Python

Warsztaty od podstaw

9.10.2018 16:30 Szczecin

infoShare ACADEMY

Informacje

// sieć Wi-Fi: szkolenie

// hasło: tach5neck

// slajdy, kod: **http://bit.ly/UPythona**

Trener

Paweł Żal

Agenda

- 16:30 - 18:30 - podstawy języka Python
- 18:30 - 18:40 - przerwa
- 18:40 - 20:00 - aplikacja „Uwolnić Pytona”

1.

Podstawy języka Python

Python

// interpretowany - linijka po linijce

// obiektowy

// popularny – używają go: 

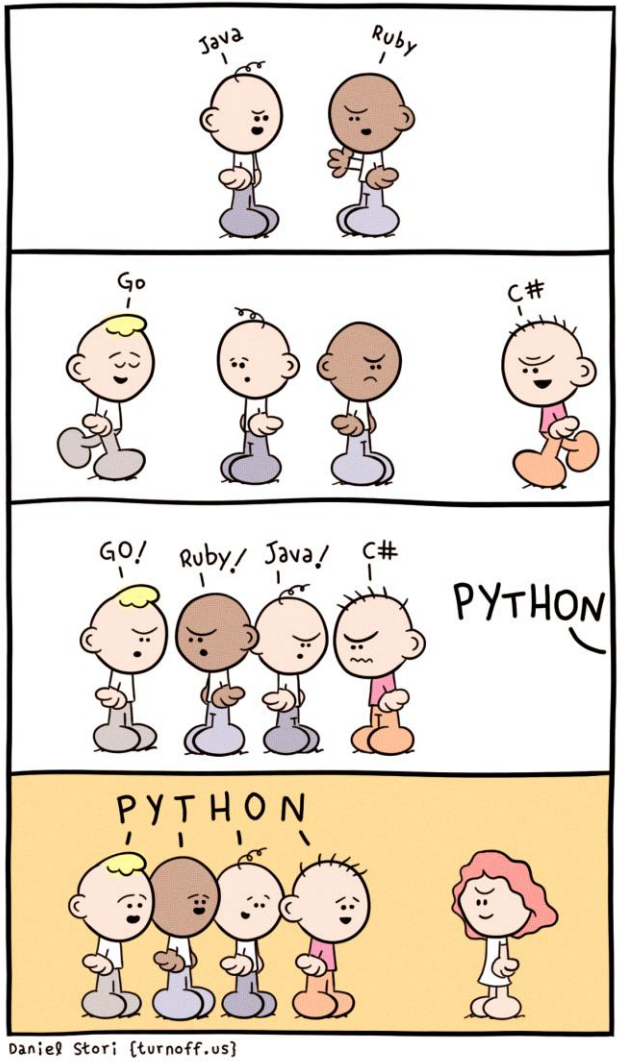
// powszechny - (Big Data, Machine Learning, Web,

Blockchain, AI, devops, hacking, computer vision)

Sympatia do Pythona

Obrazek z wczoraj, tj. 08.10.2018

http://turnoff.us/geek/the-depressed-developer-35/



Tworzenie kodu

- interpreter

- zwykły notatnik
- pliki tekstowe z rozszerzeniem **.py**
- IDE - dodatkowa funkcjonalność (podpowiedzi, kolorowanie składni, debugger, testy)
- Python IDLE, **PyCharm**, VS Code, Sublime, Atom

Uruchamianie kodu

// interpreter (python.exe)

// konsola / terminal / wiersz polecenia

// można mieć kilka wersji Pythona

// IDE umożliwiają uruchamianie bezpośrednio

// nie zawsze program zadziała bez IDE

Praktyka: source code/01.py

Typy danych, ale nie zmiennych(!)

