



## Zagadka Wież Hanoi w języku Python

- [Wprowadzenie](#)
- [Przeczytaj](#)
- [Symulacja interaktywna](#)
- [Sprawdź się](#)
- [Dla nauczyciela](#)



## Zagadka Wież Hanoi w języku Python

Źródło: Thuan Pham, domena publiczna.

Łamigłówka Wież Hanoi polega na tym, by ułożone w wieżę krążki o różnej średnicy przenieść pojedynczo z pierwszego słupka na trzeci, tak aby odtworzyć na nim wieżę – przy czym należy robić to według określonych zasad. Zagadka ta oparta jest na starej legendzie, według której buddyjscy mnisi ze świątyni Brahmy mają w ten sposób przenieść wieżę składającą się aż z 64 krążków, czyli w celu rozwiązania problemu muszą wykonać aż  $2^{64} - 1$  ruchów. W e-materiale [Zagadka Wież Hanoi](#) przedstawiliśmy najważniejsze informacje dot. tego zagadnienia.

W tym e-materiale poznasz implementację algorytmu rozwiązywania tego problemu w języku Python.

Więcej przykładów, ćwiczeń i rozwiązań znajdziesz w e-materiale [Zagadka Wież Hanoi – zadania maturalne](#).

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych lekcjach z tej serii:

- [Zagadka Wież Hanoi w języku C++](#),
- [Zagadka Wież Hanoi w języku Java](#).

O tym, jak zagadnienie rekurencji wyjaśnia matematyka, przeczytasz w e-materiałach:

- Ciąg określony rekurencyjnie,
- Ciąg geometryczny określony rekurencyjnie,
- Wzór ogólny ciągu określonego rekurencyjnie,
- Ciąg arytmetyczny określony wzorem rekurencyjnym.

### **Twoje cele**

- Przypomnisz sobie algorytm rozwiązania zagadki Wież Hanoi.
- Przeanalizujesz implementację tego algorytmu w języku Python.
- Rozwiążesz samodzielnie zadania związane z zagadką Wież Hanoi.



# Przeczytaj

---

## Implementacja rozwiązania w języku Python

### Ciekawostka

Liczba ruchów niezbędnych do przeniesienia wszystkich krążków ze słupka A na słupek C wynosi  $(2^n - 1)$ , czyli rośnie wykładniczo wraz ze zwiększaniem liczby krążków. Jeśli przyjmiemy, że jeden krążek przemieszczamy w ciągu jednej sekundy, to rozwiązanie łamigłówki dla trzech krążków zajmuje tylko siedem sekund, podczas gdy w przypadku 20 krążków czas ten wydłuża się już do 1048576 sekund (ponad 17000 minut, czyli około 11 dni).

Jeden z najszybszych współczesnych komputerów, MareNostrum 4, wykonuje  $13,667 \cdot 10^{12}$  operacji na sekundę. By rozwiązać zagadkę dla 64 krążków potrzebowałby nieco ponad dwóch tygodni.

### Przykład 1

Zdefiniujemy funkcję, która wykorzystując podany wcześniej algorytm, zapisze listę ruchów koniecznych do rozwiązania łamigłówki Wież Hanoi.

```
1 def hanoi(n, a, b, c):
2     if n > 0:
3         hanoi(n - 1, a, c, b)
4         print(f"przenieś {a} -> {b}")
5         hanoi(n - 1, c, b, a)
```

Wywołajmy funkcję dla trzech krążków. Otrzymamy listę ruchów potrzebnych do przeniesienia krążków ze słupka A na słupek C przy użyciu słupka B.

```
1 hanoi(3, 'A', 'C', 'B')
2
3 przenieś A -> C
4 przenieś A -> B
5 przenieś C -> B
6 przenieś A -> C
7 przenieś B -> A
8 przenieś B -> C
9 przenieś A -> C
```

