Stosowanie funkcji w języku Python do realizacji algorytmu Euklidesa z odejmowaniem

-

Stosowanie funkcji w języku Python do realizacji algorytmu Euklidesa z odejmowaniem

Logo języka Python

Algorytm Euklidesa opisuje metodę znajdowania największego wspólnego dzielnika dwóch liczb naturalnych. Nazwa pochodzi od greckiego matematyka Euklidesa, który jako pierwszy opisał algorytm w IV w p.n.e., chociaż nie był jego autorem. Problem wyznaczania NWD rozwiązywany jest na wiele sposobów, w tym materiale wykorzystamy wersję opartą na odejmowaniu.

> **Aby zrozumieć poruszane w tym materiale zagadnienia, przypomnij sobie:**

- Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych w języku Python
- Stosowanie instrukcji warunkowej w języku Python
- Wprowadzenie do programowania w języku Python

Twoje cele

- Przeanalizujesz zapis algorytmu Euklidesa znajdowania NWD opartego na odejmowaniu.
- Napiszesz program obliczający NWD dla liczb pobranych od użytkownika.
- Zastosujesz instrukcję iteracyjną (pętlę) i instrukcję warunkową.
- Zbadasz liczbę operacji potrzebnych do uzyskania wyniku.

NWD

Największy wspólny dzielnik (NWD) dwóch dodatnich liczb naturalnych `a` i `b` to największa liczba naturalna `d`, która jest dzielnikiem każdej z nich, czyli dzieli je bez reszty. Przyjmijmy, że będziemy ją oznaczać jako `d = NWD(a, b)`.

NWD ma wiele zastosowań, m.in. pozwala skracać ułamki zwykłe do postaci, w której licznik i mianownik są liczbami względnie pierwszymi, czyli `NWD(a, b) = 1`.

Algorytm Euklidesa z zastosowaniem odejmowania

Skonstruujemy algorytm, który obliczy największy wspólny dzielnik dwóch dodatnich liczb naturalnych.

Specyfikacja algorytmu:

Dane wejściowe:

- `a`, `b` – dodatnie liczby naturalne

Wynik:

- `d` - dodatnia liczba naturalna, `NWD(a, b)`

Do rozwiązania postawionego zadania użyjemy algorytmu Euklidesa w wersji z odejmowaniem.

> **Przykład 1**
>
> Przeanalizujmy wyznaczanie NWD według algorytmu Euklidesa z wykorzystaniem odejmowania dla danych: `a = 4`, `b = 10`.

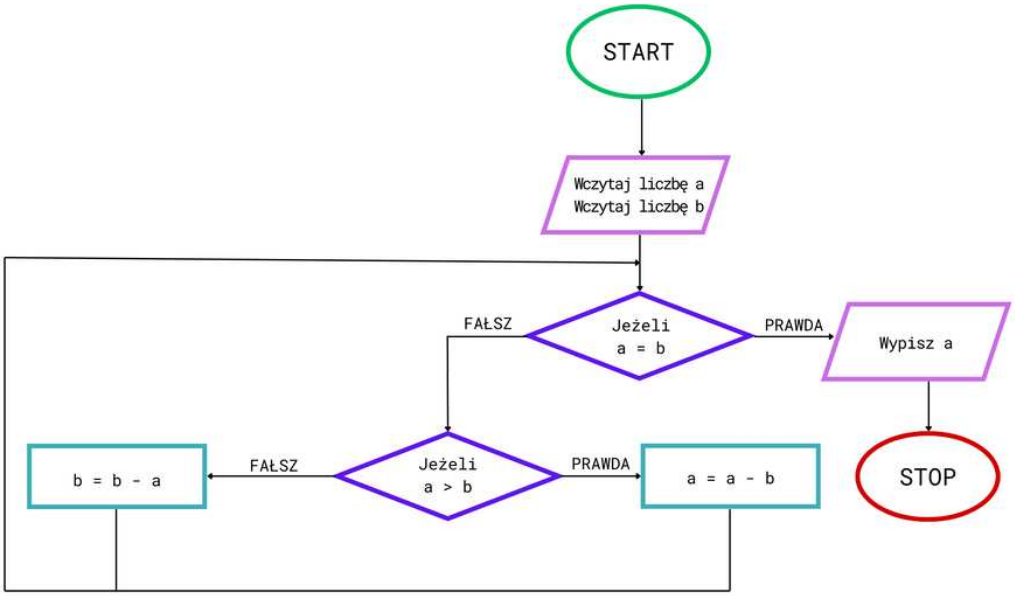
Powtórzenia	a	b	działanie
1	4	10	b = 10 - 4
2	4	6	b = 6 - 4
3	4	2	a = 4 - 2
4	2	2	-
Liczba powtórzeń: 3. **Wynik**: NWD (4, 10) = 2			

Z powyższego przykładu wynika, że dopóki liczby `a` i `b` są różne, w każdym powtórzeniu sprawdzamy, która liczba jest większa. Jeżeli `a > b`, obliczamy `a = a - b`, w przeciwnym razie wyliczamy `b = b - a`. Kiedy `a` i `b` są równe, wynikiem jest dowolna z tych liczb.