123	- int	– liczby całkowite
54.45	- float	– liczby zmienno-przecinkowe
"Basia"	- str	– łańcuchy znaków (string)
True/False	- bool	– True/False
None	- None	– nic, pustka, nieokreśloność

listy, słowniki, tuple (krotki), pliki, własne typy (klasy)

Zmienna

// nazwany obszar pamięci, w którym znajduje się jakaś wartość

// pozwala na ponowne użycie wartości w innym miejscu w kodzie

```
moja_liczba = 124
nazwisko = "Kowalski"
czy_obecny = True
```

Operatory

Matematyczne:

```
+, -, *, /, //, %, **
```

Logiczne:

```
==, !=, <, >, <=, >=
```

```
in,  is,  and,  or,  not
```

Operatory

obliczane jest wyrażenie **po prawej** stronie znaku, następnie wartość jest przypisywana do zmiennej po lewej stronie znaku

```
wynik = (5 != 4)
```

Praktyka: 02.py

Atrybuty wbudowane typów

Każdy typ danych posiada zdefiniowane atrybuty (metody i pola), które pozwalają na wykonanie różnych (najpopularniejszych) działań, właściwych dla tego typu.

typ.funkcja()

```
”ala ma kota”.capitalize()
```

Praktyka: 03.py

String (łańcuch znaków)

```
nazwisko = "Kowalski"
# długość
len(nazwisko) -> 8
# Indeksowanie
nazwisko[0] -> K
nazwisko[3] -> a
nazwisko[8] -> błąd, nie ma takiego indeksu!
```

Praktyka: 04.py

int - float - str

5 - int - liczba całkowita

65.987 - float - liczba zmiennoprzecinkowa

'45' - str - łańcuch znaków

"3434.434" - str - łańcuch znaków

int - float - str

```
nazwisko = input(”Podaj nazwisko: ”)
```

input() przyjmuje od użytkownika dane i zapisuje do zmiennej. Wszystko jest stringiem

```
print(nazwisko)
```

print() służy do wydrukowania tekstu na ekranie; automatycznie dodaje na końcu stringa znak specjalny nowej linii **\n**

blok kodu

```
Instrukcja/wyrażenie:
    Instrukcja
    Instrukcja
    Instrukcja:
        instrukcja

Instrukcja/wyrażenie:
    Instrukcja
    Instrukcja

    Instrukcja
```

Dwukropek rozpoczynający blok

Indentacja 1 poziom (4 spacje)

instrukcja warunkowa

```
if (warunek):
    # kod wykonany gdy warunek prawdziwy
elif (inny warunek):
    # kod wykonany gdy warunek w if był fałszywy
    # warunek w tym elif musi być prawdziwy aby ten kod wykonać
elif (inny warunek):
    # elif-ów może być wielu lub żadnego, kod wewnątrz elif
    # wykona się tylko gdy wszystkie wyższe warunki były fałszywe
else:
    # przypadek domyślny, tu nie sprawdzamy warunku, kod w else
    # będzie wykonany gdy wszystkie w if- elif były fałszywe
    # else może być tylko jeden lub wcale
```

Praktyka: 05-07.py

Tablica logiczna

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

import

import moduł

from moduł **import** funkcja

from moduł **import** *

string, datetime, copy, math, decimal,

random, os, csv, antigravity

range()

range(stop)

range(3) - <0, 1, 2> // len() == 3

range(start, stop)

range(4, 8) - <4, 5, 6, 7>

range(start, stop, krok)

range(0, 10, 3) – <0, 3, 6, 9>

list()
[]

```
lista = [1, 2, 3]

lista2 = ["kwiatek", "doniczka", "ziemia", "woda"]

lista3 = []

lista4 = [1, "dwa", 3, 4]

lista5 = list(range(2,5))
```

Możemy indeksować (używać indeksów), slice'ować (wycinać)

Do elementu odwołujemy się przez indeks **Praktyka 08.py**

tuple() – inaczej: krotka ()

Tuple jest typem niezmiennym – raz zdefiniowanej tupli nie można zmienić

```
tuple1 = ("raz", "dwa", "trzy")
```

`tuple1[0] = "jeden"` – spowoduje błąd

```
x = "raz",
y = "raz", "dwa"
```

– ten przecinek jest bardzo istotny

dict()
{}

{ klucz: wartość }

klucz – musi być typu niezmiennego (string, tuple, liczba), musi być unikalny (tylko jeden w słowniku)

wartość – mogą być powtórzone

Odwołujemy się poprzez klucz a nie indeks!!!

Praktyka: 09.py

pętla while

```
while (wartość logiczna True):
        kod
        ...
        update wartości logicznej na False
```

Kod wewnątrz pętli **while**, będzie powtarzany dopóki wartość logiczna (wyrażenia lub zmiennej) nie zmieni się na **False***.

