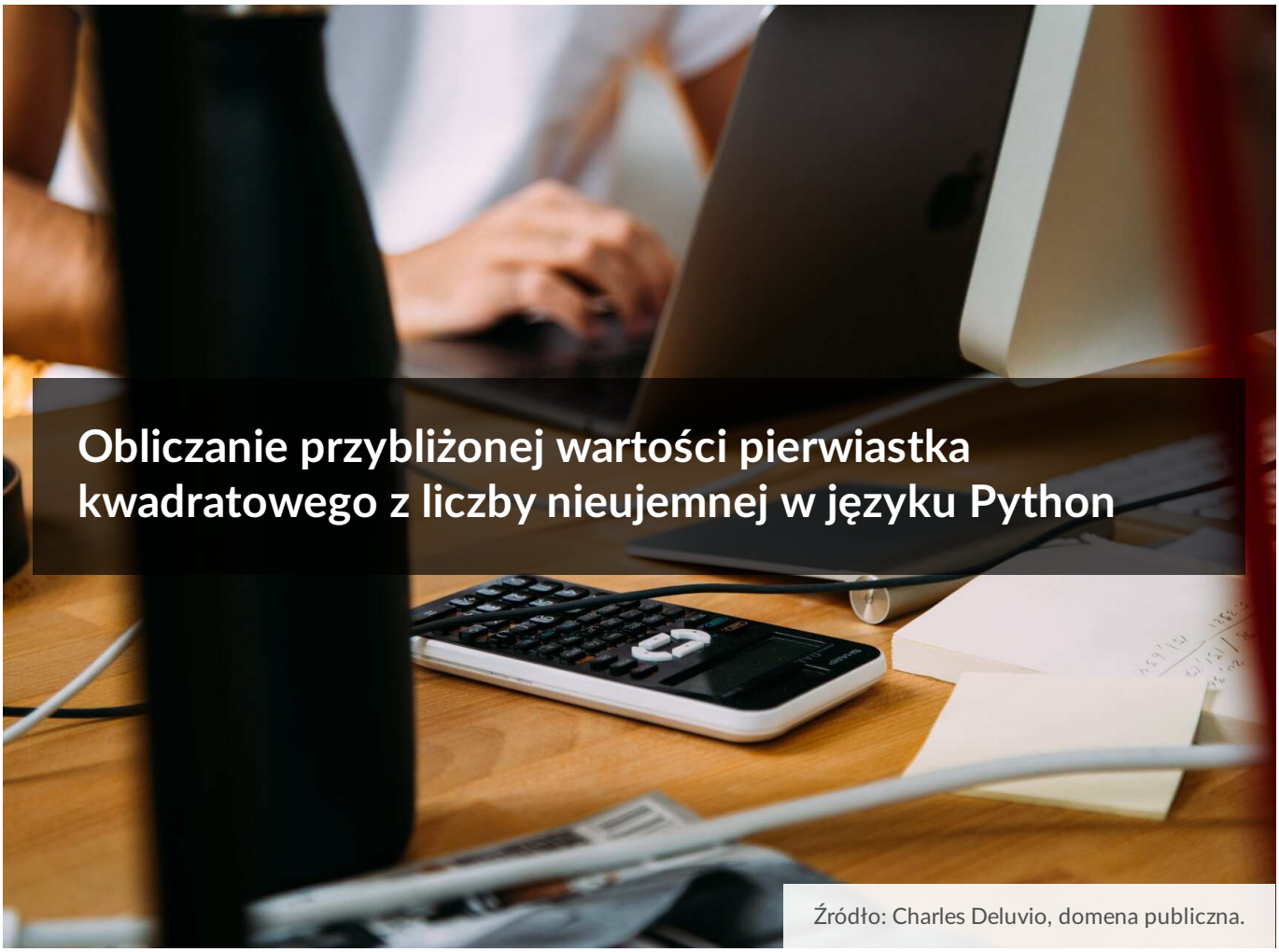
Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej w języku Python



Źródło: Charles Deluvio, domena publiczna.

Z e-materiału Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej znasz algorytm obliczania pierwiastka kwadratowego. Metoda ta pozwala wyznaczyć przybliżoną wartość pierwiastka kwadratowego z nieujemnej liczby rzeczywistej. W tym e-materiale zaimplementujemy ten algorytm w języku Python.

Implementację w innych językach programowania znajdziesz w e-materiałach:

- Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej w języku C++,
- Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej w języku Java.

Więcej zadań? Przejdź do: Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej – zadania maturalne.

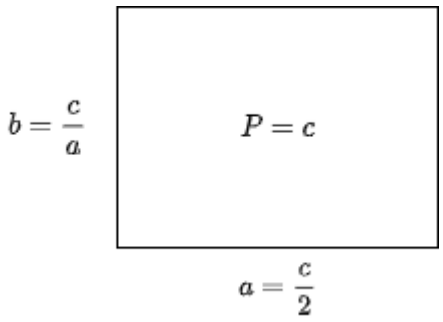
Twoje cele

- Utrwalisz założenia metody Newtona-Raphsona i sprawdzisz jej działanie w praktyce.
- Przeanalizujesz implementację algorytmu Newtona-Raphsona w języku Python.
- Zapiszesz programy wymagające obliczenia przybliżonej wartości pierwiastka kwadratowego w języku Python.

