



Rekurencja w języku Python

- [Wprowadzenie](#)
- [Przeczytaj](#)
- [Film samouczek](#)
- [Sprawdź się](#)
- [Dla nauczyciela](#)



Rekurencja w języku Python

Źródło: schrupp, domena publiczna.

Rekurencja jest metodą rozwiązywania problemów, która korzysta z rozwiązań innych problemów. Jej wyjątkowość polega na tym, że za „innymi problemami” kryje się problem właściwy, ale rozwiązywany dla podproblemu, który powstał w wyniku podziału problemu wyjściowego.

W e-materiale [Rekurencja](#) przedstawiliśmy podstawowe informacje dotyczące omawianego zagadnienia. Kolejnym krokiem będzie implementacja w wybranym języku programowania. Ten e-materiał poświęcony jest implementacji algorytmów rekurencyjnych w języku Python.

Więcej przykładów, ćwiczeń i rozwiązań znajdziesz w:

- [Rekurencja – ćwiczenia](#),
- [Rekurencja w zadaniach](#).

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

- [Rekurencja w języku C++](#).
- [Rekurencja 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

- Przeanalizujesz rekurencyjne podejście do algorytmu Euklidesa i poznasz zalety jego stosowania.
- Zapoznasz się z różnymi implementacjami rekurencyjnego algorytmu Euklidesa.
- Wyjaśnisz, czym rekurencja różni się od iteracji.
- Rozwiążesz kilka zadań z wykorzystaniem rekurencji.

Przeczytaj

W e-materiale [Algorytm Euklidesa](#) poznaliśmy algorytm służący do znajdowania największego wspólnego dzielnika dwóch liczb naturalnych. Wersję iteracyjną Algorytmu Euklidesa w języku Python omówiliśmy w e-materiale [Algorytm Euklidesa w języku Python](#). Tym razem zbadamy podejście rekurencyjne.

Algorytm Euklidesa z odejmowaniem – implementacja algorytmu w języku Python

Zaimplementujmy z użyciem rekurencji algorytm Euklidesa w języku Python.

W funkcji `NWD()`, która przyjmuje dwie zmienne, użylibyśmy instrukcji warunkowej `if`, w której warunek będzie taki sam, jak w przypadku pętli `while` w wykonaniu iteracyjnym, czyli $a \neq b$.

Jeżeli warunek jest spełniony, to znaczy $a \neq b$, funkcja będzie wywoływana rekurencyjnie. W przypadku gdy wartość a będzie większa od wartości b , funkcja zwróci $NWD(a - b, b)$, a w przeciwnym przypadku $NWD(a, b - a)$. W algorytmie korzystamy z równości $NWD(a, b) = NWD(a - b, b)$ dla $a > b$ lub $NWD(a, b) = NWD(a, b - a)$ dla $b > a$, które pokazaliśmy w e-materiale [Algorytm Euklidesa](#).

Jeżeli warunek nie jest spełniony, funkcja zwróci a . Oczywiście mogłaby też zwrócić b – jeżeli warunek nie zostanie spełniony, będzie to oznaczać, że wartość obu zmiennych jest identyczna.

Nasza funkcja wygląda następująco:

```
1 def NWD(a, b):
2     if a != b:
3         if a > b:
4             return NWD(a - b, b)
5         else:
6             return NWD(a, b - a)
7     return a
```

Algorytm Euklidesa z dzieleniem – implementacja algorytmu w języku Python

Zadeklarujemy funkcję `NWD()`. Przyjmie ona parametry `a` i `b`. W ciele funkcji umieścimy instrukcję warunkową `if`, której testem logicznym będzie `b != 0`. Jeżeli zależność zachodzi, funkcja zwróci samą siebie z argumentami `b` i `a % b`. W przeciwnym razie funkcja zwróci zmienną `a`. Kod wygląda następująco:

```
1 def NWD(a, b):
2     if b != 0:
3         return NWD(b, a % b)
4     else:
5         return a
```

Istotne jest, żeby funkcja zwróciła **przypadek podstawowy** w momencie, gdy `b` przyjmie wartość `0`. W innym przypadku w następnym wywołaniu funkcji pojawiłby się argument, który doprowadziłby do wykonania operacji niedozwolonej, czyli szukania reszty z dzielenia przez `0`.