* chyba, że pętla zostanie przerwana lub zmodyfikowana

Praktyka: 10.py

pętla for

```
for element in kolekcja:
        tu możemy użyć element
        …
```

Pętla „**for**” wykona się tyle razy ile elementów jest w kolekcji*.

* chyba, że pętla zostanie przerwana lub zmodyfikowana

Praktyka: 11.py

pliki

```
plik = open("ścieżka_do_pliku", 'tryb')
```

tryby:

- **r** – tylko do odczytu
- **w** – zapisywanie pliku (zawartość starego pliku o tej samej nazwie zostanie usunięta)
- **r+** - do odczytu i zapisu
- **a** – dopisywanie do pliku (dane są dopisane na koniec istniejącego pliku)

pliki tekstowe odczyt

`plik.read()` – odczytanie całego pliku, zwracany jest string zawierający cały tekst pliku (włącznie ze znakami ”\n”); opcjonalny argument określa ilość bajtów do wczytania

`plik.readline()` – odczytanie jednej linii z pliku, zwracany jest string z linijką testu, włącznie ze znakiem ”\n”

`plik.readlines()` – odczytuje cały tekst – zwraca listę stringów - linijek

Praktyka: 11 - 12.py

pliki tekstowe zapis

`plik.write(string)` – zapisuje string do pliku w obecnej pozycji kursora, zwraca liczbę zapisanych znaków – należy pamiętać o znaku ”\n”

`plik.writelines(iterable)` – zapisuje elementy z kolekcji jako poszczególne linie w pliku

Plik musi być otworzony w trybie do zapisu aby móc go zmieniać!

funkcje - definiowanie

definiowanie:

```
def do_nothing():
    pass
```

wywołanie:

do_nothing()

funkcje - argumenty

```
def do_nothing():
  pass
```

nie ma argumentów

```
def do_nothing(x):
  pass
```

jeden argument

```
def do_nothing(x, y, z):
  pass
```

wiele argumentów

funkcje - zwracanie wartości

```
def print_square(x)
    print(x**2)

def give_square(x)
    return x**2
```

Aby użyć funkcję zwracającą obiekt należy ten obiekt zapisać w zmiennej.

```
>>> wynik = give_square(3)
>>> print(wynik)
9
```

funkcje - argumenty domyślne

```
def do_nothing(x, y=10):
    pass

def do_nothing(x, y, z=12, w = „Ola”):
    pass

def do_nothing(y=10):
    pass
```

Argumenty domyślne muszą być po argumentach wymaganych.

funkcje - argumenty domyślne

```
def do_something(x, y, z=12, w =„Ola”):
    pass

>>> do_something(1)
```

<- błąd! – wszystkie argumenty pozycyjne muszą zostać podane

```
>>> do_something(1, 23)
>>> do_something(1, 2, "trzy")
>>> do_something(1, 2, 34, "Ola")
>>> do_something(1, 33, w="Ola")
```

Praktyka: 14-15.py

2.

Aplikacja „Uwolnić Pytona”

O co chodziło w lipcu 2018?

Pyton tygrysi z okolic Warszawy może już być pod Toruniem. Szokujące ustalenia

WP Wiadomości

3 godziny temu

Pyton się ukrył i czeka na cieplejsze dni. Strażacy przerwali poszukiwania

Gazeta Wyborcza Warsz...