Przeczytaj

Implementacja metody Newtona-Raphsona w języku Python

W metodzie Newtona-Raphsona korzystamy z faktu, że pierwiastkowana liczba jest równa polu kwadratu, którego boki są takiej długości jak wartość wyniku pierwiastkowania. Będziemy więc tworzyć kolejne prostokąty, coraz bardziej zbliżone do poszukiwanej figury. Załóżmy, że pierwiastkujemy pewną liczbę c, a pierwszy wielokąt wyglądać będzie następująco:



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

Na podstawie długości boków prostokąta wyliczać będziemy kolejne poszukiwane wartości. Zapiszmy więc funkcję odpowiedzialną za proces wyszukiwania docelowej długości boku a kwadratu. Pamiętajmy, że jedną z niezbędnych zmiennych będzie również `precyzja`, której rolą zajmiemy się w następnym kroku. Na razie w funkcji znajdą się jedynie założenia początkowe, dotyczące długości boków pierwszego prostokąta.

```
def pierwiastkowanie(pierwiastkowany, precyzja):
    a = pierwiastkowany / 2
    b = pierwiastkowany / a
```

Tworzymy pętlę odpowiedzialną za wyliczanie kolejnych długości boków wielokątów. Operację tę przeprowadzamy do momentu, gdy wartość bezwzględna różnicy pomiędzy długościami obu boków będzie mniejsza niż zadana precyzja. Im mniejszą wartość zmiennej `precyzja` ustalimy, tym wynik będzie dokładniejszy. Długości boków prostokątów wyliczamy w następujący sposób:

$$a = \frac{(a + b)}{2}$$

$$b = \frac{c}{a}$$

Mając już niezbędne informacje, możemy przejść do dokończenia funkcji. Powtarzanie przedstawionych operacji, dopóki nie spełnimy wspomnianego warunku dotyczącego precyzji i różnicy boków, oznacza konieczność zastosowania pętli. Dodatkowo w warunku zastosujemy wartość bezwzględną. Użyjemy funkcji `abs`.

```
def pierwiastkowanie(pierwiastkowany, precyzja):
    a = pierwiastkowany / 2
    b = pierwiastkowany / a

    while abs(a - b) > precyzja:
        a = (a + b) / 2
        b = pierwiastkowany / a

    return a
```

Teraz wystarczy już tylko pobrać od użytkownika pierwiastkowaną liczbę oraz żądaną precyzję. Po wywołaniu funkcji otrzymamy wypisany przybliżony wynik pierwiastkowania kwadratowego.

```
c = float(input("Wprowadz liczbe do pierwiastkowania\n"))
p = float(input("Wprowadz precyzje\n"))

print("Wynik pierwiastkowania")
print(pierwiastkowanie(c, p))
```

Cały program prezentuje się następująco:

```
import math

def pierwiastkowanie(pierwiastkowany, precyzja):
    a = pierwiastkowany / 2
    b = pierwiastkowany / a

    while abs(a - b) > precyzja:
        a = (a + b) / 2
        b = pierwiastkowany / a
```

```
10
11      return a
12
13 c = float(input("Wprowadz liczbe do pierwiastkowania\n"))
14 p = float(input("Wprowadz precyzje\n"))
15
16 print("Wynik pierwiastkowania")
17 print(pierwiastkowanie(c, p))
```

W ten sposób zapisaliśmy algorytm Newtona-Raphsona obliczania przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej w języku Python.

Słownik

przybliżenie

przybliżeniem liczby `a` z ustaloną dokładnością `d` jest liczba, która różni się od liczby `a` o nie więcej niż `d`

metoda Newtona-Raphsona

(inaczej: metoda Newtona lub metoda stycznych) algorytm wyznaczania wartości pierwiastka funkcji oparty na wyliczaniu kolejnych przybliżeń coraz bliższych rzeczywistej wartości pierwiastka; metoda ta może być wykorzystywana między innymi do omawianego wyznaczania wartości pierwiastka kwadratowego

parametr

element składni w określonym języku programowania umożliwiający komunikację pomiędzy podprogramem wywołanym a programem wywołującym; parametry określa się wraz z deklaracją określonego podprogramu w jego nagłówku

Film samouczek

Polecenie 1

Metoda **Herona** jest graficzną interpretacją algorytmu Newtona-Raphsona.
Aby obliczyć przybliżoną wartość pierwiastka kwadratowego liczby a, stosujemy trzy kroki. Pierwszy polega na podaniu dowolnej dodatniej wartości początkowej, która powinna być jak najbliższa szukanemu pierwiastkowi (x_0).

W drugim kroku obliczamy kolejne przybliżenia według wzoru:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

Trzeci krok polega na powtarzaniu kroku drugiego aż do osiągnięcia pożądanej dokładności.

