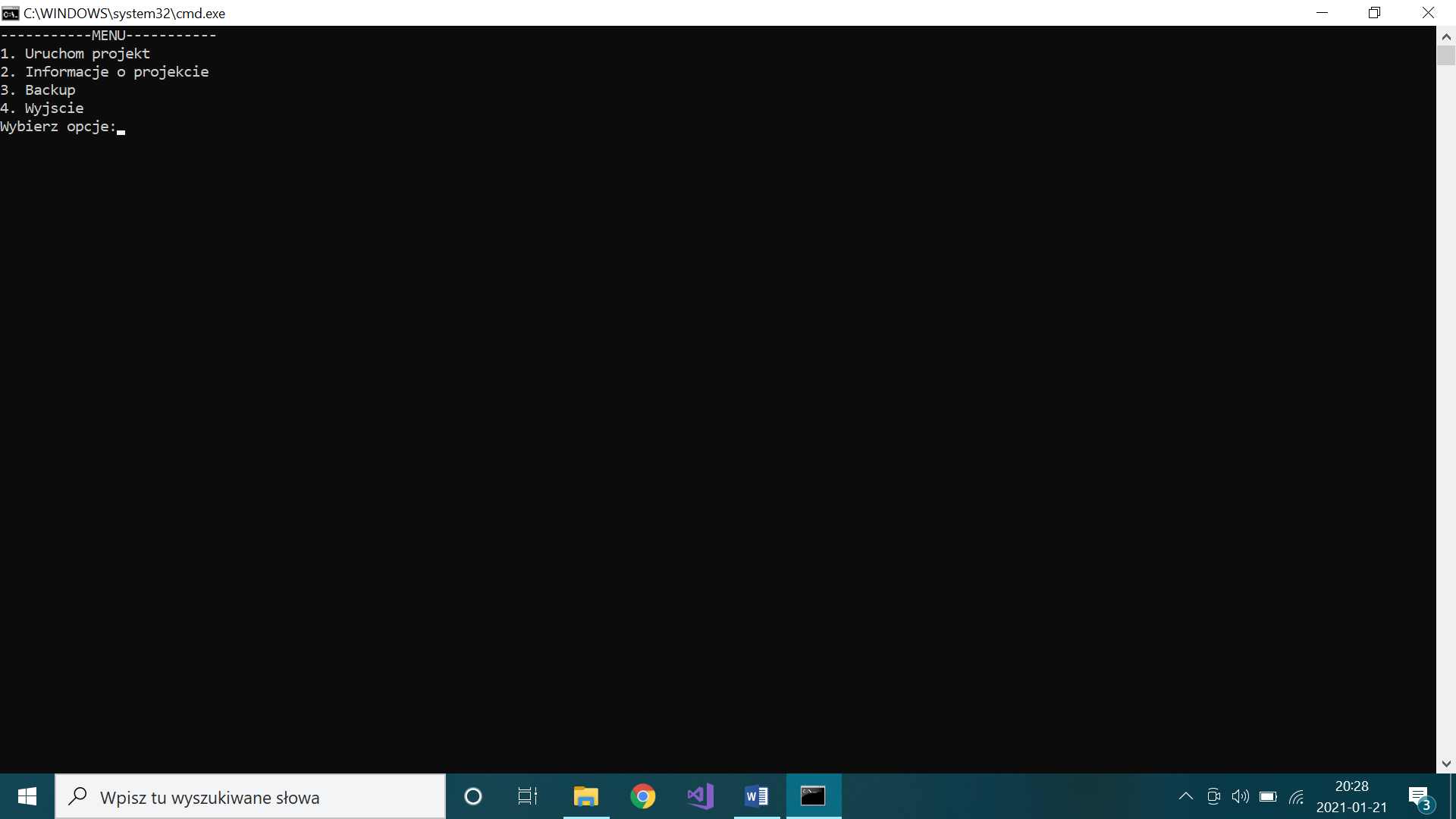


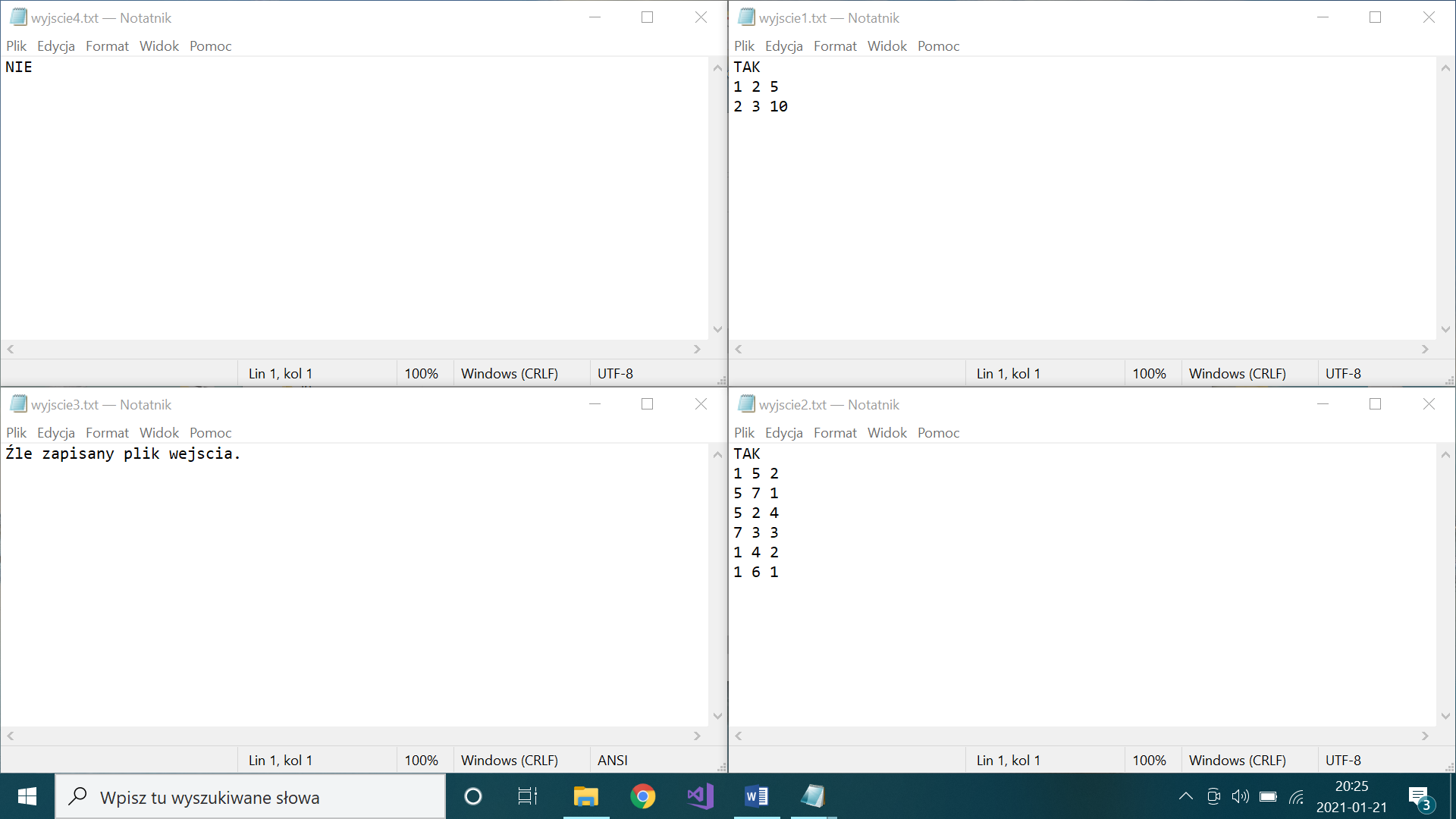
**Testy**