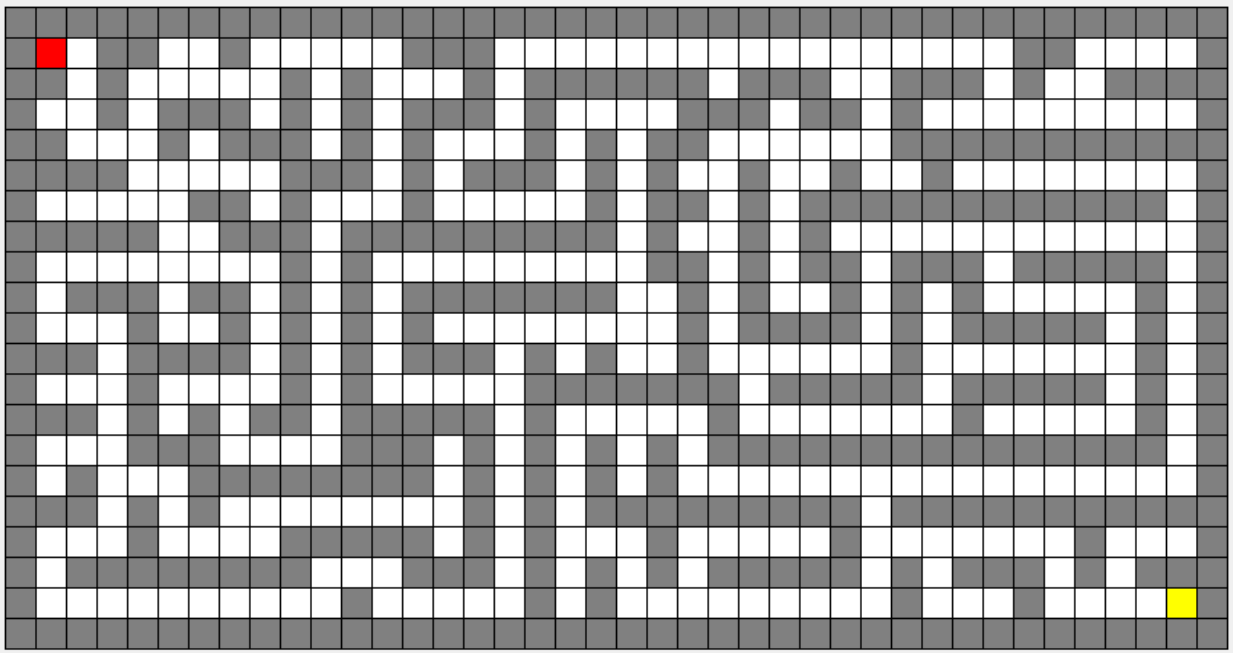
dzień temu

Gdzie jest pyton znad Wisły? Trwa analiza zdjęć

Planeta FM

22 godziny temu

Jak uwolnić Pytona?



Algorytm „zalewania wodą”

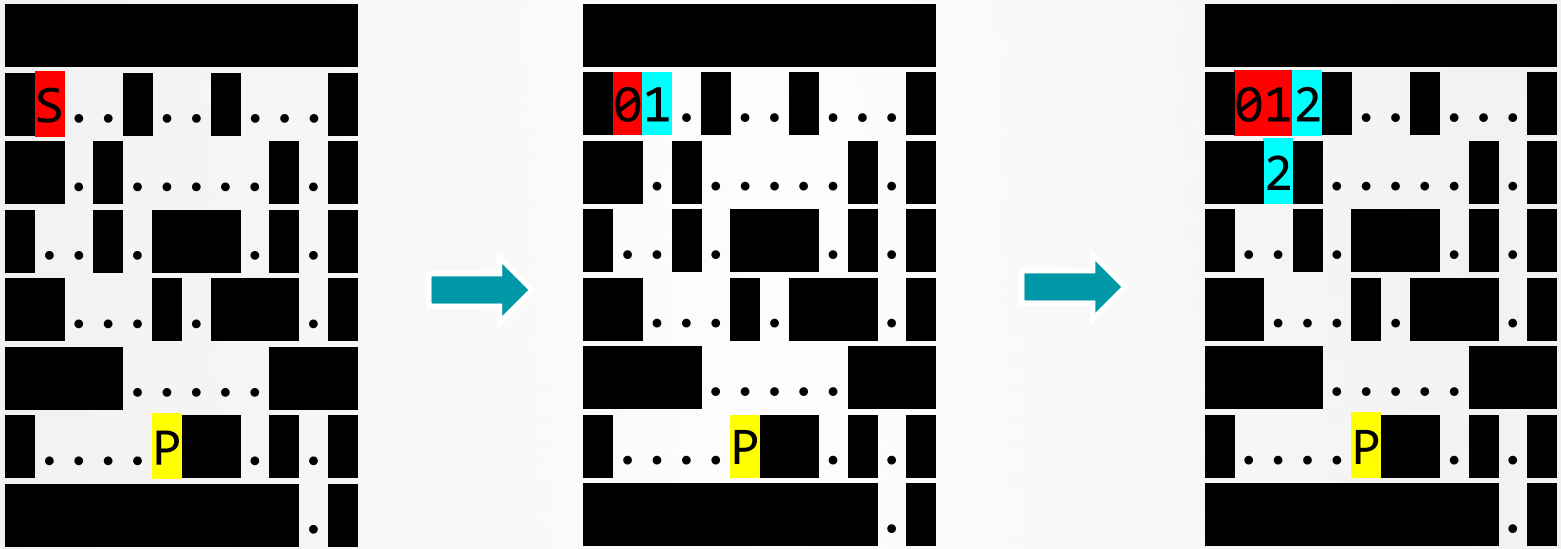
- prosty algorytm zwany też „symulacją pożaru”