Zaimplementuj algorytm, którego celem będzie obliczenie przybliżonej wartości pierwiastka kwadratowego za pomocą metody Herona. Przetestuj działanie programu dla:

- `a = 6`
- `epsilon = 0.01`

Specyfikacja problemu:

Dane:

- a – liczba naturalna; liczba, której pierwiastka szukamy
- `epsilon` – liczba rzeczywista; dokładność przybliżenia pierwiastka

Wynik:

- `pierwiastek` – liczba rzeczywista; przybliżona wartość pierwiastka kwadratowego liczby `liczba`

Polecenie 2

Porównaj swoje rozwiązanie z przedstawionym w filmie.

Trwa wczytywanie danych ..

Film dostępny pod adresem /preview/resource/R1IvG3TwO985d

Film nawiązujący do treści materiału

Plik o rozmiarze 291.00 B w języku polskim

Kod programu zaprezentowanego w filmie:

```
if __name__ == '__main__':
    a = 6
```

```
    epsilon = 0.01
    pierwiastek = 0.0
    x = a

    while True:
        pierwiastek = 0.5*(x+(a/x))

        if abs(pierwiastek - x) < epsilon:
            break

        x = pierwiastek

    print('Pierwiastek z ' + str(a) + ' wynosi: ' + str(pierwiast
```

> **Polecenie 3**
> Zapoznajmy się z kolejnym algorytmem obliczania przybliżonej wartości pierwiastka kwadratowego. Przeanalizujemy działanie funkcji, która wyznacza pierwiastek kwadratowy podanej liczby z zastosowaniem metody równego podziału (metody bisekcji). Zwrócimy szczególną uwagę na wartości zmiennych `liczba_zgadywana`, `wynik`, `epsilon`.
>
> Metoda służy do wyznaczenia miejsca zerowego danej funkcji i polega na cyklicznym połowieniu zadanego z góry przedziału (w którym znajduje się pierwiastek) aż do osiągnięcia zadanej dokładności.
>
> Inną implementację metody równego podziału znajdziesz w e-materiale Algorytmy numeryczne i przybliżone w języku Python.

Specyfikacja problemu:

Dane:

- `liczba` – liczba naturalna; liczba, której pierwiastka kwadratowego szukamy
- `dokładność` – liczba rzeczywista; dokładność przybliżenia pierwiastka

- `pierw_l_mniejszej` – liczba naturalna; początkowe przybliżenie mniejszej wartości granicznej
- `pierw_l_wiekszej` – liczba naturalna; początkowe przybliżenie większej wartości granicznej

Wynik:

- `l_zgadywana` – liczba rzeczywista; przybliżona wartość pierwiastka kwadratowego liczby `liczba`

> **Polecenie 4**

> **Polecenie 5**

> **Polecenie 6**

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program, który za pomocą metody Newtona-Raphsona obliczy wartość pierwiastka kwadratowego liczby c przy ustalonej precyzji p. Oblicz, ile iteracji wykona algorytm, a następnie podaj wynik pierwiastkowania z dokładnością o wartości $0,01$ (bez zaokrąglania). Wypisz wyniki oddzielone pojedynczym znakiem odstępu. Swój program przetestuj dla następujących danych:

- $c = 35$
- $p = 0,0001$

Specyfikacja problemu:

Dane:

- c – liczba podpierwiastkowa; liczba rzeczywista
- p – precyzja, z jaką algorytm powinien wyznaczyć wartość pierwiastka; liczba rzeczywista

Wynik:

- `iteracje` – liczba iteracji wykonana przez algorytm; liczba naturalna
- `wynik` – wynik pierwiastkowania z dokładnością do $0,01$ bez zaokrąglania; liczba rzeczywista

Przykładowe wyjście:

```
5 5.91
```

Dla nauczyciela

Autor: Adam Jurkiewicz
Przedmiot: Informatyka

Temat: Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej 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. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

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

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:

g) obliczania przybliżonej wartości pierwiastka kwadratowego,

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):

- Utrwalisz założenia metody Newtona-Raphsona i sprawdzisz jej działanie w praktyce.
- Przeanalizujesz implementację algorytmu Newtona-Raphsona w języku Python.
- Zapiszesz programy wymagające obliczenia przybliżonej wartości pierwiastka kwadratowego w języku Python.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium 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: „Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej w języku Python". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Następnie wspólnie z uczniami ustala kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
2. **Praca z multimedium.** Nauczyciel wyświetla sekcję „Film samouczek". Uczniowie wspólnie wykonują polecenia 1 oraz 2. W kolejnym kroku uczniowie w parach zapoznają się z poleceniem 3.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1, a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.

Faza podsumowująca:

1. Sprawdzenie realizacji celów.
2. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku Python.

Praca domowa:

1. Uczniowie wykonują polecenia 4–6 z sekcji „Film samouczek".

Materiały pomocnicze:

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

Wskazówki metodyczne:

- Treści w sekcji „Film samouczek" można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.