Po podwojeniu liczby krążków lista operacji wydłuża się dziewięciokrotnie. Krążków jest sześć, należy zatem wykonać 63 ruchy ( $2^6 - 1$ ).

```
1 hanoi(6, "A", "B", "C")
2
3 przenieś A -> C
4 przenieś A -> B
5 przenieś C -> B
6 przenieś A -> C
7 przenieś B -> A
8 przenieś B -> C
9 przenieś A -> C
10 przenieś A -> B
11 przenieś C -> B
12 przenieś C -> A
13 przenieś B -> A
14 przenieś C -> B
15 przenieś A -> C
16 przenieś A -> B
17 przenieś C -> B
18 przenieś A -> C
19 przenieś B -> A
20 przenieś B -> C
21 przenieś A -> C
22 przenieś B -> A
23 przenieś C -> B
24 przenieś C -> A
25 przenieś B -> A
26 przenieś B -> C
27 przenieś A -> C
28 przenieś A -> B
29 przenieś C -> B
30 przenieś A -> C
31 przenieś B -> A
32 przenieś B -> C
33 przenieś A -> C
34 przenieś A -> B
35 przenieś C -> B
36 przenieś C -> A
37 przenieś B -> A
38 przenieś C -> B
```

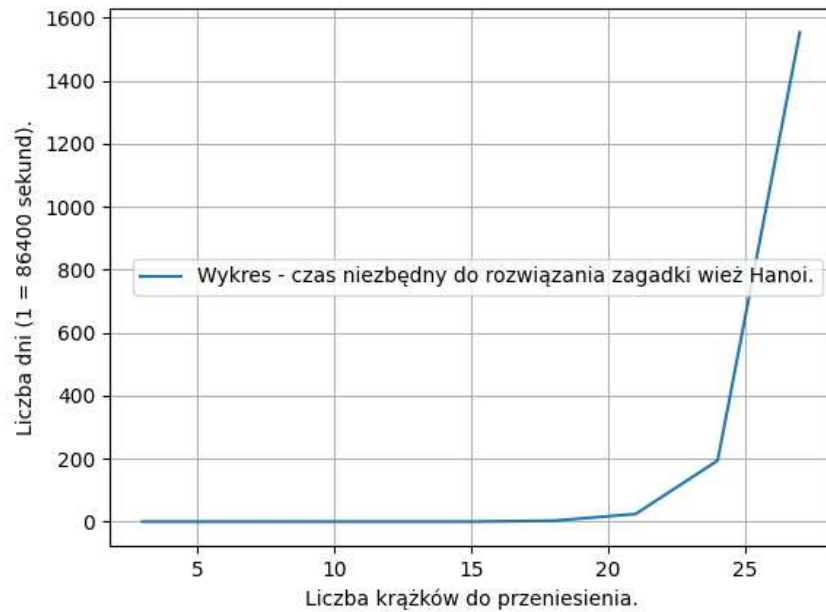
```
39 przenieś A -> C
40 przenieś A -> B
41 przenieś C -> B
42 przenieś C -> A
43 przenieś B -> A
44 przenieś B -> C
45 przenieś A -> C
46 przenieś B -> A
47 przenieś C -> B
48 przenieś C -> A
49 przenieś B -> A
50 przenieś C -> B
51 przenieś A -> C
52 przenieś A -> B
53 przenieś C -> B
54 przenieś A -> C
55 przenieś B -> A
56 przenieś B -> C
57 przenieś A -> C
58 przenieś A -> B
59 przenieś C -> B
60 przenieś C -> A
61 przenieś B -> A
62 przenieś C -> B
63 przenieś A -> C
64 przenieś A -> B
65 przenieś C -> B
```

### Dla zainteresowanych

Możemy sprawdzić, ile razy wykonywana jest funkcja rekurencyjna, czyli taka, która wywołuje wielokrotnie siebie samą. W celu sprawdzenia liczby wykonywanych przez nią operacji posłużymy się zmienną, która istnieje w [przestrzeni nazw](#) poza funkcją. Aby zmieniać jej wartość, zadeklarujemy ją za pomocą słowa kluczowego `global`. Dzięki temu będziemy mogli zmieniać wartość zmiennej, która istnieje poza funkcją.