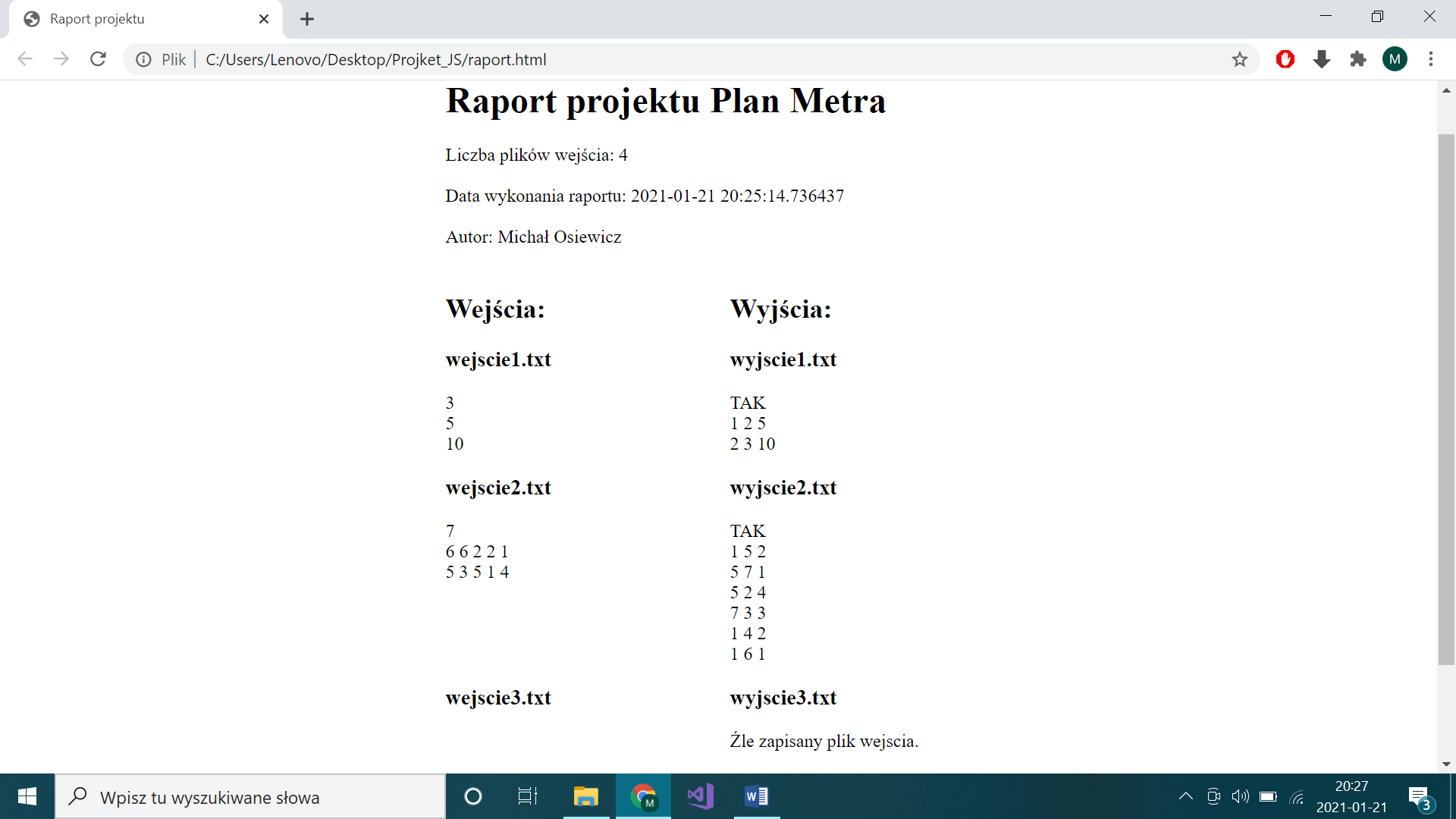
Program został przetestowany na systemie operacyjnym Windows 10 oraz interpreterze języka python 3.9 dla różnych danych zapisanych poprawnie oraz niepoprawnie.

