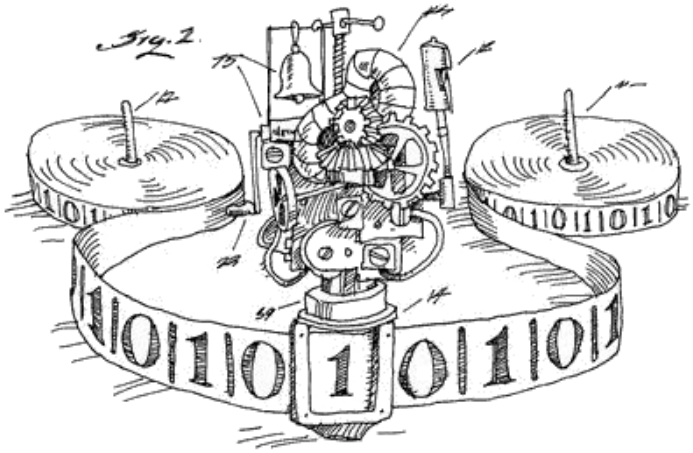
# Maszyna Turinga

## Oficjalna dokumentacja






Projektujący: Konstantin Panov

# SPIS TREŚCI



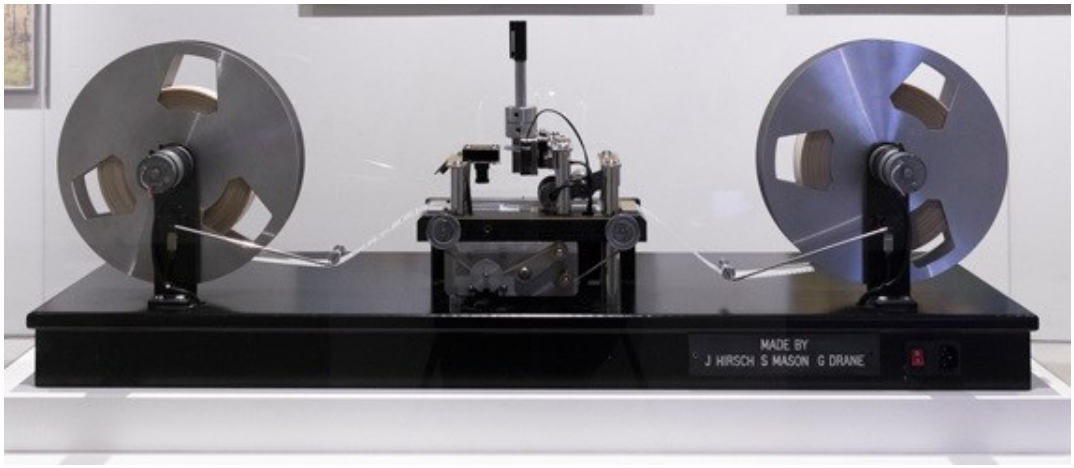
## Spis treści

ROZDZIAŁ 1. Opis projektu ... 3
Wprowadzenie ... 3
Zasady działania ... 3
Wymagania projektu ... 4
ROZDZIAŁ 2. Struktura projektu ... 4
Ogólny podział ... 4
Schemat działania ... 4
Użyte biblioteki ... 5
Ważne funkcje ... 5
ROZDZIAŁ 3. Działanie programu ... 5
Ogólny algorytm ... 5
Spotkane problemy ... 6
ROZDZIAŁ 4. Instrukcja użytkownika ... 7
Stworzenie pliku wejściowego ... 7
Wywołanie programu ... 8
*ROZDZIAŁ 5. Testy* ... 11
Jednostkowe testy ... 11
*ROZDZIAŁ 6. Ewentualny rozwój* ... 12
ROZDZIAŁ 7. Wykorzystane zasoby ... 12

# ROZDZIAŁ 1. Opis projektu

## Wprowadzenie

Projekt polega na stworzeniu programu w języku Python działającego analogicznie do maszyny Turinga, którą wymyślił Alan Turing w 1936 roku. Maszyna Turinga to abstrakcyjny model obliczeń, służący do wykonania algorytmów. Maszyna posiada w teorii nieskończoną taśmę składającą z komórek w których zapisane są dane (jeden symbol z używanego alfabetu w jednej komórce), głowicę, która sczytuję dane z taśmy oraz blok sterujący, który steruje ruchy maszyny i zmienia swój stan w zależności od sczytanych danych.