Słownik

konkatenacja

łączenie ze sobą łańcuchów znaków

modulo

reszta z dzielenia dwóch liczb całkowitych, w języku Python oznaczana znakiem `%` (procent), np.: `22 % 5 = 2`

NWD

największy wspólny dzielnik dwóch liczb naturalnych – największa liczba naturalna, która dzieli obie te liczby bez reszty

przypadek podstawowy

przypadek, w którym funkcja rekurencyjna zwraca konkretną wartość, a nie wywołanie samej siebie

rekurencja

technika programowania polegająca na odwoływaniu się procedur lub funkcji do samych siebie, aż do momentu spełnienia warunku podstawowego

Film samouczek

Polecenie 1

Napisz program, który obliczy element ciągu o indeksie n .

Przetestuj jego działanie dla ciągu o wzorze:

$$a_n = \begin{cases} 1 & \text{gdzie } n = 0 \\ a_{n-1} + r & \text{gdzie } n > 0 \end{cases}$$

gdzie $a_0 = 1, r = 5, n = 3$.

Specyfikacja problemu:

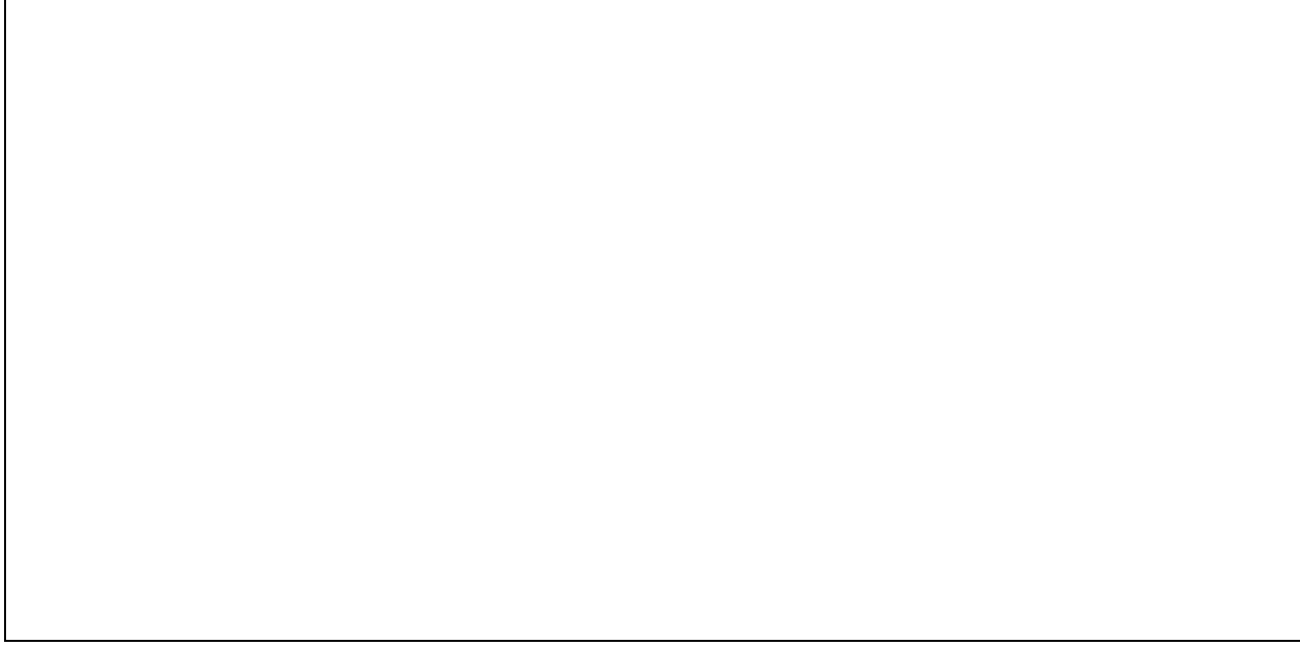
Dane:

- n – liczba naturalna
- r – liczba naturalna

Wynik:

- wyraz ciągu o indeksie n

1



Polecenie 2

Napisz program, który policzy największy wspólny dzielnik (NWD) wskazanych liczb.

Przetestuj jego działanie dla $n = 24$ oraz $k = 28$.

Rekurencyjny wzór na wyznaczanie NWD:

$$nwd(k, n) = \begin{cases} n & \text{gdzie } k = 0 \\ nwd(n \bmod k, k) & \text{gdzie } k > 0 \end{cases}$$

Specyfikacja problemu:

Dane:

- n – liczba naturalna
- k – liczba naturalna

Wynik:

- NWD liczb n oraz k

1

Polecenie 3

Porównaj swoje programy z przedstawionymi w filmie.



