Zastosowanie funkcji w prostych programach w języku Python

- Wprowadzenie
- Przeczytaj
- Symulacja interaktywna
- Sprawdź się
- Dla nauczyciela

Zastosowanie funkcji w prostych programach w języku Python

Źródło: Free-Photos, domena publiczna.

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

W e-materiale Zastosowanie funkcji w prostych programach przedstawiliśmy algorytm rozwiązywania układu dwóch równań, z dwiema niewiadomymi, za pomocą metody wyznaczników. Tym razem zajmiemy się jego implementacją w języku Python.

Implementację w pozostałych językach programowania znajdziesz w e-materiałach:

- Zastosowanie funkcji w prostych programach w języku C++,
- Zastosowanie funkcji w prostych programach w języku Java.

Więcej zadań? Przejdź do e-materiału Zastosowanie funkcji w prostych programach – zadania maturalne.

> **Twoje cele**
>
> - Zaimplementujesz funkcję rozwiązującą prosty układ dwóch równań metodą wyznaczników.
> - Rozwiążesz układ równań za pomocą programu w języku Python.
> - Użyjesz biblioteki `matplotlib` do graficznej prezentacji funkcji.

Przeczytaj

Rozwiązywanie układu dwóch równań liniowych w języku Python

Przykład 1

Wykorzystajmy znane już wzory do napisania w języku Python funkcji rozwiązującej układ dwóch równań z dwiema niewiadomymi metodą wyznaczników. Układ równań, który będziemy rozwiązywać, wygląda następująco:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Specyfikacja:

Dane:

- `a1` – współczynnik liczbowy a_1 pierwszego równania; liczba rzeczywista
- `b1` – współczynnik liczbowy b_1 pierwszego równania; liczba rzeczywista
- `c1` – współczynnik liczbowy c_1 pierwszego równania; liczba rzeczywista
- `a2` – współczynnik liczbowy a_2 drugiego równania; liczba rzeczywista
- `b2` – współczynnik liczbowy b_2 drugiego równania; liczba rzeczywista
- `c2` – współczynnik liczbowy c_2 drugiego równania; liczba rzeczywista

Wynik:

Program, na standardowym wyjściu, drukuje informacje na temat uzyskanego rozwiązania układu równań (w postaci komunikatu przedstawiającego kolejno: wartość niewiadomej `x` i wartość niewiadomej `y`) lub komunikat „Układ jest sprzeczny” w przypadku sprzeczności układu bądź „Układ ma nieskończenie wiele rozwiązań” w przypadku nieskończonej liczby rozwiązań.

Gotowa funkcja:

```
def rozwiazanie_ukladu_rownan_metoda_wyznacznikow(a1, b1, c1, a
    w = a1*b2 - b1*a2
    wx = c1*b2 - b1*c2
    wy = a1*c2 - c1*a2

```

```
    if w:
        print("Wynik równania x to {}".format(wx/w))
        print("Wynik równania y to {}".format(wy/w))
    else:
        if wx == 0 and wy == 0:
            print("Układ ma nieskończenie wiele rozwiązań")
        else:
            print('Układ jest sprzeczny')
```

Oto dwa przykłady wykorzystania funkcji wraz z otrzymanymi rezultatami.

Pierwszy przypadek:

Wywołanie funkcji:

```
rozwiazanie_ukladu_rownan_metoda_wyznacznikow(5, -4, 8, 4, 2, 2
```

Zwrócony rezultat:

```
Wynik równania x to 4.0
Wynik równania y to 3.0
```

Drugi przypadek:

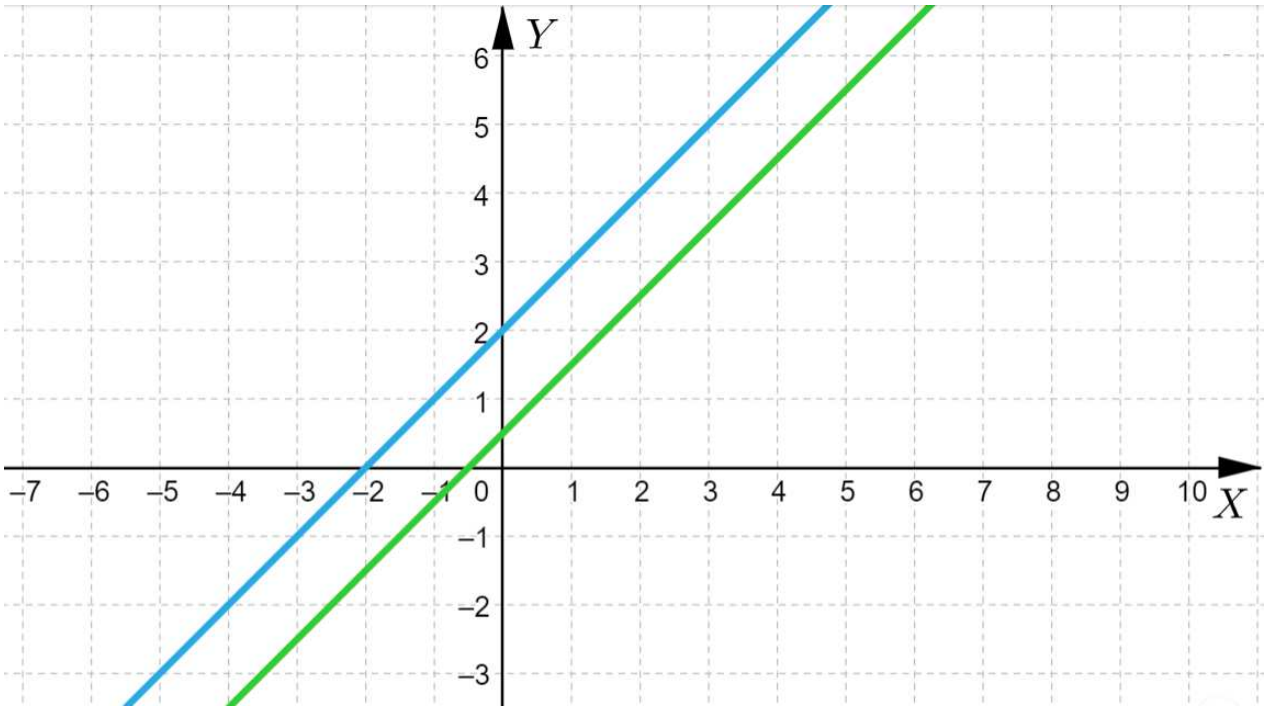
Wywołanie funkcji:

```
rozwiazanie_ukladu_rownan_metoda_wyznacznikow(1, -1, 2, 3, -3,
```

Zwrócony rezultat:

```
Układ jest sprzeczny
```

W przypadku układu sprzecznego dwie proste na wykresie są równoległe (nie mają punktu wspólnego), a zatem nie da się podać rozwiązania:

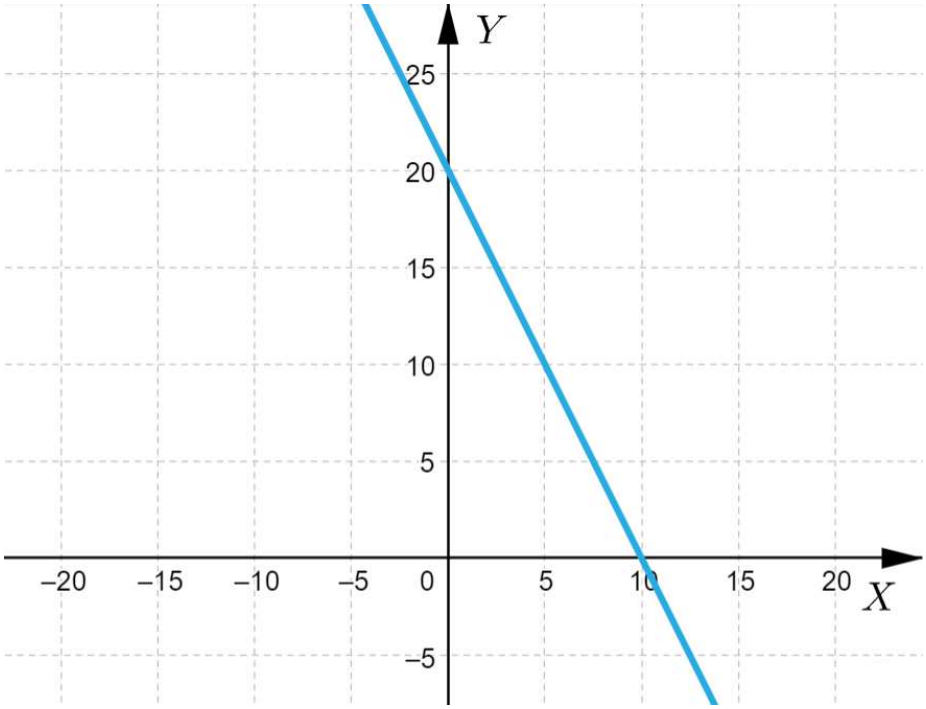


Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

W przypadku układu z nieskończoną liczbą rozwiązań, zwanym również układem tożsamościowym, rozwiązaniem są dwie linie ciągłe pokrywające się (wszystkie ich punkty są wspólne), zatem rozwiązań jest nieskończenie wiele. Oto przykład takiego układu równań:

$$x + \frac{y}{2} = 10$$

$$2x + y = 20$$



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

Przykład 2

W celu zobrazowania wyników obliczeń, posłużymy się biblioteką matplotlib i napiszemy program, który będzie generował wykres dwóch funkcji liniowych, składających się na rozwiązywany układ:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Specyfikacja:

Dane:

- `a1` – współczynnik liczbowy a_1 pierwszego równania; liczba rzeczywista
- `b1` – współczynnik liczbowy b_1 pierwszego równania; liczba rzeczywista
- `c1` – współczynnik liczbowy c_1 pierwszego równania; liczba rzeczywista
- `a2` – współczynnik liczbowy a_2 drugiego równania; liczba rzeczywista
- `b2` – współczynnik liczbowy b_2 drugiego równania; liczba rzeczywista
- `c2` – współczynnik liczbowy c_2 drugiego równania; liczba rzeczywista

Wynik

Program generuje wykres dwóch funkcji liniowych oraz zaznacza na nim uzyskane rozwiązanie układu równań. Jeżeli rozważany układ jest sprzeczny lub nieoznaczony, pomijany jest etap nanoszenia rozwiązania na wykres.

Zapiszemy najpierw równania w postaci funkcji. Posłużymy się przykładowymi równaniami, dla których przetestujemy działanie programu:

$$5x - 4y = 8 \Rightarrow f_{(x)} = 1\frac{1}{4}x - 2$$

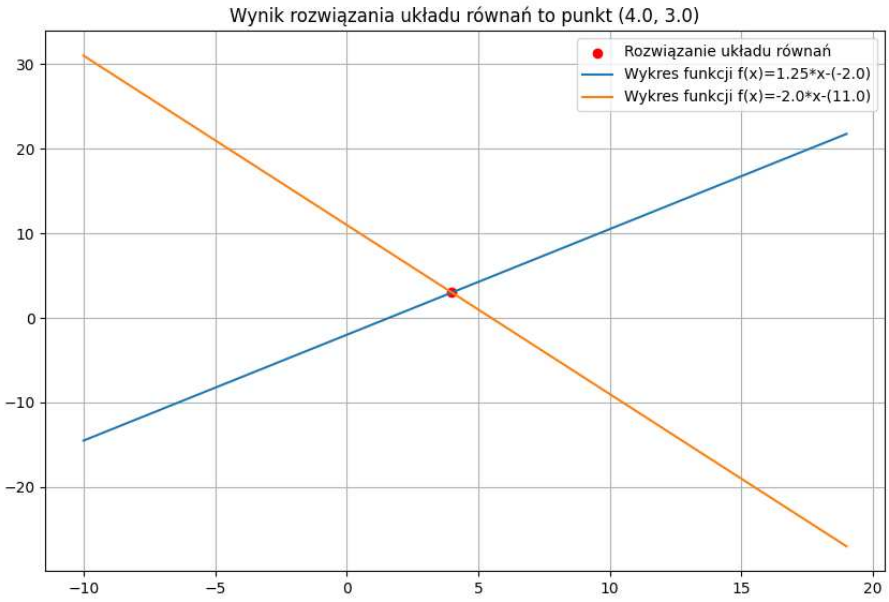
$$4x + 2y = 22 \Rightarrow f_{(x)} = -2x + 11$$