Rys. 1 Model maszyny Turinga

## Zasady działania

1. Głowica może poruszać się tylko o jedną komórkę w dowolną stronę.
2. Wszystkie używane na taśmie symbole, muszą zawierać się w alfabecie (łącznie ze znakiem pustej komórki).
3. Każdy stan musi zawierać instrukcje dla każdego symbolu alfabetu.
4. Każda instrukcja musi zawierać trzy polecenia:
    a. Co wpisać w aktualną komórkę (instrukcja komórki).
    b. W którą stronę przesunąć głowicę (instrukcja przysunięcia).
    c. W który stan przejść (instrukcja następnego stanu).
5. Przynajmniej jedna instrukcja musi zatrzymywać maszynę (stan akceptujący)

## Wymagania projektu

1. Na wejściu program maszyny Turinga przyjmuje plik tekstowy z podaną taśmą początkową, alfabetem oraz listą instrukcji poszczególnych stanów.



2. W trakcie działania program wypisuje zawartość komórek w okolicach głowicy oraz aktualny stan.

3. Na wyjściu maszyna zapisuje końcową zawartość taśmy w pliku tekstowym.

# ROZDZIAŁ 2. Struktura projektu

## Ogólny podział

1. Podstawowy program – turing_machine.py
    a. Klasa TuringMachine
    b. Klasa Tape
    c. Klasa State
2. Program realizujący GUI – gui.py

## Schemat działania

Pierwszy sposób – przez konsole.

1. Stwórz plik wejściowy
2. Uruchom program padając ścieżkę do pliku wejściowego jako pierwszy argument oraz nazwę pliku wyjściowego jako drugi argument.
3. Program się wykonuje.
4. Program zapisuje wynik w podanym pliku.

Drugi sposób – przez GUI.

1. Uruchom program, nie podając argumentów.
2. Korzystając z interfejsu wprowadź dane wejściowe przez plik lub manualnie.
3. Przejrzyj proces działania maszyny w GUI.

## Użyte biblioteki

W trakcie tworzenia projektu korzystano z bibliotek:

- Os
- Re
- Sys
- Tkinter



Ale w aktualnym stanie projektu program korzysta tylko z biblioteki sys w celu pobrania argumentów z wiersza poleceń oraz z biblioteki tkinter w celu stworzenia graficznego interfejsu użytkownika.

## Ważne funkcje

1. TuringMachine.move(way) – przesuwa głowicę.
2. TuringMachine.current_instruction() – pobiera aktualną instrukcję.
3. TuringMachine.step() – wykonuję pobraną instrukcję.
4. TuringMachine.show(radius) – zwraca znaczenia komórek w podanych okolicach głowicy.
5. Tape.read() – sczytuję znaczenie komórki, na której znajduje się głowica.

# ROZDZIAŁ 3. Działanie programu

## Ogólny algorytm

1. Program sczytuje dane wejściowe
2. Tworzy obiekt klasy TuringMachine
3. Tworzy taśmę – obiekt klasy Tape jako parametr klasy TuringMachine
4. Tworzy lista obiektów klasy State jako parametr TuringMachine
5. Wystawia głowicę na początek taśmy oraz pierwszy stan, jako aktualny
6. Pobiera instrukcje
7. Wykonuje instrukcje
8. Sczytuje nowy symbol z taśmy
9. Powtarza punkty 6,7,8 do momentu spotkania „stop” jako instrukcje nowego stanu.
10. Wypisuje końcową taśmę w podanym pliku.

# Spotkane problemy



1. Cały projekt na początku wydawał się trudnym, ale po rozbiciu całego mechanizmu na oddzielne drobne fragmenty, prawie każdy z tych fragmentów okazał się prymitywny. Z tego powodu podczas stworzeniu projektu nie było spotkano dużej ilości problem.