- algorytm to 2 pętle:

  pętla pierwsza

  - bada 4 sąsiadów wskazanego pola labiryntu
  - jeśli pola są puste, to:
    - zalewa je wodą (i numeruje)
    - odkłada na listę kandydatów do badania
  - jeśli znajdzie cel to przerywa działanie pętli

Algorytm „zalewania wodą”

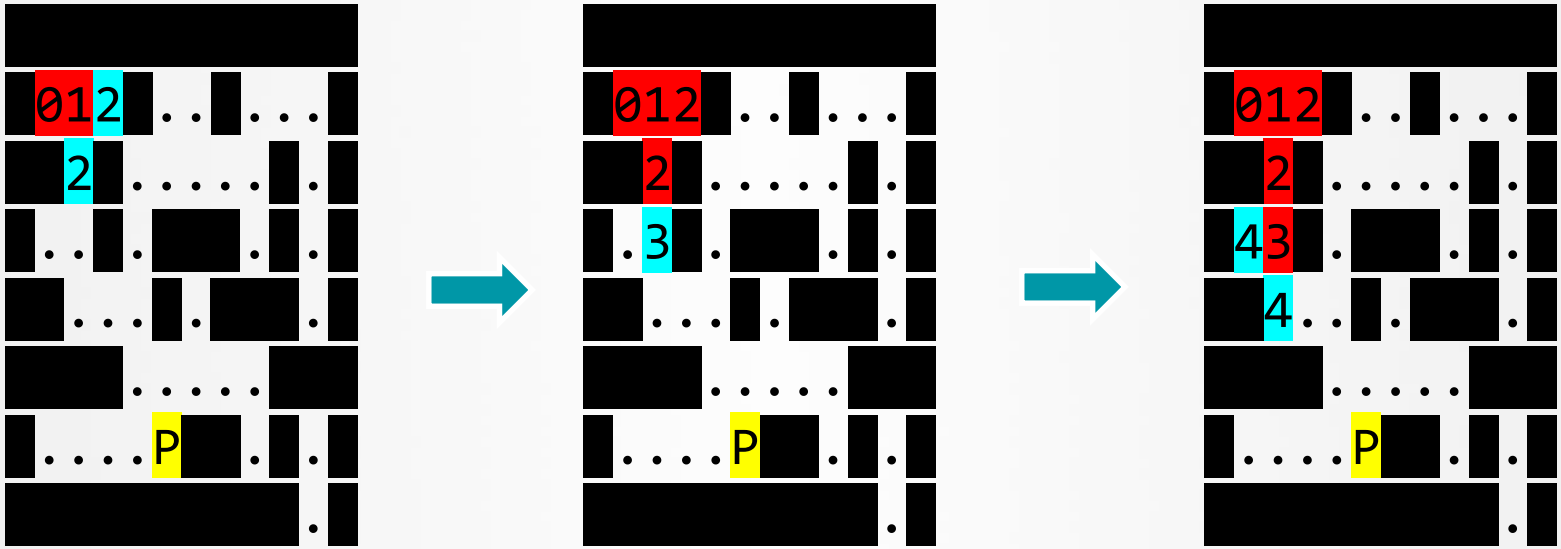


Krok 0

Krok 1

Krok 2

Algorytm „zalewania wodą”