Przygotujmy kod w języku Python, który obliczy punkty niezbędne do narysowania wykresu i wyznaczy punkt przecięcia dwóch prostych.

```
1 def uklad_rownan_graficznie(a1, b1, c1, a2, b2, c2):
2     import matplotlib.pyplot as plt
3 
4     def ukl_wyzn():
```

```
        w = a1 * b2 - b1 * a2
        wx = c1 * b2 - b1 * c2
        wy = a1 * c2 - c1 * a2

        if w:
            wynik = True;
            return (wx / w, wy / w)
        else:
            wynik = False;
            return None

    def f1(x):
        return (a1 * x - c1)/(-b1)

    def f2(x):
        return (a2 * x - c2)/(-b2)

    wynik = ukl_wyzn()
    if wynik:
        W0 = [wynik[0]]
        W1 = [wynik[1]]
        plt.scatter(W0, W1, color='red')
        plt.title('Wynik rozwiązania układu równań to punkt ' +

    X0 = [x for x in range(-10, 20)]
    Y1 = [f1(x) for x in range(-10, 20)]
    Y2 = [f2(x) for x in range(-10, 20)]

    plt.plot(X0, Y1)
    plt.plot(X0, Y2)
    plt.grid(True)
    if wynik:
        plt.legend(['Rozwiązanie układu równań', 'Wykres funkcj
    else:
        plt.legend(['Wykres funkcji f(x)=' + str(a1 / -b1) + '*
    plt.show()


# wykonanie
uklad_rownan_graficznie(5, -4, 8, 4, 2, 22)
```

Wynikiem działania programu jest następujący wykres:



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

Ważne!

Zwróćmy uwagę, że czerwony punkt przecięcia rysowany jest tylko wtedy, gdy funkcja `ukl_wyzn(...)` zwróci wartość (wyznacznik główny jest różny od zera), czyli gdy istnieje dokładnie jedno rozwiązanie układu.

Dla zainteresowanych

Programiści mają do dyspozycji ciekawą bibliotekę, wykorzystywaną do rozwiązywania problemów algebraicznych. Jest nią SymPy (została ona w całości napisana w języku Python).

Słownik

matplotlib

biblioteka służąca do przedstawienia obrazów składających się z punktów o współrzędnych `x` oraz `y` (np. wykresów, histogramów, rozkładów itp.); moduł `matplotlib` nie jest dostępny w standardowej instalacji języka Python – należy go zainstalować, korzystając z mechanizmu `pip`

biblioteka SymPy

biblioteka Pythona o otwartym kodzie źródłowym; oferuje narzędzia do rozwiązywania szerokiej gamy problemów analitycznych i symbolicznych – https://www.sympy.org/en/index.html

układ równań

co najmniej dwa równania połączone operatorem logicznym AND (koniunkcja co najmniej dwóch równań)

Symulacja interaktywna

Polecenie 1

Zapoznaj się z symulacją. Samodzielnie zmieniaj wartości współczynników i obserwuj przebieg prostych oraz ewentualny punkt przecięcia. Zastanów się, jak mógłby wyglądać program sprawdzający, czy dwa wykresy funkcji mają punkt przecięcia.

Geogebra – Interaktywna symulacja

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

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi symulacji.