2. Na początku stworzenia projektu nie wiedziałem, że taśma maszyny Turinga musi być nieskończona tylko w jedną stronę, więc zrobiłem ją nieskończoną w obie strony, co nie przeszkadza algorytmom napisanym dla maszyny Turinga z taśmą nieskończoną w prawą stronę, bo takie algorytmy zawsze oczekują znak pustej komórki w najbardziej lewej komórce, który będzie zawsze tam stał z powodów struktury taśmy opisanych dalej. Taśma jest obiektem klasy Tape. Jako posiada parametr p_cell i n_cell oznaczające komórki z pozytywnym i negatywnym indexem odpowiednio w postaci listy symboli. Kiedy głowica przybliża się do dowolnego końca taśmy tworzy się następna komórka w postaci symbolu '_' oznaczającego pustą komórkę. Komórki w taki sposób stworzonej taśmie skończą się tylko razem z pamięcią wydzieloną na listę w Python'ie.

3. Stworzenie systemy wyboru odpowiedniej instrukcji w zależności od sczytanego symbolu, za pomocą wyrażeń „if" i porównaniu każdego elementu listy podanej przez użytkownika długości, wyglądało by strasznie nie czytelnie oraz miałoby nienajlepszą złożoność obliczeniową. Zatem stworzony został analog wyrażenia switchcase za pomocą kolektora dictionary w którym jako klucze zapisane wszystkie elementy alfabetu, a jako znaczenia odpowiednie elementom alfabetu instrukcje.

# ROZDZIAŁ 4. Instrukcja użytkownika

## Stworzenie pliku wejściowego

1. Pierwszym wiersz - zawartość początkowa taśmy, bez żadnych separacji.
2. Drugi wiersz - alfabet, symbole rozdzielone pojedynczą spacją
3. Pozostałe wiersze -Każdy wierszy odpowiada jednemu stanu. Każdy stan trzeba opisać w następny sposób: „numer stanu, instrukcja dla pierwszego elementu alfabetu, instrukcja dla drugiego elementu alfabetu, …., instrukcja ostatniego elementu alfabetu". Najpierw idzie numer stanu, a po nim instrukcje dla wszystkich elementów alfabetu w odpowiedniej kolejności. Numer stanu i wszystkie instrukcje oddzielone są przecinkiem ze spacją. Każda instrukcja to trzy elementy – element do wpisania, kierunek przesunięcia „R" – w prawo, „L" – w lewo, numer następnego stanu. Każdy element instrukcji oddzielony jest spacją.
4. Podany plik wejściowy musi spełniać zasady działania maszyny.

Przykładowy plik wejściowy:

*00100*

*0 1 _*

*2, 1 R 2, 0 R 2, _ R stop*

*1, 1 R 1, 0 R 2, _ R stop*

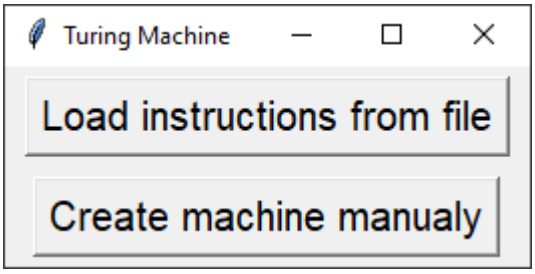
## Wywołanie programu

Z GUI:

W celu wywołania interfejsu programu, w wiersz poleceń trzeba wpisać:

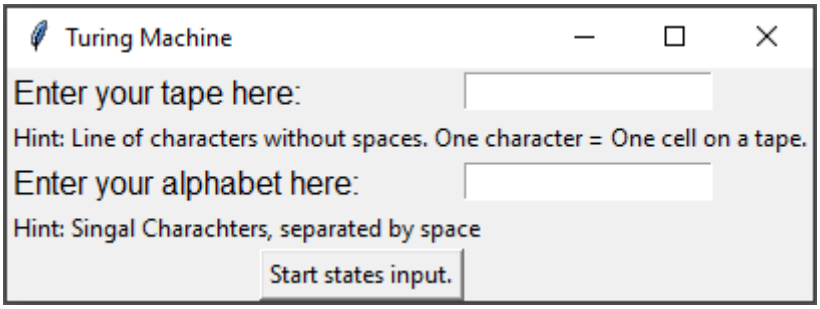
*python „ścieżka do pliku programy”*

Wtedy odtworzy się interfejs, w którym można wybrać plik wejściowy (Load instructions from file) po czym otworzy się interfejs, działającej maszyny lub wpisać dane ręcznie (Create machine manualy).



