Proste projekty i funkcje w języku Python

- Wprowadzenie
- Film samouczek
- Przeczytaj
- Sprawdź się
- Dla nauczyciela

Źródło: Free-Photos, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Z pojęciem funkcji spotkaliśmy się już na lekcjach matematyki. W materiale Proste projekty i funkcje poznaliśmy ich zastosowanie w informatyce. Funkcje w programach pozwalają podzielić kod na mniejsze części, dzięki czemu staje się on czytelniejszy, a także łatwiej znaleźć w nim błąd. Funkcje są powszechnie wykorzystywane przez programistów.

Więcej na ten temat znajdziesz w e-materiale Proste projekty i funkcje. W tym e-materiale zapoznamy się z wykorzystaniem funkcji w prostym projekcie w języku Python.

Implementację projektów w pozostałych językach programowania znajdziesz w e-materiałach:

- Proste projekty i funkcje w języku C++,
- Proste projekty i funkcje w języku Java.

Więcej zadań? Przejdź do e-materiału Proste projekty i funkcje – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz informacje dotyczące tworzenia i wywoływania funkcji w języku Python.

- Zapiszesz algorytm rozwiązywania równania kwadratowego, wykorzystując język programowania Python.
- Porównasz metodę rozwiązania równania kwadratowego za pomocą algorytmu opartego o obliczanie delty z metodą algorytmu stabilnego, wykorzystującą wzory Viète'a.

Film samouczek

Polecenie 1
Dane są wartości *a*, *b*, *c,* będące współczynnikami równania kwadratowego $ax^2 + bx + c = 0$. Napisz program, który wypisze rozwiązanie tego równania.

Zakładamy, że $a \neq 0$. Aby uniknąć błędów zaokrągleń, zastosuj algorytm stabilny.

Specyfikacja problemu:

Dane:

- a, b, c – współczynniki równania; liczby rzeczywiste

Wynik:

Program wypisuje rozwiązania równania kwadratowego.

Jeśli równanie nie ma rozwiązania, program powinien wypisać komunikat:
`brak rozwiązania`.

Polecenie 2
Porównaj swoje rozwiązanie z filmem.

Trwa wczytywanie danych ..

Film dostępny pod adresem /preview/resource/RFxsBV1CyM1sj

Nagranie filmowe dotyczące stabilnego algorytmu rozwiązywania równania kwadratowego.

Polecenie 3

Przeczytaj

Rozwiązywanie równania kwadratowego – implementacja w języku Python

Przykład 1

Zapoznajmy się z programem obliczającym rozwiązanie równania kwadratowego.

```
def rownanie_kwadratowe_delta(a, b, c):
    from math import sqrt

    if a == 0:
        print("Nie jest to równanie kwadratowe, tylko liniowe,
    else:
        delta = b ** 2 - (4 * a * c)
        if delta > 0:
            d = sqrt(delta)
            x1 = (-b + d) / (2 * a)
            x2 = (-b - d) / (2 * a)
            print("Równanie ma dwa rozwiązania: x1 =",x1," oraz
        elif delta == 0:
            x1 = -b / (2 * a)
            x2 = x1
            print("Równanie ma jedno rozwiązanie: x =",x1)
        else:
            print("Równanie nie ma rozwiązań")

# przykładowe wykonanie
rownanie_kwadratowe_delta(0.0000001, 1, -0.0000001)
```

Wynik działania programu:

```
Równanie ma dwa rozwiązania: x1 = 9.992007221626409e-08  oraz x
```

Jest to algorytm niestabilny.

W niektórych przypadkach skuteczność i dokładność takiego sposobu obliczania można zakwestionować. Jeżeli wartość iloczynu `4ac` jest mała w stosunku do wielkości `b ** 2`, to `delta` dąży do kwadratu z wartości `b`, a więc wyrażenie:

$$x_1 = \frac{-b \pm \sqrt{\Delta}}{2a}$$

dąży do zera (w zależności od wartości `b`).

Aby zapobiec błędom niedokładności obliczeń, stosuje się tak zwany **algorytm stabilny**, który opiera się na wzorach Viète'a.