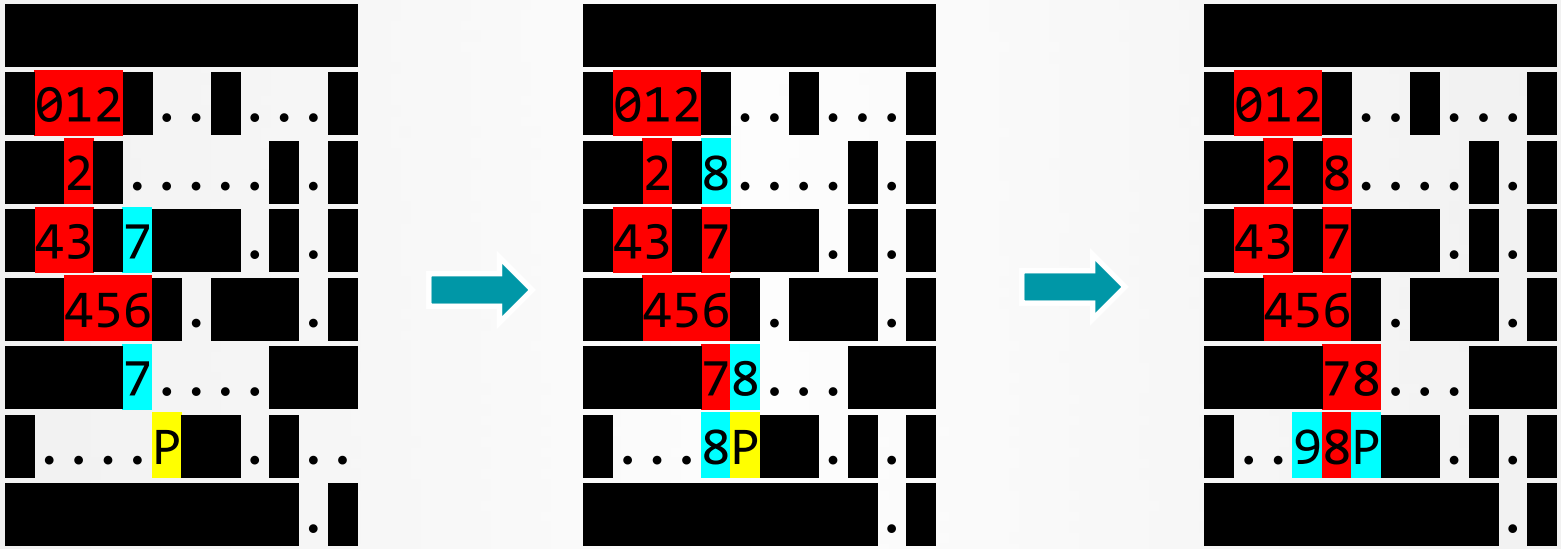


Krok 2

Krok 3

Krok 4

Algorytm „zalewania wodą”



Krok 7

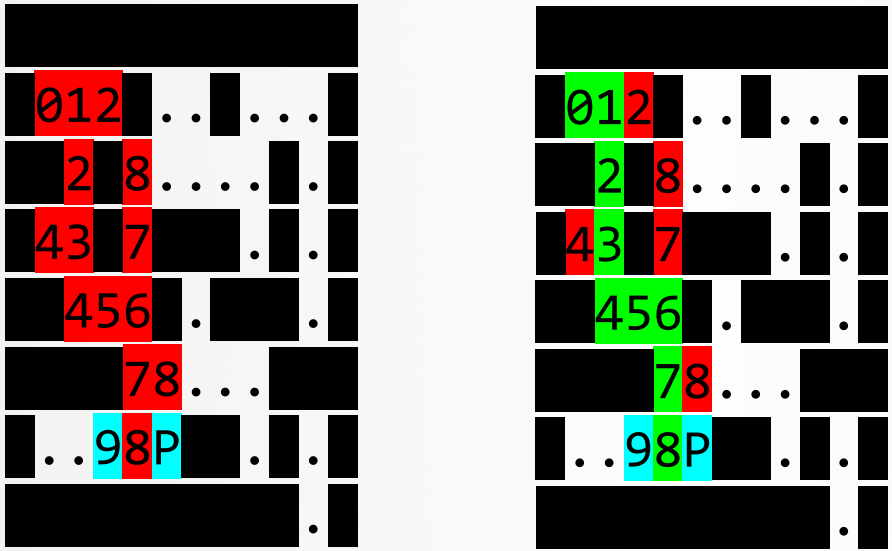
Krok 8

Krok 9

Algorytm „zalewania wodą”

pętla druga

- odtworzenie ścieżki wg numerów pól (malejąco)



Krok 9

Implementacja

Implementacja modułu logiki: **labirynt.py**
Implementacja interfejsu tekstowego: **labirynt.py**

Implementacja interfejsu graficznego: **uwolnic_pythona.py**

Moduł Tkinter:

- https://wiki.python.org/moin/TkInter
- https://docs.python.org/3.6/library/tk.html

Dziękuję