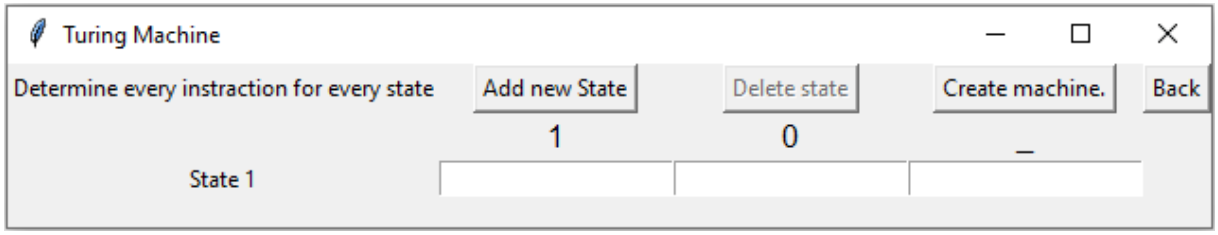

Rys. 4.1 GUI strona wyboru sposobu stworzenia maszyny

W przypadku wpisania danych ręcznie interfejs się zmieni, pojawi się możliwość wpisania taśmy i alfabetu, jeśli wprowadzone dane nie są poprawne przycisk „Start states input” nie będzie działał.




Rys. 4.2 GUI strona wprowadzenia taśmy i alfabetu

Po wprowadzeniu poprawnych danych interfejs znowu się zmieni i pojawi się możliwość wpisania, dodania i usuwania stanów. Po użyciu przycisku „Create machine” otworzy się interfejs działającej maszyny.

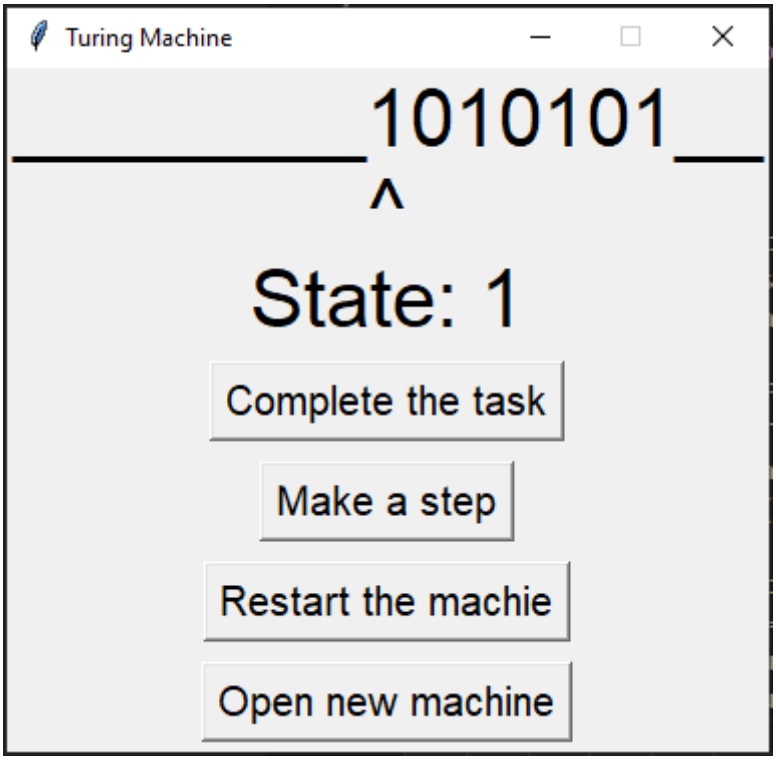



Rys. 4.3 GUI przykładowa strona wprowadzenia stanów

Z interfejs działającej można:

1. Wypełnić działanie maszyny do końca.
2. Przejść jeden krok do przodu.
3. Zresetować maszynę do początkowego stanu.
4. Stworzyć nową maszynę.