> **Przykład 2**
>
> Przygotujmy kod, który pozwoli zobaczyć różnicę rozwiązań dla algorytmu stabilnego i niestabilnego. Użyjemy konwencji f-string, aby podać wynik z dokładnością do 25 miejsc po przecinku, wówczas zobaczymy różnicę w wynikach obliczeń. Różnica między wynikami algorytmów pojawia się dopiero na **17. miejscu po przecinku**.

```
def rownanie_niestabilne_test(a, b, c):
    from math import sqrt

    if a == 0:
        print("Nie jest to równanie kwadratowe, tylko liniowe,
    else:
        delta = b ** 2 - (4 * a * c)
        if delta > 0:
            d = sqrt(delta)
            x1 = (-b - d) / (2 * a)
            x2 = (-b + d) / (2 * a)

            print(f"x1 = {x1:.25f}")
            print(f"x2 = {x2:.25f}")


        elif delta == 0:
            x1 = -b / (2 * a)
            x2 = x1
            print(f"x = {x1:.25f}")
        else:
            print("Równanie nie ma rozwiązań")
```

```python

def rownanie_kwadratowe_stabilne_test(a, b, c):
    from math import sqrt
    if a == 0:
        print("Nie jest to równanie kwadratowe, tylko liniowe,
    else:
        delta = b ** 2 - (4 * a * c)
        if delta > 0:
            d = sqrt(delta)

            if b < 0:
                x1 = (-b + d) / (2 * a)
                x2 = c / (a * x1)

                print(f"x1 = {x1:.25f}")
                print(f"x2 = {x2:.25f}")


            else: #b >= 0:
                x2 = (-b - d) / (2 * a)
                x1 = c / (a * x2)
                print(f"x1 = {x1:.25f}")
                print(f"x2 = {x2:.25f}")

        elif delta == 0:
            x2 = -b / (2 * a)
            x1 = x2
            print(f"x = {x1:.25f}")
        else:
            print("Równanie nie ma rozwiązań")

# przykładowe wykonanie
print("Równanie niestabilne:")
rownanie_niestabilne_test(0.001, 7845369758436, 0.0001)

print("Równanie stabilne:")
rownanie_kwadratowe_stabilne_test(0.001, 7845369758436, 0.0001)
```

Już wiesz

- Niektóre algorytmy zachowują się niestabilnie przy pewnych zestawach danych wejściowych.
- Komputery nie są w stanie dokładnie przechowywać pewnych liczb w swojej pamięci, ponieważ w komputerze liczby wymierne przechowywane są w postaci binarnej, zgodnie z formatem IEEE 754.

Słownik

f-string

sposób zapisu zmiennych w łańcuchu znaków przeznaczonym do wypisania funkcją `print()` – dokładnie opisany w dokumencie PEP 498 – *Literal String Interpolation*; zakłada format: `f"Napis, a w nim {zmienna} do wypisania"`

IEEE 754

standard zapisu zmiennoprzecinkowego, opracowany przez Williama Kahenema, definiuje 32-bitowe oraz 64-bitowe liczby zmiennoprzecinkowe

wzory Viète'a

wzory wiążące współczynniki wielomianów i ich pierwiastki; w 1591 r. Francois Viète jako pierwszy wprowadził oznaczenia literowe dla zmiennych – w trakcie wojny francusko-hiszpańskiej złamał szyfr używany przez Hiszpanów

Sprawdź się

Pokaż ćwiczenia: 