Zapiszmy algorytm w postaci **listy kroków**:

1. Wczytaj `a`, `b`.
2. Jeżeli `a` jest równe `b`, przejdź do kroku 5.
3. Jeżeli `a > b`, zmiennej `a` przypisz `a - b`, w przeciwnym razie zmiennej `b` przypisz `b - a`.
4. Przejdź do kroku 2.
5. Wypisz `a` i zakończ algorytm.

Przeanalizujmy również **schemat blokowy algorytmu**:



Schemat blokowy algorytmu

Z obydwu sposobów zapisu algorytmu wynika, że na początku pobieramy dane wejściowe, czyli dwie dodatnie liczby naturalne. Następnie porównujemy je i większą pomniejszamy o mniejszą. Operacje te powtarzają się, gdy liczby `a` i `b` są różne. Kiedy zrównają się, wyprowadzamy wynik, tj. dowolną z tych liczb, i kończymy.

Zapis algorytmu w postaci programu

Napiszemy prosty program, który będzie obliczał NWD według omówionego algorytmu.

> Do kodowania użyj dostępnego w swoim środowisku edytora, np. IDLE dołączonego do standardowej instalacji Pythona.

> **Przykład 2**
>
> Kopiujemy poniższy kod, wklejamy do pustego pliku i zapisujemy pod nazwą np. `nwd_odejmowanie.py`.

```
# pobierz liczbę naturalną a


# pobierz liczbę naturalną b


# pętla obliczająca NWD


# wypisanie wyniku
print("NWD(", a, ",", b,") = ", a)
```

Dane

Jak wynika z komentarzy w kodzie programu, na początku musimy pobrać dane, czyli dwie dodatnie liczby naturalne.

Polecenie 1

W pliku `nwd_odejmowanie.py` poniżej odpowiednich komentarzy dodaj instrukcje pobierania z klawiatury dwóch liczb naturalnych i zapamiętywania ich pod nazwami `a` i `b`. Użyj funkcji `input()` i `int()`.

Pętla

W najważniejszej części algorytmu porównujemy podane liczby, dopóki nie są równe. Warunek ten możemy zapisać jako `a != b`. Ponieważ nie wiemy, ile trzeba powtórzeń, aby uzyskać wynik, w programie musimy użyć pętli `while`. Wewnątrz pętli do sprawdzania, która z liczb jest większa, będziemy potrzebowali instrukcji warunkowej `if` badającej warunek `a > b`. Do zakodowania operacji zmniejszania większej liczby o mniejszą użyjemy instrukcji przypisania, np. `a = a - b`.

Polecenie 2

W pliku `nwd_odejmowanie.py` pod komentarzem `pętla obliczająca NWD` zapisz:

- pętlę, sprawdzającą warunek `a != b` – użyj instrukcji `while`,
- wewnątrz pętli warunek `a > b` oraz operacje zmniejszania `a = a - b` lub `b = b - a` – użyj instrukcji warunkowej `if`.

Przetestujmy działanie napisanego programu.

Polecenie 3

Uruchom program `nwd_odejmowanie.py` i przetestuj jego działanie dla przykładowych danych:

```
# Pierwszy zestaw danych
a = 4
b = 10

# Drugi zestaw danych
a = 1020
b = 20
```

Użycie funkcji

Obliczanie NWD to operacja, którą można w programie wykonywać wielokrotnie, np. gdy umożliwimy użytkownikowi wielokrotne podawanie liczb wejściowych. Kod realizujący tę operację umieścimy więc w funkcji, która do działania będzie wymagała dwóch parametrów, tzn. pobranych liczb.

Polecenie 4

Plik `nwd_odejmowanie.py` zapisz pod nazwą `nwd_odejmowanie_funkcja.py`.

Na początku pliku `nwd_odejmowanie_funkcja.py` umieść definicję funkcji `oblicz_nwd()`.

```
def oblicz_nwd(a, b):
    # kod pętli

```

W ciele funkcji należy umieścić zasadniczą część algorytmu, czyli pętlę obliczającą NWD.