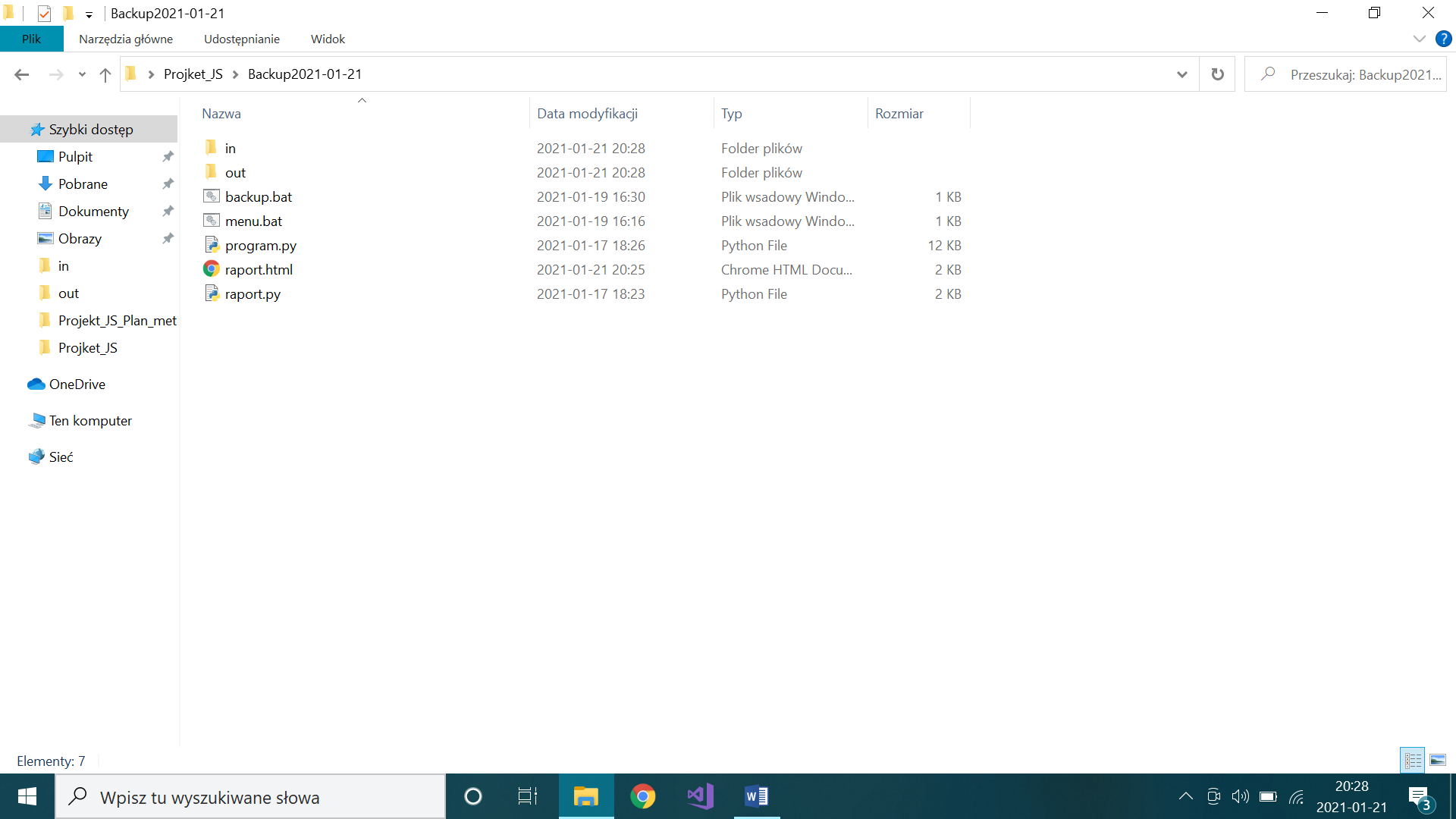
Przykład:







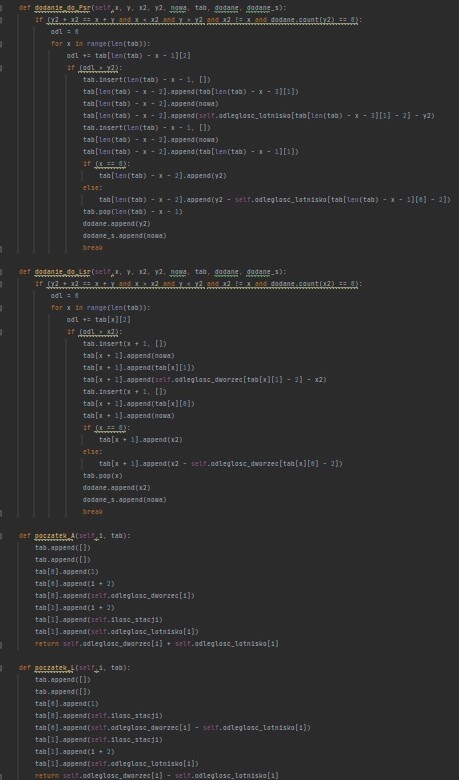


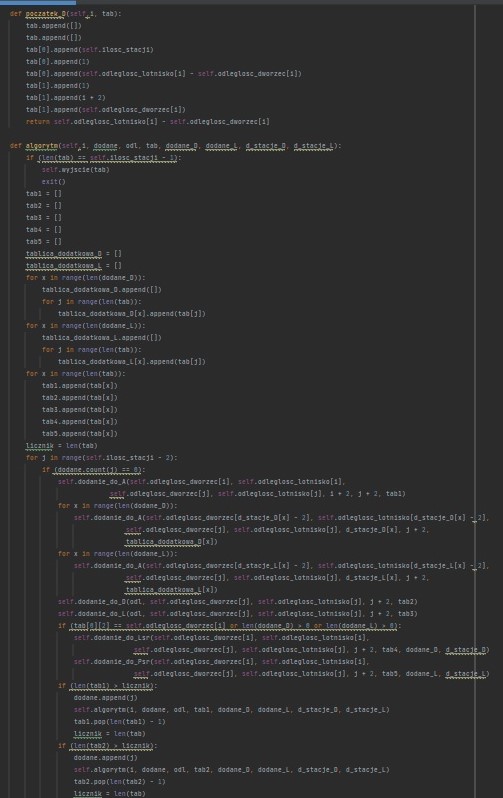


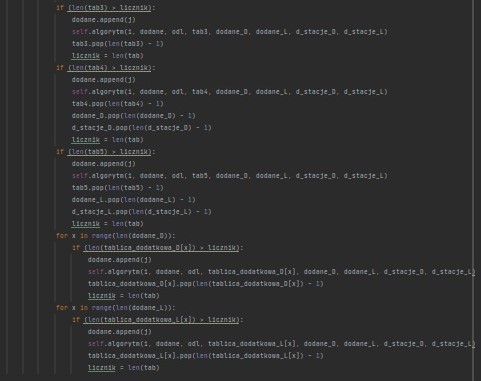
**Pełen kod aplikacji**

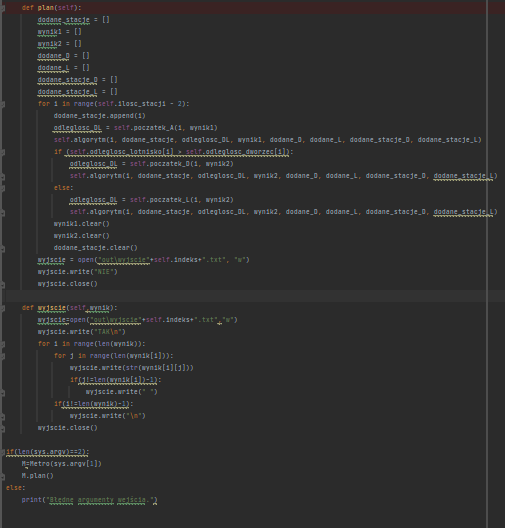