Zdefiniujemy funkcję, która obliczy liczbę wywołań funkcji, niezbędnych dla rozwiązania zagadki Wież Hanoi. Przyjmijmy założenie, że jedno wywołanie trwa 1 sekundę.

W celu zobrazowania wyników obliczeń posłużymy się biblioteką [matplotlib](#) i napiszemy program wyświetlający wykres zależności liczby kroków od liczby krążków. Przyjmiemy, że przeniesienie 1 krążka trwa 1 sekundę. Wyniki podamy w dobach, a więc na wykresie 1 jednostka osi Y będzie odpowiadać 86400 sekund (przenosin krążków).



```

1 import matplotlib.pyplot as plt
2
3 def hanoi(n, a, b, c):
4     global zlicz
5
6     if n == 0:
7         return None
8     if n > 0:
9         hanoi(n - 1, a, c, b)
10        zlicz += 1
11        hanoi(n - 1, c, b, a)
12
13 # przygotowujemy listę, dla której będziemy obliczać
14 X = [x for x in range(3, 28, 3)]
15 Y = []
16
17 # w pętli wykonujemy kolejne obliczenia
18 for e in X:
19     zlicz = 0
20     hanoi(e, "A", "B", "C")
21     y = round(zlicz / 86400, 0)
22     print(f"Liczę dla: {e} krążków = {zlicz} kroków, {y} dni.")
23     Y.append(y)
24
25 # generujemy wykres
26 plt.plot(X, Y)

```

```
27 plt.grid(True)
28 plt.xlabel("Liczba krążków do przeniesienia.")
29 plt.ylabel("Liczba dni (1 = 86400 sekund).")
30 plt.legend(["Wykres - czas niezbędny do rozwiązania zagadki wie
31 plt.show()
32
33 # wyniki
34 # Liczę dla: 3 krążków = 7 kroków, 0.0 dni.
35 # Liczę dla: 6 krążków = 63 kroków, 0.0 dni.
36 # Liczę dla: 9 krążków = 511 kroków, 0.0 dni.
37 # Liczę dla: 12 krążków = 4095 kroków, 0.0 dni.
38 # Liczę dla: 15 krążków = 32767 kroków, 0.0 dni.
39 # Liczę dla: 18 krążków = 262143 kroków, 3.0 dni.
40 # Liczę dla: 21 krążków = 2097151 kroków, 24.0 dni.
41 # Liczę dla: 24 krążków = 16777215 kroków, 194.0 dni.
42 # Liczę dla: 27 krążków = 134217727 kroków, 1553.0 dni.
```

### Ważne!

Musimy pamiętać, że język Python jest dynamicznie typowany, a więc każda zmienna może w dowolnym momencie przechowywać dane różnego typu. Dodatkowo jest to interpreter, a nie kompilator. Z tych powodów nawet proste obliczenia zajmują dużo czasu procesora, dlatego odradzamy liczenie w taki sposób dla więcej niż 30 krążków. Na komputerze wyposażonym w procesor Core i 5 obliczenia dla 33 i więcej krążków trwają kilka godzin.



## Problem 1

### Zadanie:

Zapisz program wykorzystujący rekurencję, który wypisze kolejne kroki, jakie trzeba wykonać, aby przenieść wieżę składającą się z  $n$  krążków o różnej średnicy ze słupka A na słupek C. Swój program przetestuj dla 3 krążków.

### Zasady gry:

- Gra składa się z trzech słupków i zestawu krążków o różnych średnicach.
- Na jednym słupku umieszczona jest wybudowana wieża.
- Celem gry jest odbudowanie wieży na trzecim słupku.
- Nie wolno położyć krążka o większej średnicy na krążku o mniejszej średnicy.
- Na raz można przenosić tylko jeden krążek (znajdujący się na szczycie wieży).
- Do przenoszenia można wykorzystać dodatkowy słupek.

