

PODSTAWY PROGRAMOWANIA W JĘZYKU PYTHON

Dzień 3



Agenda

- Instrukcja warunkowa
- Schemat blokowy
- Pętla while
- Lista
- Range
- Pętla for

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 warunek1 and inny warunek2):

elif-ów może być wiele lub nie być żadnego,

kod wewnątrz elif wykona się tylko gdy wszystkie

poprzedzające go warunki były niespełnione (fałszywe)

else:

przypadek domyślny, nie ma warunku,

kod w else będzie wykonany gdy wszystkie poprzednie warunki

były fałszywe, else może być tylko jeden lub wcale

pythontutor.com

[Start shared session](#)[What are shared sessions?](#)

Python 3.6

```
1 a = 'Hello world'
2 b = 34
3 b *= 3
4
5 print(b)
```

[Edit code](#) | [Live programming](#)

→ line that has just executed

→ next line to execute

NEW! Click on a line of code to set a breakpoint. Then use the Forward and Back buttons to jump there.

<< First

< Back

Step 3 of 4

Forward >

Last >>

Print output (drag lower right corner to resize)

Frames

Objects

Global frame

a	"Hello world"
b	34

Generate permanent link

Generate shortened link

Click the button above to create a permanent link to your visualization. To report a bug, paste the link along with a brief error description in an email addressed to philip@pgbovine.net

Generate embed code

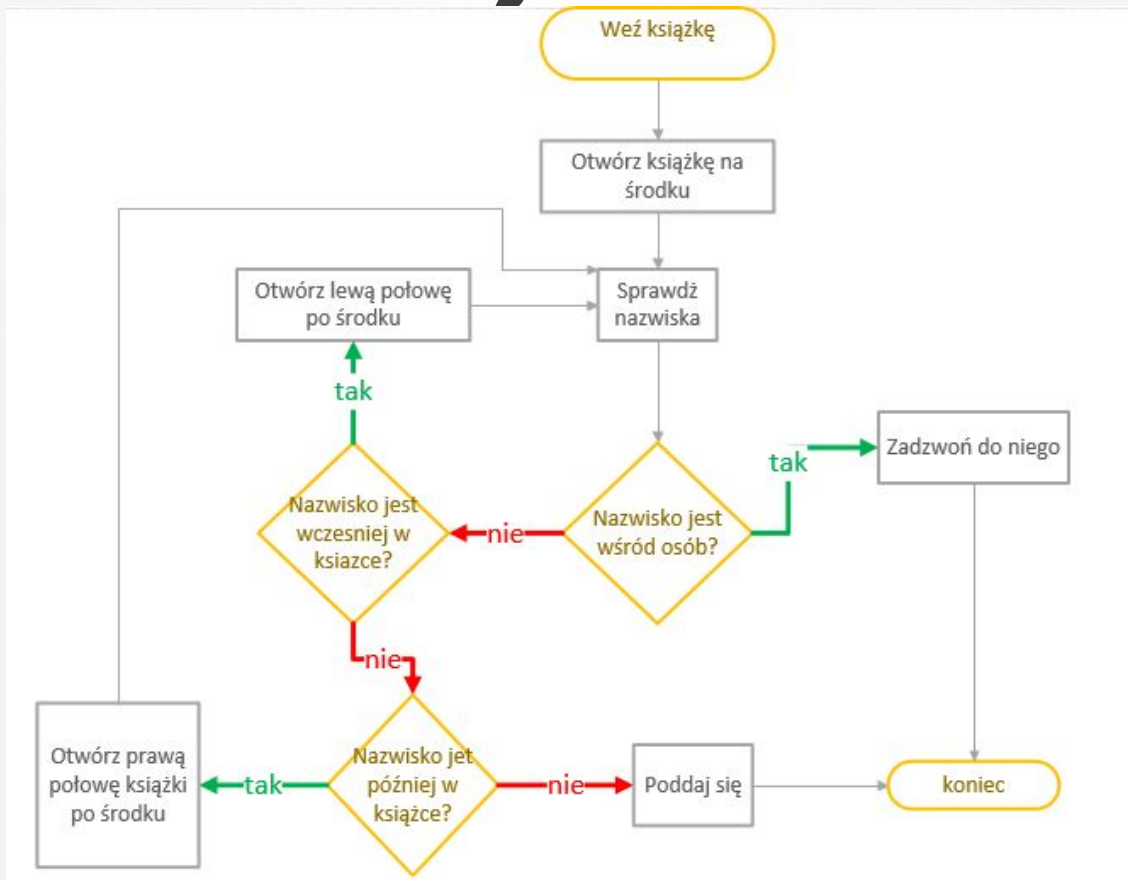
To embed this visualization in your webpage, click the 'Generate embed code' button above and paste the resulting HTML code into your webpage. Adjust the height and width parameters and change the link to **https://** if needed.