Polecenie 5

W pliku `nwd_odejmowanie_funkcja.py` w funkcji `oblicz_nwd()` pod komentarzem `kod pętli` umieść kod pętli obliczającej NWD.

Wskazówka: zaznacz, wytnij i wklej dotychczasowy kod pętli razem z komentarzem `pętla obliczająca NWD`. Zwróć uwagę na poprawne wcięcia kodu.

Zwracanie wartości z funkcji

Po umieszczeniu pętli wyznaczającej NWD w funkcji musimy zdecydować, co zrobimy z wynikiem jej działania, tzn. ze zmienną `a`, która zawiera NWD. Moglibyśmy od razu wypisać wynik, ale w funkcjach, które coś obliczają, zazwyczaj się tego nie robi, chyba że testujemy ich działanie. Wynik działania funkcji najczęściej zwracamy do miejsca wywołania po to, żeby go wykorzystać, np. wypisać w terminalu albo użyć w dalszych obliczeniach.

Do zwracania wartości z funkcji używamy instrukcji `return wyrażenie`. Wyrażeniem może być zmienna, kilka zmiennych czy nawet wynik wywołania innej funkcji.
W naszym przypadku wystarczy, że po pętli umieścimy instrukcję `return a`.

> **Polecenie 6**
>
> W pliku `nwd_odejmowanie_funkcja.py` na końcu funkcji `oblicz_nwd()` umieść instrukcję zwracającą wynik: `return a`. Zwróć uwagę na jej wcięcie, powinno odpowiadać pętli `while`.

Wywołanie funkcji

Wykonywanie naszego programu rozpoczyna się od pobrania dwóch liczb od użytkownika. Po pobraniu tych danych musimy umieścić wywołanie naszej funkcji w postaci `oblicz_nwd(a, b)`. Liczby `a` i `b` przekazujemy jako argumenty.

Funkcja `oblicz_nwd()` zwraca wynik do miejsca wywołania. Możemy go przypisać do zmiennej `d`, a na koniec wypisać.

Polecenie 7

W pliku `nwd_odejmowanie_funkcja.py` po instrukcjach pobierania danych wejściowych umieść instrukcję przypisania i wywołania funkcji:

```
d = oblicz_nwd()
```

Końcową instrukcję `print()` zastąp kodem:

```
print("NWD(", a, ",", b,") = ", d)
```

Użycie funkcji ma jeszcze jedną zaletę – ułatwia wypisywanie komunikatu końcowego.