### Specyfikacja:

#### *Dane:*

- $n$  – liczba krążków w grze, liczba naturalna
- A, B, C – symboliczne nazwy słupków, znaki

#### *Wynik:*

Program na wyjściu standardowym wypisuje w kolejnych liniach kroki, które prowadzą do rozwiązania zagadki Wież Hanoi.

### Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w filmie.



# Zagadka Wież Hanoi

Implementacja algorytmu w języku Python



Film dostępny pod adresem </preview/resource/RoxU7lRTafplf>

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Film nawiązujący do zagadki Wież Hanoi.

---

Plik o rozmiarze 388.00 B w języku polskim

## Już wiesz

Podsumujmy najważniejsze informacje:

- liczba operacji niezbędnych do rozwiązania zagadki Wież Hanoi rośnie wykładniczo wraz ze wzrostem liczby krążków składających się na wieżę,
- operacje przenoszenia krążków składających się na Wieże Hanoi przypominają sposób odwoływania się procesora do elementów stosu.

## Słownik

### funkcja rekurencyjna

funkcja, która wywołuje samą siebie, aż do momentu osiągnięcia stanu początkowego – każda taka funkcja zawiera tzw. warunek zakończenia rekurencji (zwany także warunkiem początkowym) oraz przynajmniej jedno miejsce, w którym procedura wywołuje samą siebie

### matplotlib

biblioteka służąca do przedstawienia obrazów złożonych z punktów o współrzędnych (x, y), np. wykresów, histogramów, rozkładów itp.; moduł `matplotlib` nie jest dostępny

w standardowej instalacji języka Python – można go zainstalować, korzystając z mechanizmu `pip`

### **przypadek rekurencyjny**

warunek w funkcji rekurencyjnej, po spełnieniu którego funkcja ponownie wywołuje siebie samą

### **przypadek bazowy**

warunek w funkcji rekurencyjnej, po spełnieniu którego funkcja nie wywołuje samej siebie po raz kolejny

### **rekurencja**

proces polegający na wywoływaniu funkcji przez siebie samą do momentu rozwiązania określonego zadania

### **stos**

struktura danych, w której informacje są pobierane ze szczytu i na niego odkładane; struktura typu LIFO (*Last In, First Out* – ostatni na wejściu, pierwszy na wyjściu)

### **zagadka Wież Hanoi**

łamigłówka polegająca na odbudowaniu, z zachowaniem kształtu, wieży z krążków o różnych średnicach; podczas przekładania nie wolno kłaść krążka o większej średnicy na mniejszy, ani przekładać kilku krążków jednocześnie

### **przestrzeń nazw**

# Symulacja interaktywna

---

## Polecenie 1

Uzupełnij tabelę.

By sprawdzić, jak wygląda formuła zamieszczona w odpowiedniej komórce trzeciej kolumny (**liczba ruchów**), kliknij ją dwa razy.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

## Polecenie 2

Zapoznaj się z prezentacją dotyczącą rozwiązania zagadki Wież Hanoi dla trzech krążków. W prezentacji słupki są oznaczone jako A, B, C, a krążki - k1, k2, k3 (k1 jest najmniejszy i znajduje się na górze, a k3 to największy krążek, umieszczony na spodzie). Słupki będą reprezentowane przez obiekty typu `list`.

Do przenoszenia wykorzystamy metody `pop()` oraz `append()`.

## Ważne!

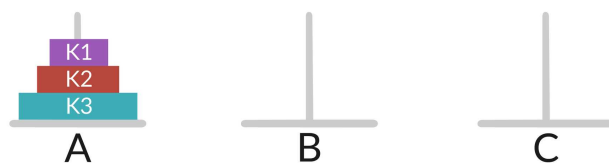
Metody `pop()` i `append()` są metodami `list` w języku Python służącymi odpowiednio do usuwania elementu z listy i dodawania elementów do końca listy.