Wprowadzenie do rekurencji

Implementacja w języku Python



Film dostępny pod adresem </preview/resource/R19pudypm55e3>

Film nawiązujący do treści materiału wprowadzenia do rekurencji, implementacja w języku Python.

Plik o rozmiarze 220.00 B w języku polskim

Polecenie 4

Zastanów się, jak wyglądałoby rozwiązanie problemu postawionego w filmie, gdyby zapisać je iteracyjnie. Jak wyglądałaby wydajność programu?

Polecenie 5

Zastanów się, jakie różnice dostrzegasz w zaprezentowanej implementacji w stosunku do tej, którą przedstawiono wcześniej.

Ważne!

W filmie mowa o tym, że dodanie poprzednich wyrazów wedle wyznaczonego wzoru jest możliwe z wykorzystaniem definicja rekurencyjnej – możemy tak zrobić w przypadku **ciągów arytmetycznych**.

Sprawdź się

Pokaż ćwiczenia:   

Ćwiczenie 1



Wskaż poprawny zapis funkcji rekurencyjnej.

1 # funkcje rekurencyjne



```
1 def rekurencja(numer):  
2     return 1 if numer == 0 else rekurencja(numer-1)
```



```
1 def rekurencja(numer):  
2     wynik = 1  
3     for i in range(numer):  
4         wynik=pow(numer,2)  
5     return wynik
```



```
1 def rekurencja(numer):  
2     wynik = 1  
3     for i in range(numer):  
4         wynik = rekurencja(i)  
5     return wynik
```



Napisz program, który obliczy sumę n kolejnych liczb naturalnych z użyciem rekurencji.

Specyfikacja problemu:

Dane:

- n – liczba naturalna

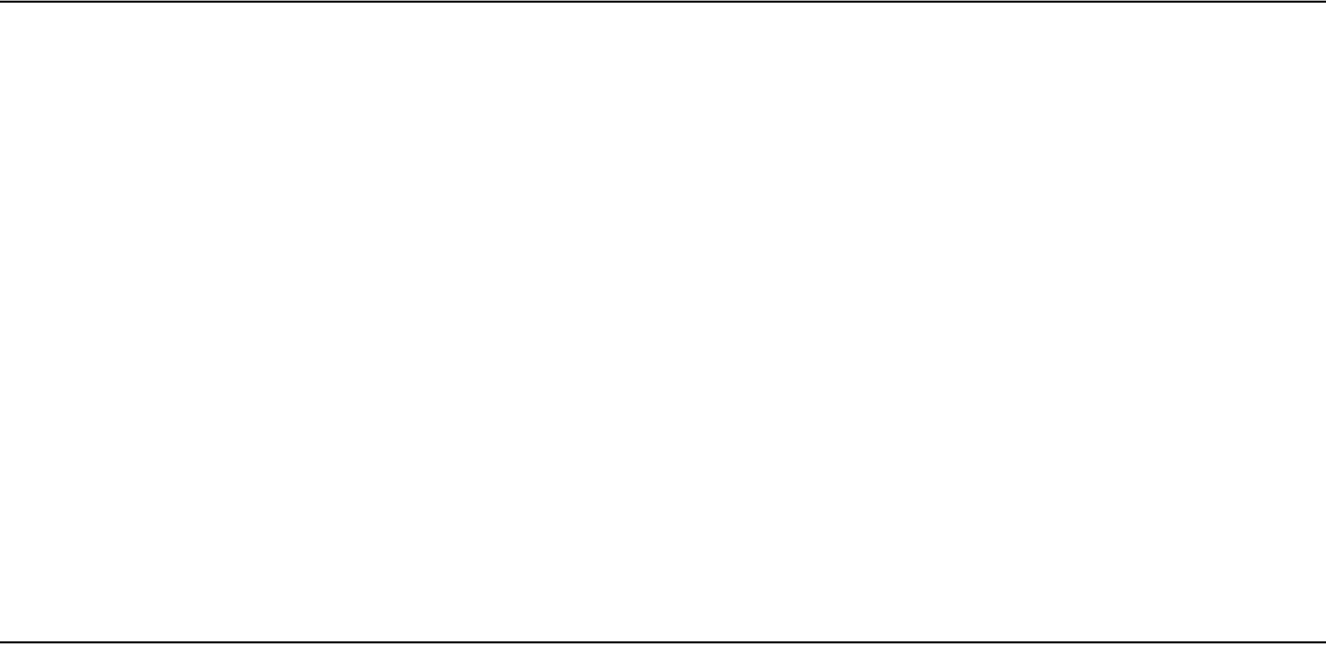
Wynik:

- x – liczba naturalna; suma n kolejnych liczb naturalnych

Twoje zadania

1. Zdefiniowanie funkcji suma.
2. Sprawdzenie, czy funkcja dla podanego parametru zwraca typ int.
3. Funkcja suma zwraca wartość 55 dla danych wejściowych $n = 10$.
4. Funkcja suma zwraca wartość 105 dla danych wejściowych $n = 14$.

```
1 def suma(n):  
2     # Tu uzupełnij kod  
3     return None
```



Ćwiczenie 3



Napisz program, który wygeneruje n -te słowa Thuego-Morse'a oddzielone znakiem nowej linii. Kolejne wartości n podane są w tablicy.

Słowa Thuego-Morse'a tworzone są według poniższej zasady:

$$T_n = \begin{cases} 0 & \text{gdy } n = 0 \\ T_{n-1} \cdot \overline{T_{n-1}} & \text{gdy } n > 0 \end{cases}$$

Gdzie:

$x.y$ oznacza **konkatenację** słowa x ze słowem y – łączymy słowo x oraz y , przy czym słowo x występuje na pierwszej pozycji.

$\bar{a}$ oznacza negację bitów w słowie a – znaki 0 zamieniamy na 1 i odwrotnie.

Przykład:

$$T_0 = 0$$

$$T_1 = 01$$

$$T_2 = 0110$$

$$T_3 = 01101001$$

Specyfikacja problemu:

Dane:

- `tablica` – tablica liczb całkowitych

Wynik:

- `x` – ciąg znaków

Twoje zadania

1. Zdefiniowanie funkcji T.
2. Funkcja zwraca wartość "0" dla $n = 0$.
3. Funkcja zwraca wartość "0110" dla $n = 2$.
4. Funkcja zwraca wartość
"011010011001011010010110011001100110010110011010010110100101101001100
10110" dla $n = 6$.

```
1 def T(n):  
2     # Tu uzupełnij kod  
3     pass  
4  
5 tablica = [0, 2, 6]
```

```
1
```






Napisz funkcję skracającą ułamki. Powinna przyjmować dwie liczby naturalne: `licznik` i `mianownik`, a zwracać napis będący skróconą postacią ułamka; np. dla liczb 2 i 4 funkcja powinna zwrócić `1 / 2`.

Specyfikacja problemu:

Dane:

- `licznik` – liczba całkowita
- `mianownik` – liczba całkowita

Wynik:

Program na wyjściu standardowym zwróci skrócony ułamek.

Twoje zadania

1. Zdefiniowanie funkcji `skroc_ułamek`.
2. Funkcja zwraca napis `"1 / 3"` dla wartości 3 oraz 9.
3. Funkcja zwraca napis `"1 / 6"` dla wartości 6 oraz 36.

```
1 def skroc_ułamek(licznik, mianownik):  
2     # Tu uzupełnij kod  
3     pass
```


Dla nauczyciela

Autor: Adam Jurkiewicz

Przedmiot: Informatyka

Temat: Rekurencja w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I + II. Zakres rozszerzony. 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:

b) 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):

- Przeanalizujesz rekurencyjne podejście do algorytmu Euklidesa i poznasz zalety jego stosowania.
- Zapoznasz się z różnymi implementacjami rekurencyjnego algorytmu Euklidesa.
- Wyjaśnisz, czym rekurencja różni się od iteracji.
- Rozwiążesz kilka zadań z wykorzystaniem rekurencji.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimediu i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

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. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Rekurencja w języku Python”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów lekcji. Określenie wiążących dla uczniów kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują przykład z sekcji „Przeczytaj” i testują zaprezentowane rozwiązania na swoim komputerze.
2. **Praca z multimediami.** Uczniowie wspólnie zapoznają się z treścią i filmem umieszczonym w sekcji „Film samouczek”. Następnie indywidualnie przechodzą do wykonywania poleceń: 1, 3 i 5. W kolejnym kroku przedstawiają swoje rezultaty osobie z ławki. W przypadku wątpliwości i trudności przy rozwiązywaniu zadania nauczyciel omawia je na forum klasy.
3. **Ćwiczenie umiejętności.** Uczniowie realizują indywidualnie ćwiczenia nr 1–3, po ich wykonaniu porównują otrzymane wyniki z inną osobą.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia z programowania w Pythonie, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie szukają problemów matematycznych i informatycznych, w których wykorzystanie rekurencji może być przydatne.

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”, „Film samouczek”, „Sprawdź się” jako materiał do lekcji powtórkowej.