Polecenie 3

W dostępnych źródłach poszukaj informacji, jak obliczyć punkt przecięcia dwóch wykresów funkcji. Zapisz algorytm obliczania punktu przecięcia dwóch wykresów funkcji, używając listy kroków.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż poprawny zapis.

```
# aby uzyskać wyjściowo następującą listę
[-1.4, 1.6, 10.6, 25.6, 46.6, 73.6, 106.6, 145.6, 190.6,
241.6, 298.6, 361.6, 430.6, 505.6, 586.6]
#należy wykonać polecenie
```

- ◯

```
def f(z):
    return z*3 - 1.4

def g(z):
    return z*z

print([ fg(z)) for z in range(15) ])
```

- ◯

```
def f(z):
    return z*3 - 1.4

def g(z):
    return z*z

print([ f(g(x)) for x in range(15) ])
```

- ◯

```
def f(z):
    return z*3 - 1.4

def g(z):
    return z*z

print([ f(x)*g(x) for x in range(15) ])
```

Ćwiczenie 2

Napisz funkcję, która sprawdzi, czy podana liczba całkowita jest liczbą parzystą i jednocześnie mniejszą od 100. Oba warunki powinny zostać sprawdzone w funkcji, która przyjmie jeden parametr – liczbę naturalną do sprawdzenia.

> **Ważne!**
>
> Twoim zdaniem jest wyłącznie zdefiniowanie odpowiedniej funkcji. Po uruchomieniu programu automatycznie zostanie on przetestowany dla następujących wartości: 88, 77, 102, 101.

Specyfikacja:

Dane:

- `liczba` – sprawdzana liczba; liczba całkowita

Wynik:

Funkcja zwraca wartość logiczną `True`, gdy sprawdzana liczba jest parzysta oraz mniejsza od 100. W przeciwnym wypadku (gdy liczba nie spełnia co najmniej jednego warunku), funkcja zwraca wartość logiczną `False`.

Twoje zadania

1. Zdefiniowanie funkcji `czy_parzysta_i_mniejsza_od_stu`.
2. Funkcja zwraca wartość logiczną `True` dla wartości `88`.
3. Funkcja zwraca wartość logiczną `False` dla wartości `77`.
4. Funkcja zwraca wartość logiczną `False` dla wartości `102`.
5. Funkcja zwraca wartość logiczną `False` dla wartości `101`.

```
1 def czy_parzysta_i_mniejsza_od_stu(liczba):
2     # Tu uzupełnij kod
3     pass
```

```
```

```
1
```

Ćwiczenie 3

Napisz program, który rozwiąże podany układ dwóch równań metodą wyznaczników. Wypisz wyznaczone wartości x oraz y, rozdzielając je spacją. Jeśli układ nie ma rozwiązania, wypisz „`Brak rozwiązania`”, a jeśli ma ich nieskończenie wiele, wypisz: „`Nieskończenie wiele rozwiązań`”.

Swój program przetestuj dla układu równań:

$$\begin{cases} 2x + 3y = 5 \\ 4x + 3y = 3 \end{cases}$$

Specyfikacja:

Dane:

- `uklad_rownan` – lista składająca się z dwóch list zawierających odpowiednio: współczynniki liczbowe pierwszego układu równań oraz współczynniki liczbowe drugiego układu równań (w kolejności a, b, c); definicja układu równań

Wynik:

Program na standardowym wyjściu wypisuje rozwiązanie układu równań: wartości x oraz y (rozdzielone znakiem spacji). Jeżeli układ nie ma rozwiązań, program wyświetla komunikat: „`Brak rozwiązania`”. W przypadku, gdy układ równań ma nieskończenie wiele rozwiązań, program drukuje komunikat: „`Nieskończenie wiele rozwiązań`”.

Twoje zadania

1. Program rozwiązuje układ dwóch równań liniowych metodą wyznaczników.

```
def wyznacznik(uklad, kolumna1, kolumna2):
    # Tu uzupełnij kod

# Tu uzupełnij kod

```

```
# 2.0*x + 3.0*y = 5.0
# 4.0*x + 3.0*y = 3.0
uklad_rownan = [
    [2.0, 3.0, 5.0],
    [4.0, 3.0, 3.0],
]

# Tu uzupełnij kod
```

```

```

Ćwiczenie 4

Zdefiniuj funkcję `testowa(poczatek, koniec, granica)`, tworzącą listę zawierającą kwadrat co trzeciej liczby z zadanego przedziału. Przyjmij, że jeżeli podana wartość końca przedziału jest mniejsza niż `granica`, wówczas funkcja zwraca jedynie wartość logiczną `False`.