Przykłady użycia:

```
1 lista_elementów = [1, 2, 3, 4, 5]
2 lista_elementów.append(6)
3 print(lista_elementów)
4
5 # [1, 2, 3, 4, 5, 6]
```

```
1 lista_zakupów = ['jajka', 'mleko', 'chleb']
2 ostatni_element = lista_zakupów.pop()
3 print(ostatni_element)
4 # 'chleb'
5
6 print(lista_zakupów)
```





1

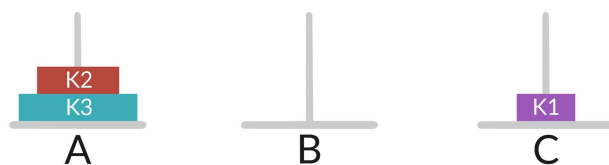
Na początku wszystkie krążki znajdują się na słupku A:

```
1 A = [k3, k2, k1]
2 B = []
3 C = []
```

2

Przenosimy k1 z A na C – wykonujemy operacje pobierania elementu ze stosu i dodania go do innego stosu:

```
1 krazek = A.pop()
2 C.append(krazek)
```



3

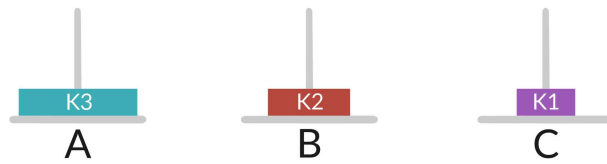
Układ krążków po wykonaniu operacji `pop()` i `append()`:

```
1 A = [k3, k2]
2 B = []
3 C = [k1]
```

4

Przenosimy k2 z A na B – wykonujemy operacje pobierania elementu ze stosu i dodania go do innego stosu:

```
1 krazek = A.pop()
2 B.append(krazek)
```



5

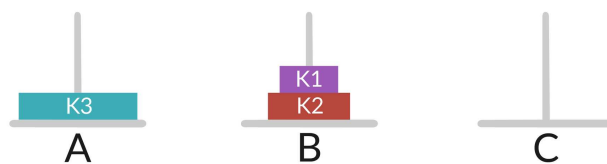
Układ krążków po operacji pop() i append():

```
1 A = [k3]
2 B = [k2]
3 C = [k1]
```

6

Przenosimy k1 z C na B – wykonujemy operacje pobierania elementu ze stosu i dodania go do innego stosu:

```
1 krazek = C.pop()
2 B.append(krazek)
```



7

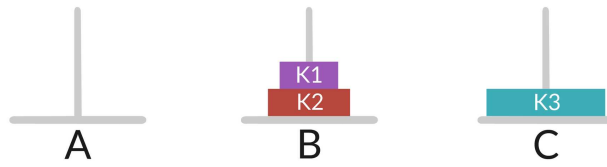
Układ krążków po operacji pop() i append():

```
1 A = [k3]
2 B = [k2, k1]
3 C = []
```

8

Przenosimy k3 z A na C – wykonujemy operacje pobierania elementu ze stosu i dodania go do innego stosu:

```
1 krazek = A.pop()
2 C.append(krazek)
```



9

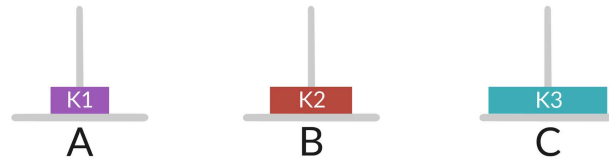
Układ krążków po operacji pop() i append():

```
1 A = []
2 B = [k2, k1]
3 C = [k3]
```

10

Przenosimy k1 z B na A – wykonujemy operacje pobierania elementu ze stosu i dodania go do innego stosu:

```
1 krazek = B.pop()
2 A.append(krazek)
```



11

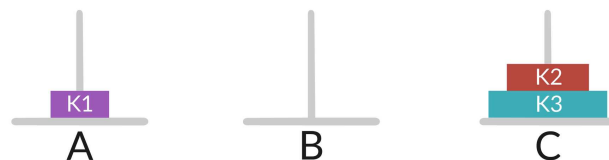
Układ krążków po operacji `pop()` i `append()`:

```
1 A = [k1]
2 B = [k2]
3 C = [k3]
```