**Główny algorytm program.py**

****

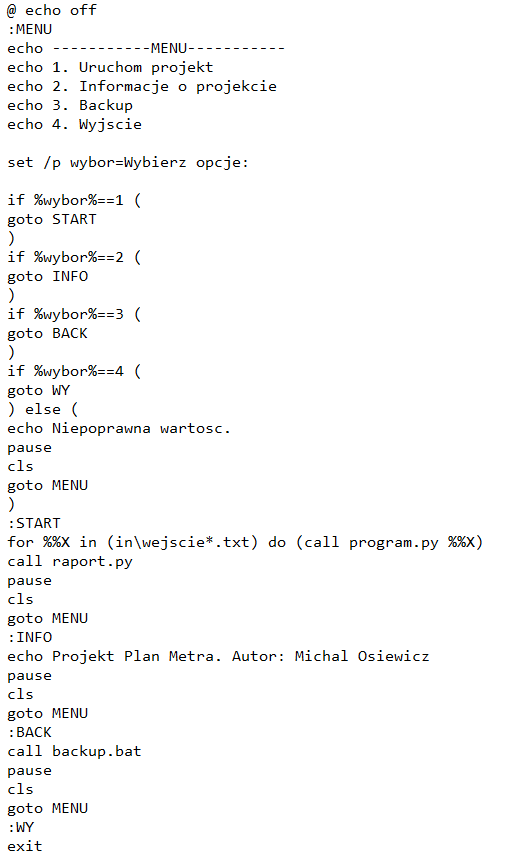
****

****

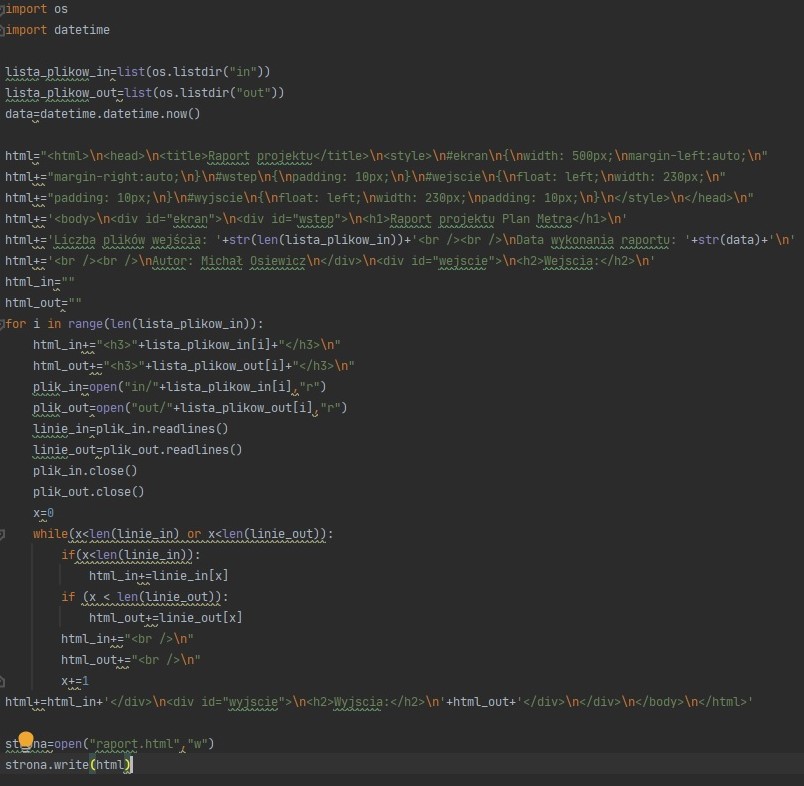
****

****

**Menu.bat**

****

**Raport.py**



**Backup.bat**