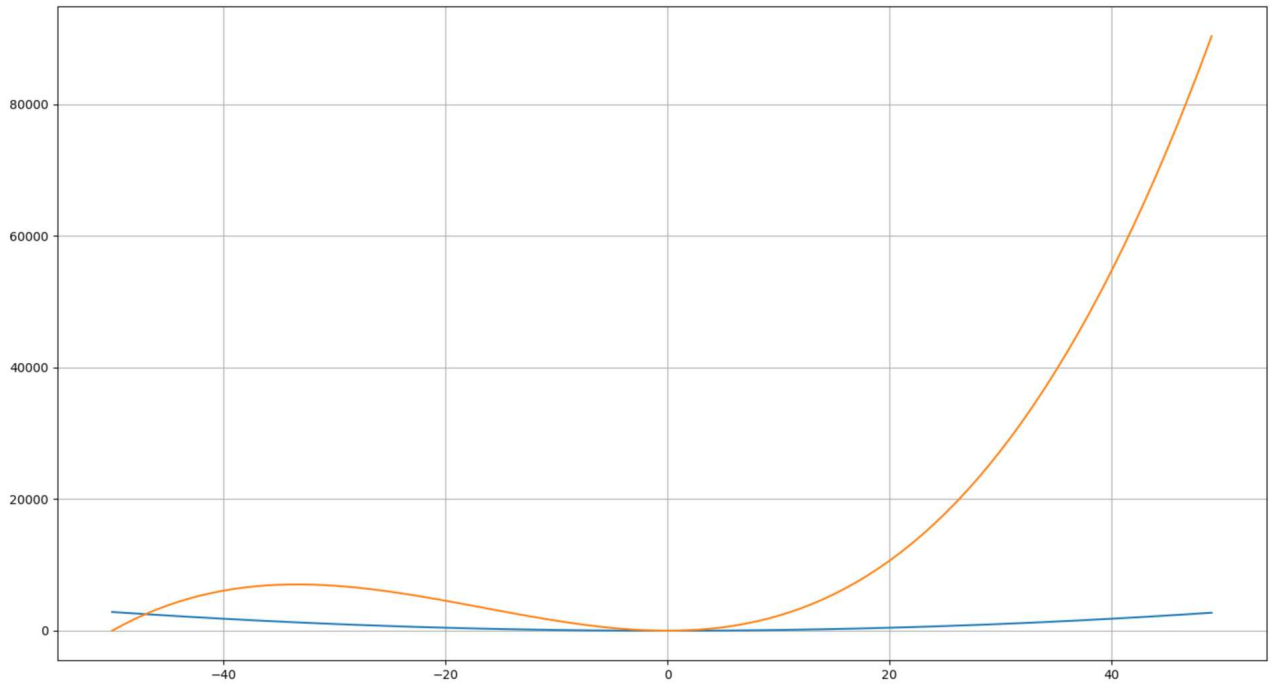
Ćwiczenie 1

W dowolnym środowisku wykonaj kod i porównaj wynik z wykresami. Następnie odpowiedz na pytanie.

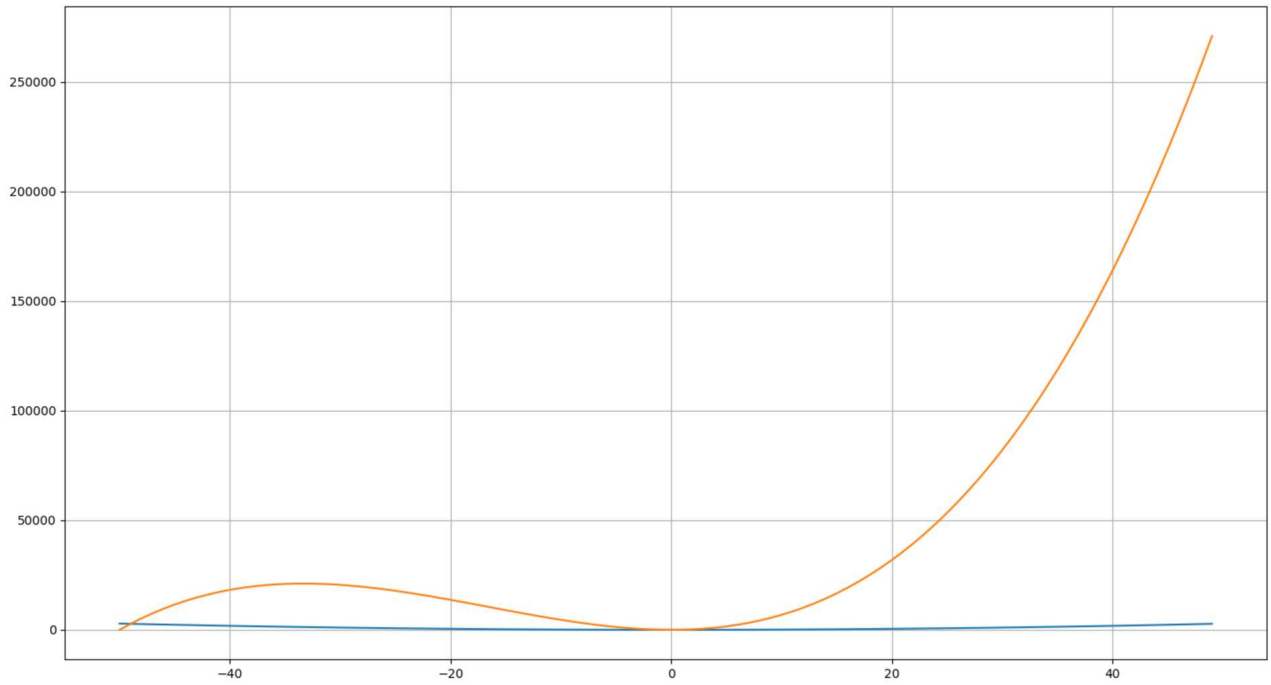
```
X = [x for x in range(-50, 50)]
A = [x * x * 1.14 for x in range(-50, 50)]
B = [i * j for i, j in enumerate(A)]

import matplotlib.pyplot as plt
plt.plot(X, A)
plt.plot(X, B)
plt.grid(True)
plt.show()
```