12

Przenosimy k2 z B na C – wykonujemy operacje pobierania elementu ze stosu i dodania go do innego stosu:

```
1 krazek = B.pop()
2 C.append(krazek)
```



13

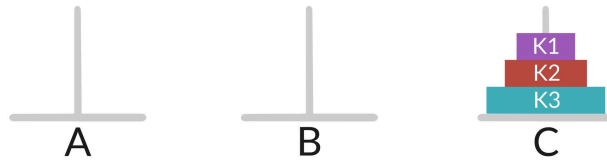
Układ krążków po operacji `pop()` i `append()`:

```
1 A = [k1]
2 B = []
3 C = [k3, k2]
```

14

Przenosimy k1 z A na C – wykonujemy operacje pobierania elementu ze stosu i dodania go do innego stosu:

```
1 krazek = A.pop()  
2 C.append(krazek)
```



15

Układ krążków po operacji pop() i append():

```
1 A = []  
2 B = []  
3 C = [k3, k2, k1]
```




16

Jak widzimy, w przypadku trzech krążków wystarczy wykonać siedem operacji w celu rozwiązania zagadki Wież Hanoi. Liczba ta rośnie wykładniczo wraz ze zwiększaniem liczby krążków.



# Sprawdź się

---

Pokaż ćwiczenia:   

## Ćwiczenie 1



Napisz program wyświetlający liczbę ruchów niezbędnych do rozwiązania zagadki wież Hanoi dla  $k$  krążków. Twój program powinien wypisać na ekranie liczbę wykonanych ruchów. Przetestuj swój program dla 7 krążków.