Schemat blokowy

Pseudokod

1. Weź książkę telefoniczną.
2. Otwórz książkę na środku.
3. Sprawdź nazwiska na otwartych stronach.
4. Jeśli "Wojtkowiak" jest wśród osób:
 - a. Zadzwoń do niego
5. W przeciwnym razie jeśli "Wojtkowiak" jest wcześniej w książce:
 - a. Otwórz lewą połowę po środku.
 - b. Idź do kroku 3.
6. W przeciwnym razie jeśli "Wojtkowiak" jest później w książce:
 - a. Otwórz prawą połowę po środku.
 - b. idź do kroku 3.
7. W przeciwnym razie:
8. poddaj się.

Schemat blokowy



Pętla while

Pętle

1. Weź książkę telefoniczną.
2. Otwórz książkę na środku.
3. Sprawdź nazwiska na otwartych stronach.
4. Jeśli "Wojtkowiak" jest wśród osób:
 - a. Zadzwoń do niego
5. W przeciwnym razie jeśli "Wojtkowiak" jest wcześniej w książce:
 - a. Otwórz lewą połowę po środku.
 - b. Idź do kroku 3.**
6. W przeciwnym razie jeśli "Wojtkowiak" jest później w książce:
 - a. Otwórz prawą połowę po środku.
 - b. idź do kroku 3.**
7. W przeciwnym razie:
- 8. poddaj się.**

Pętla while

while (warunek):

blok kodu

- Kod w pętli **while** wykonywany będzie tak długo jak długo warunek będzie spełniony (rzutowalny na True).
- Oznacza to że warunek może być niespełniony już na początku (pętla nie wykona się nigdy).
- Kod wewnątrz pętli while, będzie powtarzany dopóki wartość logiczna (wyrażenia lub zmiennej) nie zmieni się na False*.

** chyba, że pętla zostanie przerwana lub zmodyfikowana*

Lista

Lista

`list()`, `[]`

- Zbiór kolejnych par indeks - wartość, mogących różnić się typem wartości.
- W Pythonie indeksy są liczbami całkowitymi, pierwszym indeksem jest zawsze 0 (zero).
- Do elementu odwołujemy się przez indeks.
- Możliwe jest wycinanie fragmentu listy (*slice*), który staje się **nową** listą.

Przykłady list

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

```
>>> lista2 = ["owca", "lama", "stado"]
```

```
>>> lista3 = [] # pusta lista
```

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

```
>>> lista5 = list(range(3)) # [0, 1, 2]
```

Operacje na listach

- Sprawdzanie długości:

```
>>> len([0, 1, 2]) # 3
```

```
>>> lista2 = ["owca", "lama", "stado"]
```

- Wyciągnięcie elementu o indeksie 1:

```
>>> lista2[1] # "lama"
```

- Przypisanie do indeksu 0 nowej wartości

```
>>> lista2[0] = "wataha"
```

- Dodanie nowego indeksu do listy:

```
>>> lista2.append("grupa"); print(lista2)
```

```
["owca", "lama", "stado", "grupa"]
```

- **String** wspiera niektóre operacje list (indeksowanie, długość)!

Range

Range

Funkcja zwracająca kolejne wartości z zadanego przedziału.

```
range(stop)
```

```
>>> range(3) # <0, 1, 2>
```

```
range(start, stop)
```

```
>>> range(4, 8) # <4, 5, 6, 7>
```

```
range(start, stop, krok)
```

```
>>> range(0, 10, 3) # <0, 3, 6, 9>
```


Petla for

Pętla for

for element in kolekcja:

blok kodu

- Pętla **for** wykona się dla każdego elementu w kolekcji*.
- Przestrzeń nazw bloku kodu jest rozszerzana o zmienną **element** (nazwa zmiennej może być dowolna).
- Kolekcja może być pusta.
- Pętle można zagnieżdżać (dotyczy również pętli **while**).

** chyba, że pętla zostanie przerwana lub zmodyfikowana*

Zmienianie przebiegu pętli

continue – program pomija pozostałe instrukcje w bloku i wraca do sprawdzenia warunku (**while**) lub do kolejnego elementu (**for**).

break – działanie pętli jest przerywane, program przechodzi do kolejnej instrukcji po całym bloku pętli.

enumerate()

```
for (indeks, element) in enumerate(iterable):  
    # kod pętli for
```

- **enumerate** daje nam dwie wartości:
 - indeks bieżącego elementu,
 - oraz ten element.

zip()

```
for (element_a, element_b) in zip(iterable_a, iterable_b):  
    # kod pętli
```

- **zip** daje nam elementy z tej samej pozycji w kilku kolekcjach.
- Gdy kolekcje są różnej długości, długość najkrótszej kolekcji będzie określała liczbę powtórzeń pętli.
- Inne funkcje wbudowane:
<https://docs.python.org/3/library/functions.html>

Podsumowanie

- Schemat blokowy
- Pętla while
- Lista
- Range
- Pętla for



Thanks!!