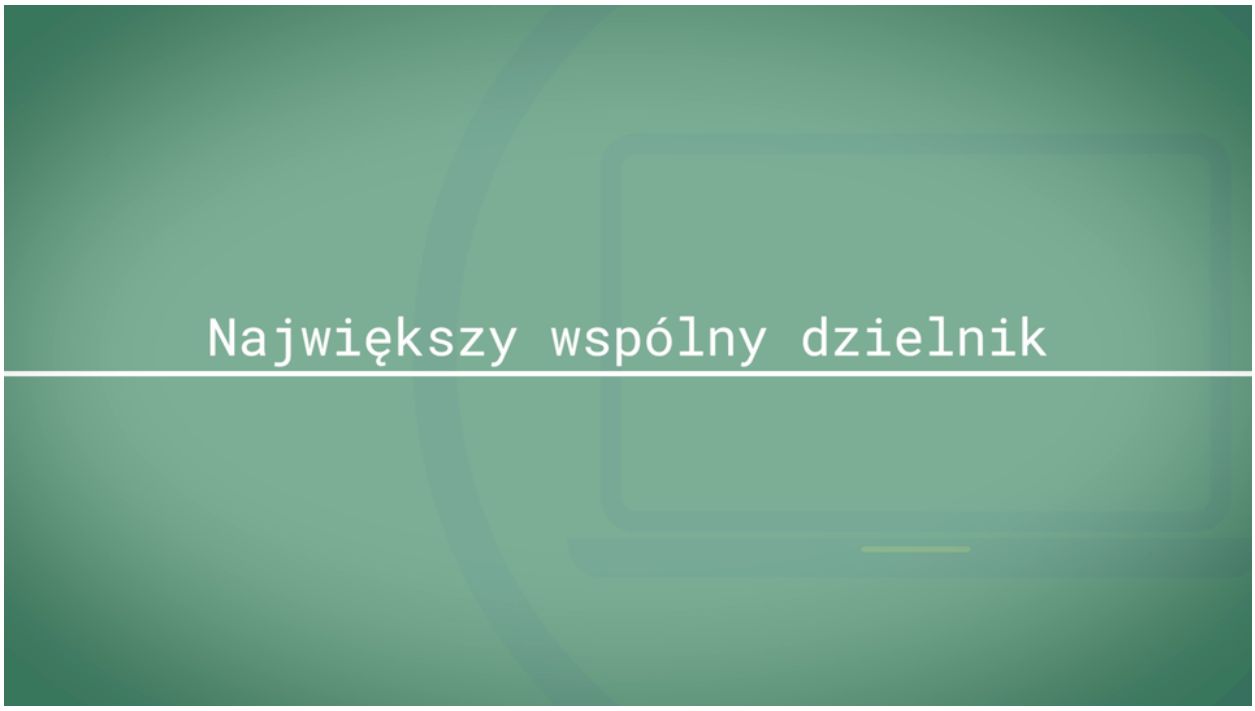
Notatnik

Miejsce na twoje notatki:

Film

Polecenie 8

Zapoznaj się z animacją, która pokazuje, w jaki sposób możemy uzyskać odpowiedź na pytanie o to, ile trzeba wykonać operacji odejmowania, zanim znajdziemy największy wspólny dzielnik z wykorzystaniem algorytmu Euklidesa.



Film dostępny pod adresem /preview/resource/R1E3bvJEBiR8S

Największy wspólny dzielnik

Animacja pod tytułem Największy wspólny dzielnik

Program `nwd_odejmowanie_licznik.py` zaprezentowany w animacji:

Plik o rozmiarze 574.00 B w języku polskim

Polecenie 9

Uruchom program `nwd_odejmowanie_licznik.py` i przetestuj jego działanie dla podanych danych:

```
# Pierwszy zestaw danych
a = 8
b = 32

# Drugi zestaw danych
a = 8
b = 2848

# Trzeci zestaw danych
a = 2
b = 10

# Czwarty zestaw danych
a = 2
b = 22222
```

Na podstawie wypisanych liczb operacji zastanów się, od czego zależy liczba wykonywanych operacji. Zapisz notatkę.

Polecenie 10

Do wykonania polecenia wykorzystaj następujące dwa zestawy danych:

```
# Pierwszy zestaw danych
a = 34
b = 16

# Drugi zestaw danych
a = 16
b = 34
```

Przy użyciu programu `nwd_odejmowanie_licznik.py` przeprowadź kolejne testy dla podanych danych i zanotuj wyniki.

Zastanów się, czy liczba wykonanych operacji zależy od kolejności danych wejściowych.

Polecenie 11

Przy użyciu programu `nwd_odejmowanie_licznik.py` przeprowadź testy dla wybranych przez siebie liczb. Zanotuj wyniki.

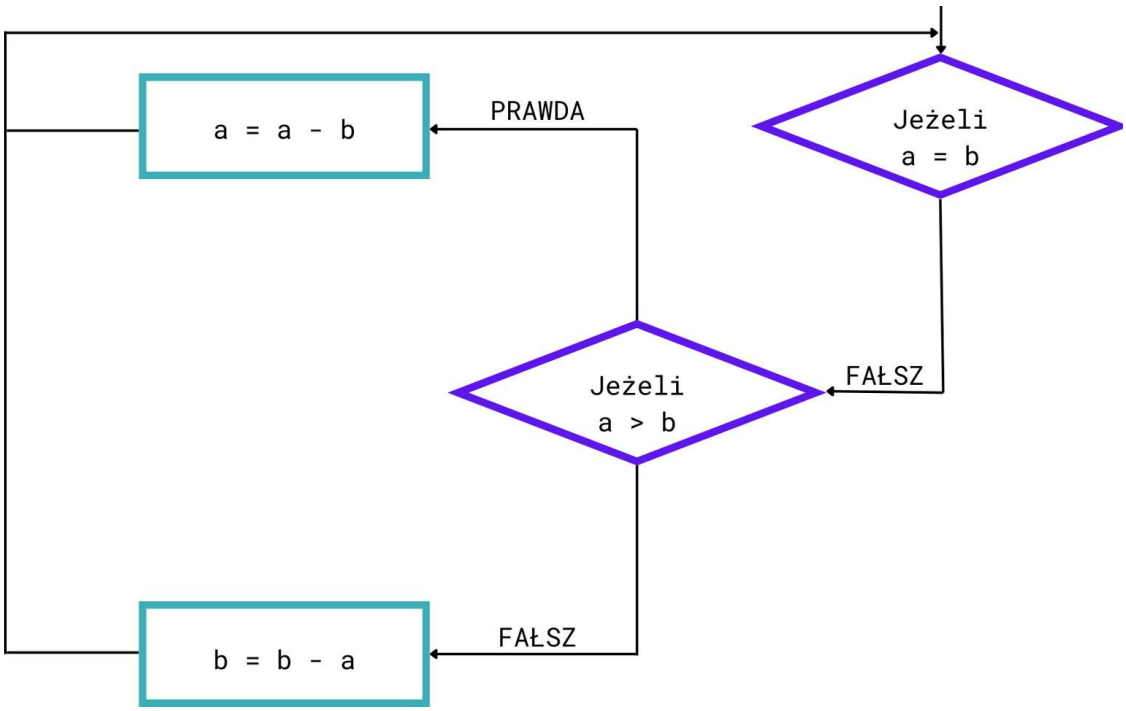
Zastanów się, czy uzyskane wyniki potwierdzają, że liczba operacji wykonywanych przez algorytm Euklidesa z wykorzystaniem odejmowania zależy od różnicy danych wejściowych.