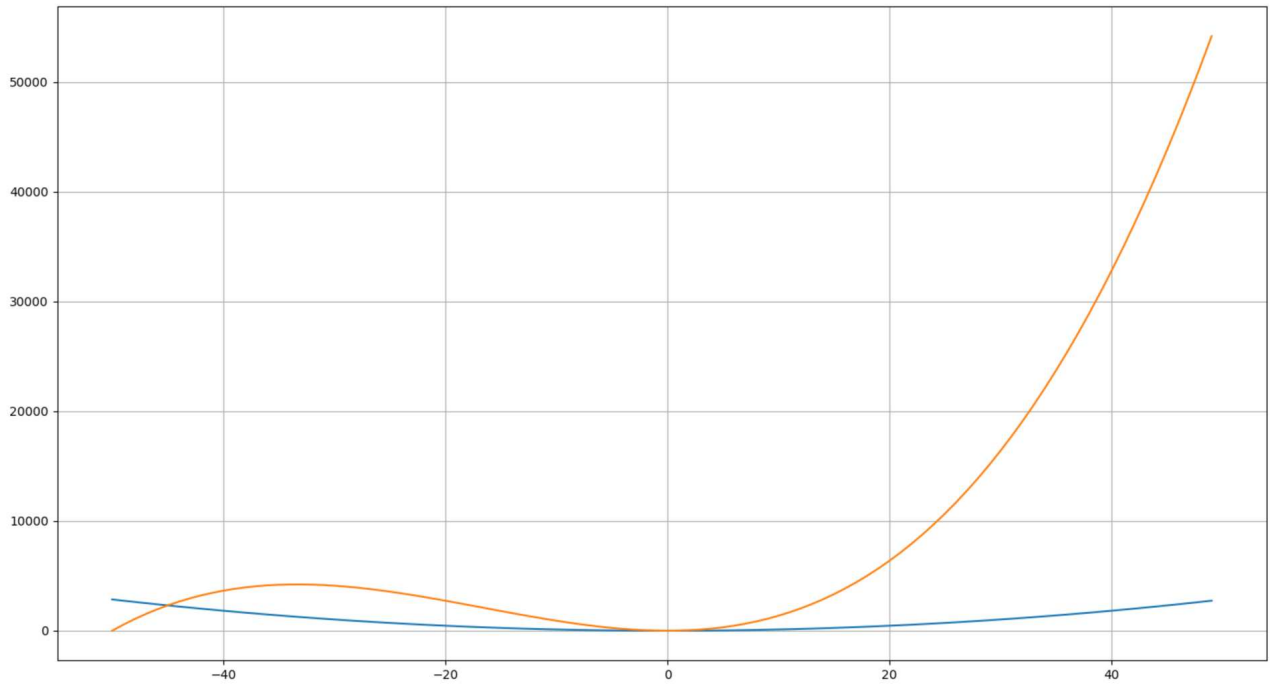
Wykres 1:



Wykres 2:


250000
200000
150000
100000
50000
0
−40
−20
0
20
40

Wykres 3:


50000
40000
30000
20000
10000
0
−40
−20
0
20
40

Ćwiczenie 2

Napisz funkcję, która zwróci wartość logiczną: `True`, gdy z odcinków o danych długościach można utworzyć trójkąt, w przeciwnym wypadku – zwróci wartość `False`. Funkcja powinna przyjmować jako parametry długości boków trójkąta oznaczone jako a, b, c. Przetestuj działanie funkcji dla dwóch zestawów danych.

```
a = 9
b = 12
c = 10
```

```
a = 11
b = 1
c = 3
```

Specyfikacja problemu:

Dane:

- *a, b, c* – liczby rzeczywiste dodatnie

Wynik:

Program wyświetla wartość `True`, jeśli z danego zestawu odcinków można utworzyć trójkąt oraz wartość `False` w przeciwnym wypadku.

Ćwiczenie 3

Program realizuje stabilny algorytm rozwiązywania równania kwadratowego. Zmodyfikuj go tak, by został rozważony przypadek, gdy $a = 0$, czyli równanie jest liniowe. Wynik wypisz zgodnie ze specyfikacją.

Najważniejsze wiadomości na temat równania liniowego znajdziesz w e-materiale Równanie liniowe.

Przetestuj jego działanie dla wartości:

```
a = 0
b = 2
c = -5
```

Specyfikacja problemu:

Dane:

- `a, b, c` – współczynniki równania kwadratowego; zmienne typu `float`

Wynik:

Program, na wyjściu standardowym, wypisze rozwiązanie równania.

Jeśli równanie jest liniowe, program wypisze jego wynik wraz z odpowiednim komunikatem:
`Ponieważ a = 0, nie jest to równanie kwadratowe tylko liniowe; rc`
albo
`Ponieważ a = 0, nie jest to równanie kwadratowe tylko liniowe; rć`
albo
`Ponieważ a = 0, nie jest to równanie kwadratowe tylko liniowe; rć`
.

Przykładowe wyjście:

```
Ponieważ a = 0, nie jest to równanie kwadratowe tylko liniowe
```

Dla nauczyciela

Autor: Adam Jurkiewicz
Przedmiot: Informatyka

Temat: Proste projekty i funkcje w języku Python

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> 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:
> >
> > > 7) wyjaśnia, jakie może być źródło błędów pojawiających się w obliczeniach komputerowych: błąd zaokrąglenia, błąd przybliżenia;
>
> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres podstawowy. Uczeń:
> >
> > > 1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

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 informacje dotyczące tworzenia i wywoływania funkcji w języku Python.
- Zapiszesz algorytm rozwiązywania równania kwadratowego, wykorzystując język programowania Python.
- Porównasz metodę rozwiązania równania kwadratowego za pomocą algorytmu opartego o obliczanie delty z metodą algorytmu stabilnego, wykorzystującą wzory Viète'a.

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: „Proste projekty i funkcje w języku Python". Nauczyciel prosi uczniów o zapoznanie się z multimedium w sekcji „Przeczytaj".

Faza wstępna:

1. Chętna lub wybrana osoba przedstawia najważniejsze informacje dotyczące funkcji w języku Python.
2. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
3. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z multimedium.** Nauczyciel czyta polecenie nr 1 z sekcji „Film samouczek", uczniowie w parach wypracowują rozwiązania, a następnie porównują je z przedstawionym w filmie.
2. **Praca z tekstem.** W razie potrzeby uczniowie przechodzą do sekcji „Przeczytaj" i na jej podstawie wprowadzają poprawki do swoich programów. Nauczyciel wyjaśnia niezrozumiałe kwestie.
3. **Ćwiczenie umiejętności.** Uczniowie, pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność pisanych kodów, porównuje je i omawia wraz z uczniami. Wskazuje najbardziej efektywne rozwiązanie.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

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

Praca domowa:

1. Uczniowie proponują alternatywny sposób rozwiązania problemów postawionych w sekcji „Film samouczek".
2. Uczniowie wykonują ćwiczenie 3 z sekcji „Sprawdź się".

Materiały pomocnicze:

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

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Film samouczek", „Przeczytaj", „Sprawdź się" jako materiał do lekcji powtórkowej.