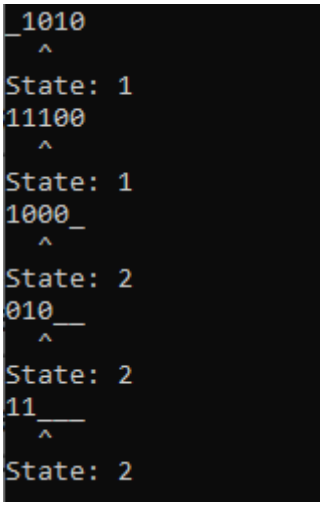

Rys. 4.4 GUI Przykładowa strona działającej maszyny

W celu wywołania programu bez interfejs, w wiersz poleceń trzeba wpisać

*python „ścieżka do pliku programy” „ścieżka do pliku wejściowego” „ścieżka do pliku wyjściowego”*



*Wtedy w wierszy poleceń zostaną wypisane znaczenia komórek w okoliczności głowicy oraz aktualny stan po każdym kroku i taśma końcowa zostanie zapisana w podany plik wyjściowy.*



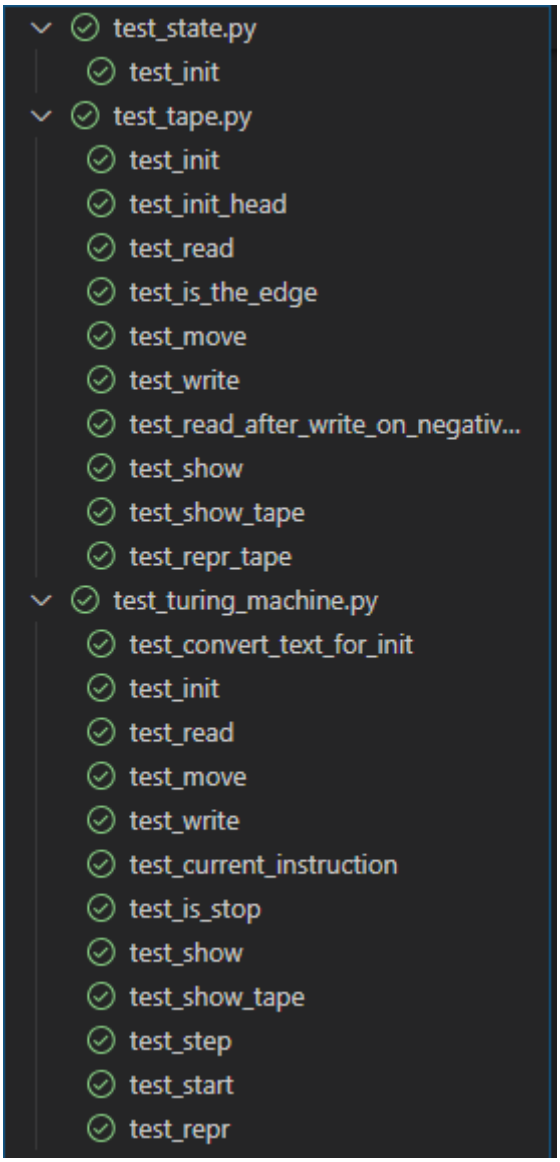

*4.5 Przykładowe komunikaty wiersza poleceń*

# ROZDZIAŁ 5. Testy

## Jednostkowe testy

*Przed napisaniem większości funkcji i metod każdej klasy w pliku turing_machine.py był pisany test, który później sprawdzał czy konkretny moduł działa poprawnie i zgodnie z oczekiwaniami. W przypadku, gdy funkcja działała inaczej, niż oczekiwano, ale akceptująco lub lepiej, zmieniony zostawał sam test.*






*Rys. 5.1 lista stworzonych testów*

# *ROZDZIAŁ 6. Ewentualny rozwój*

Projekt może być dopełniony stworzeniem preset'ów, którzy by zawierały w sobie najczęściej używane alfabet i listy stanów, ale pozwalali na zmianę zawartości taśmy

Graficzny Interface Użytkownika ma przestrzeń na rozwój. Możliwość wykonania operacji „Undo" przy działającej maszynie. Możliwość sterowania preset'ami. Animacja przysuwania się głowicy.



# ROZDZIAŁ 7. Wykorzystane zasoby

Obraz z początku dokumentacji

Dokumentacja tkinter

Maszyna Turinga Wikipedia