Rozstrzygnij, czy na wyniki ma wpływ ich kolejność?

Zestaw ćwiczeń interaktywnych

Ćwiczenie 1

Zaznacz poprawną odpowiedź.
Na załączonym zrzucie umieszczono fragment schematu blokowego algorytmu Euklidesa z wykorzystaniem odejmowania. Wskaż, czego użyjesz do zapisania go w programie.



- [ ] pętli `for`
- [ ] instrukcji pobierania
- [ ] instrukcji warunkowej
- [ ] pętli `while`

Ćwiczenie 2

Uporządkuj podane instrukcje wewnątrz funkcji obliczającej NWD przy użyciu algorytmu Euklidesa z wykorzystaniem odejmowania.

```
    if a > b:
```

```
        b = b - a
```

```
    else:
```

```
while a != b:
```

```
return a
```

```
        a = a - b
```


Ćwiczenie 3

Zaznacz poprawną odpowiedź.
Wyrażenie warunkowe `a != b` oznacza:

- ◯ `a jest równe b.`
- ◯ `b jest większe od a.`
- ◯ `a jest różne od b.`
- ◯ `a to nie b.`

Ćwiczenie 4

Zaznacz wszystkie poprawne odpowiedzi.
Wskaż, które z podanych operacji powtarzane są w pętli w algorytmie szukania NWD z wykorzystaniem odejmowania.

- [ ] pobieranie poprawnych danych
- [ ] sprawdzanie, która z liczb jest większa
- [ ] wypisywanie różnicy podanych liczb
- [ ] zmniejszanie wartości większej liczby o wartość mniejszej

Ćwiczenie 5

Przeanalizuj działanie algorytmu Euklidesa z wykorzystaniem odejmowania dla a = 16 i b = 24. Zapisz wartości zmiennych a i b przyjmowane w kolejnych powtórzeniach.

Ćwiczenie 6

Połącz w pary.

`pobranie danych`	`d = oblicz_nwd(a, b)`
`zwrócenie wartości`	`a = int(input("Podaj liczbę a: "))`
`wywołanie funkcji`	`return a`
`zmniejszenie wartości`	`a = a - b`

Ćwiczenie 7

Zaznacz wszystkie poprawne odpowiedzi.
Liczba wykonywanych operacji odejmowania podczas wyznaczania NWD algorytmem Euklidesa z wykorzystaniem odejmowania:

- [ ] zależy od różnicy danych wejściowych.
- [ ] zależy od kolejności danych wejściowych.
- [ ] dla każdych danych jest inna.
- [ ] nie zależy od kolejności danych wejściowych.

Ćwiczenie 8

Wykonaj kolejne ćwiczenia.

Wskaż, jakiego typu zmiennych używamy w programie wykorzystującym algorytm Euklidesa.

- ◯ `bool`
- ◯ `string`
- ◯ `int`
- ◯ `float`

← Wstecz

Dalej →

Słownik

funkcja

wyodrębniony i nazwany blok kodu zawierający instrukcje wykonujące najczęściej jedno zadanie, w Pythonie rozpoczyna się nagłówkiem: `def nazwa_funkcji():`

parametry i argumenty funkcji

parametry to zmienne umieszczane w nawiasach okrągłych po nazwie funkcji, które symbolizują wartości wymagane do jej działania, natomiast argumenty to wartości podawane w nawiasach okrągłych po nazwie funkcji w miejscu jej wywołania

wywołanie funkcji

> uruchomienie funkcji polegające na wpisaniu jej nazwy, nawiasów i podaniu argumentów

Bibliografia

- Największy wspólny dzielnik https://encyklopedia.pwn.pl/szukaj/najwi%C4%99kszy%20wsp%C3%B3lny%20dzielnik.html [dostęp 2022-07-07].
- Sysło M.M., *Algorytmy*, Helion, Gliwice 2016.