Program, po uruchomieniu, automatycznie przetestuje się dla następujących przedziałów (wystarczy poprawnie zdefiniować funkcję) oraz wartości `granica`:

1. $\langle 2;\ 18\rangle$; `granica = 15`
2. $\langle 3;\ 27\rangle$; `granica = 20`
3. $\langle 0;\ 24\rangle$; `granica = 12`
4. $\langle 7;\ 11\rangle$; `granica = 12`

Specyfikacja:

Dane:

- `poczatek` – początek rozpatrywanego przedziału; liczba całkowita
- `koniec` – koniec rozpatrywanego przedziału; liczba całkowita
- `granica` – minimalna wartość końca przedziału, dla której funkcja zwraca listę liczb (jeżeli koniec przedziału jest mniejszy niż `granica`, wówczas funkcja zwraca wartość logiczną `False`); liczba całkowita

Wynik:

Program zwraca listę liczb całkowitych, zawierającą kwadrat co trzeciej liczby ze zdefiniowanego przedziału (zaczynając od wartości `poczatek`). Jeżeli koniec przedziału jest mniejszy niż `granica`, program zamiast listy liczb zwraca wartość logiczną `False`.

Twoje zadania

1. Zdefiniowanie funkcji `testowa(poczatek, koniec, granica)`, działającej zgodnie z poleceniem
2. Sprawdzenie, czy funkcja dla przedziału <2, 18> oraz wartości `granica = 15` zwraca listę `[4, 25, 64, 121, 196, 289]`
3. Sprawdzenie, czy funkcja dla przedziału <3, 27> oraz wartości `granica = 20` zwraca listę `[9, 36, 81, 144, 225, 324, 441, 576, 729]`
4. Sprawdzenie, czy funkcja dla przedziału <0, 24> oraz wartości `granica = 12` zwraca listę `[0, 9, 36, 81, 144, 225, 324, 441, 576]`
5. Sprawdzenie, czy funkcja dla przedziału <7, 11> oraz wartości `granica = 12` zwraca wartość logiczną `False`

```
def testowa(poczatek, koniec, granica):
    # tutaj Twój kod
    return None

# poniżej przykładowe wywołanie
```

```
```

Dla nauczyciela

Autor: Adam Jurkiewicz
Przedmiot: Informatyka

Temat: Zastosowanie funkcji w prostych programach w języku Python

Grupa docelowa:

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

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

> 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);
>
> Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
>
> 1) projektuje i tworzy rozbudowane programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur;
>
> 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

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):

- Zaimplementujesz funkcję rozwiązującą prosty układ dwóch równań metodą wyznaczników.
- Rozwiążesz układ równań za pomocą programu w języku Python.
- Użyjesz biblioteki `matplotlib` do graficznej prezentacji funkcji.

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: „Zastosowanie funkcji w prostych programach w języku Python". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel prosi wybraną osobę o odczytanie tematu lekcji, a następnie określa cele i kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel wyświetla zawartość sekcji „Przeczytaj". Na forum klasy uczniowie analizują przedstawione w niej rozwiązania Przykładów 1 i 2.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Symulacja interaktywna". Uczniowie wspólnie zapoznają się z treścią zawartego w niej multimedium. Zapisują ewentualne problemy i pytania, po czym następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe treści.
3. **Ćwiczenie umiejętności.** Prowadzący zapowiada uczniom, że w kolejnym kroku będą rozwiązywać ćwiczenia nr 1–3 z sekcji „Sprawdź się". Każdy z uczniów robi to samodzielnie. Po ustalonym czasie wybrani uczniowie przedstawiają rozwiązania. Nauczyciel w razie potrzeby koryguje odpowiedzi, dopowiada istotne informacje, udziela uczniom informacji zwrotnej.

Faza podsumowująca:

1. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku Python.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 4 z sekcji „Sprawdź się".

Materiały pomocnicze:

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

Wskazówki metodyczne:

- Symulację interaktywną można wykorzystać w ramach powiązań międzyprzedmiotowych z matematyką, pokazując graficzne rozwiązanie układu równań. Napisany program można wykorzystać do sprawdzenia zadań rozwiązanych na lekcji matematyki.