### Specyfikacja problemu:

*Dane:*

- $k$  – liczba krążków; liczba naturalna

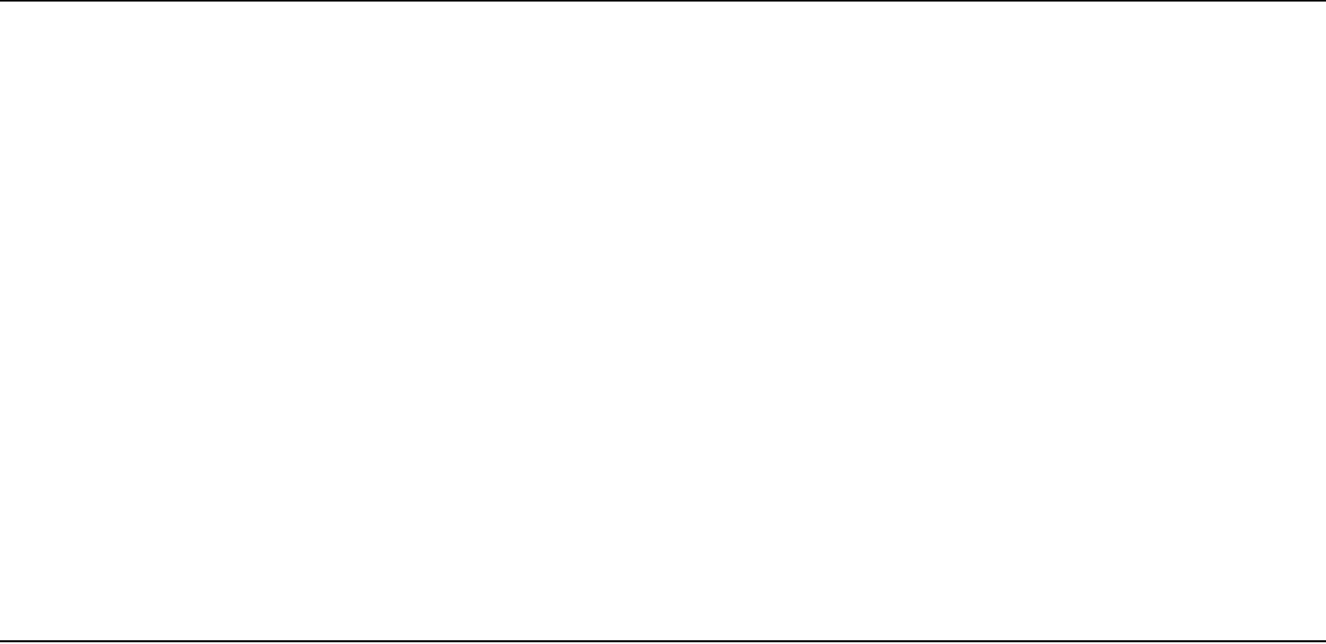
*Wynik:*

Program wypisuje liczbę ruchów, niezbędnych do rozwiązania zagadki wież Hanoi dla  $k$  krążków.

### Twoje zadania

1. Program wyświetla liczbę ruchów opisujących rozwiązanie zagadki Wież Hanoi dla  $k$  krążków.

```
1 def hanoi(n, a, b, c):
2     global zlicz
3
4     ## tutaj wpisz swój kod
5
6 k = 7
7 zlicz = 0
8 hanoi(k, "A", "B", "C")
9 print(zlicz)
```





Napisz program, który będzie wypisywał m-ty ruch, opisujący rozwiązanie Wież Hanoi dla n krążków. Niech odpowiedź będzie w postaci: „Przeniesienie z X do Y”. Przetestuj swój program dla wieży składającej się z ośmiu krążków i wypisz piąty ruch.

### Specyfikacja problemu:

*Dane:*

- n – liczba krążków; liczba naturalna
- m – numer ruchu, który ma zostać wypisany; liczba naturalna dodatnia,  $m < 2^n$

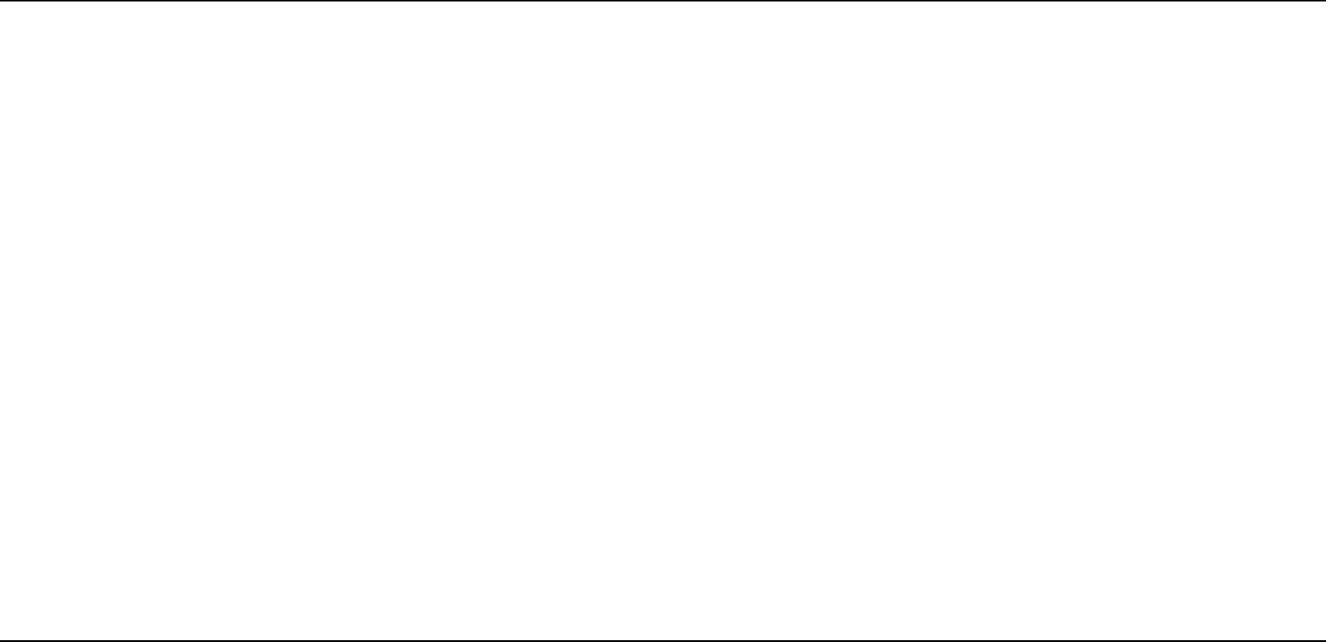
*Wynik:*

Program wyświetla m-ty ruch opisujący rozwiązanie zagadki Wież Hanoi dla n krążków.

### Twoje zadania

1. Program wyświetla opis m-tego ruchu, rozwiązującego zagadkę Wież Hanoi dla n krążków.

```
1 def hanoi(n, a, b, c):
2     global zlicz
3     // Tutaj dopisz kod
4
5     zlicz = 0
6     n = 8
7     m = 5
8     hanoi(n, "A", "B", "C")
```





# Dla nauczyciela

---

**Autor:** Adam Jurkiewicz

**Przedmiot:** Informatyka

**Temat:** Zagadka Wież Hanoi w języku Python

**Grupa docelowa:**

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres rozszerzony

**Podstawa programowa:**

Zakres podstawowy i rozszerzony

Cele kształcenia – wymagania ogólne

1. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.
2. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Zakres rozszerzony

Treści nauczania – wymagania szczegółowe

III. I + II. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3. objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

2) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),

**Kształtowane kompetencje kluczowe:**

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

### **Cele operacyjne (językiem ucznia):**

- Przypomnisz sobie algorytm rozwiązania zagadki Wież Hanoi.
- Przeanalizujesz implementację tego algorytmu w języku Python.
- Rozwiążesz samodzielnie zadania związane z zagadką Wież Hanoi.

### **Strategie nauczania:**

- konstruktywizm;
- konektywizm.

### **Metody i techniki nauczania:**

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimediu i ćwiczeń interaktywnych;
- metody aktywizujące.

### **Formy pracy:**

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

### **Środki dydaktyczne:**

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

### **Przebieg lekcji**

#### **Przed lekcją:**

1. Nauczyciel prosi uczniów o przypomnienie sobie najważniejszych informacji na temat zagadki wież Hanoi.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Zagadka Wież Hanoi w języku Python”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj” dotyczącymi programowania.

#### **Faza wstępna:**

1. Chętna lub wybrana osoba przypomina najważniejsze informacje na temat zagadki wież Hanoi.
2. Przedstawienie tematu i celów zajęć.

- 3. Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np.
- co to jest przestrzeń nazw?
  - do czego służy biblioteka matplotlib?
  - czym jest rekurencja?
  - na czym polega zagadka Wież Hanoi?

Chętni uczniowie udzielają na nie odpowiedzi.

#### **Faza realizacyjna:**

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Przeczytaj”. Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. Uczniowie uzupełniają symulację interaktywną. Charakteryzują, jak wzrasta złożoność obliczeniowa algorytmu obliczające rozwiązanie zagadki Wież Hanoi. Szukają informacji na temat mocy obliczeniowej superkomputerowych sprzed dekady oraz współczesnych. Określają skok technologiczny na podstawie tego, jak wygląda czas rozwiązywania zagadki dla trzech, 10 oraz 64 krążków.
3. Uczniowie zapoznają się z filmem oraz prezentacją. Wykonują ćwiczenia.
4. Uczniowie rozwiązują ćwiczenia z sekcji „Sprawdź się”.

#### **Faza podsumowująca:**

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie”. W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązywania ćwiczeń z programowania w języku Python.

#### **Praca domowa:**

1. Opracuj porównanie dwóch modeli rozwiązywania zagadki Wież Hanoi – wykorzystującego iterację oraz rekurencję.

#### **Materiały pomocnicze:**

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

#### **Wskazówki metodyczne:**

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Symulacja interaktywna”, „Sprawdź się” jako materiał do lekcji powtórkowej.