# Zadanie 1 - a

## Cel zadania
Poznanie typów danych w Pythonie i zrozumienie działania podstawowych operatorów arytmetycznych.

## Przebieg zadania
W interpreterze Pythona sprawdzamy typ wyniku różnych działań przy pomocy funkcji type().

## Kod programu

```
print("1. ", type(1 + 2))   # + Operator dodawania dwóch liczb całkowitych
print("2. ", type(1 + 4.5)) # + Operator dodawania liczby całkowitej i zmiennoprzecinkowej
print("3. ", type(3 / 2))   # / Operator dzielenia zwracający wynik z częścią ułamkową
print("4. ", type(4 / 2))   # / Operator dzielenia zwracający wynik z częścią ułamkową
print("5. ", type(3 // 2))  # // Operator dzielenia całkowitego (wynik bez części ułamkowej)
print("6. ", type(-3 // 2)) # // Operator dzielenia całkowitego (wynik bez części ułamkowej)
print("7. ", type(11 % 2))  # % Operator modulo – zwraca resztę z dzielenia
print("8. ", type(2 ** 10)) # ** Operator potęgowania (2 do potęgi 10)
print("9. ", type(8 ** (1/3))) # ** Operator potęgowania, / Operator dzielenia; nawiasy określają kolejność działań
```

## Wynik programu

```
1.  <class 'int'>
2.  <class 'float'>
3.  <class 'float'>
4.  <class 'float'>
5.  <class 'int'>
6.  <class 'int'>
7.  <class 'int'>
8.  <class 'int'>
9.  <class 'float'>
PS C:\Users\patry>
```

## Analiza programu
1. 1 + 2 → dodawanie dwóch liczb całkowitych daje liczbę całkowitą (int).
2. 1 + 4.5 → dodanie int i float daje float.
3. 3 / 2 → dzielenie / zawsze daje float.
4. 4 / 2 → dzielenie / również daje float, nawet jeśli wynik jest całkowity.
5. 3 // 2 → dzielenie całkowite // daje liczbę całkowitą (int).
6. -3 // 2 → dzielenie całkowite zaokrągla w dół (do minus nieskończoności).
7. 11 % 2 → operator modulo % daje resztę z dzielenia.
8. 2 ** 10 → operator potęgowania ** daje int.
9. 8 ** (1/3) → pierwiastek sześcienny daje float.

## Wnioski
- Operator / zawsze zwraca float.
- Operator // zwraca int i zaokrągla w dół.
- Operator % zwraca resztę z dzielenia.
- Dodawanie int + float daje float.
- Potęgowanie ** może zwracać int lub float w zależności od wykładnika

# Zadanie 1 - b

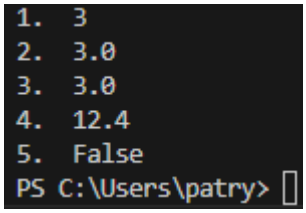
## Cel zadania
Poznanie sposobów konwersji typów danych w Pythonie.

## Przebieg zadania
Testujemy działanie funkcji int(), float(), str() i bool().

## Kod programu

```
print("1. ", int(3.0))  # Funkcja int() konwertuje liczbę zmiennoprzecinkową (float) na liczbę całkowitą (int)
print("2. ", float(3))  # Funkcja float() zamienia liczbę całkowitą (int) na zmiennoprzecinkową (float)
print("3. ", float("3.0"))  # Funkcja float() przekształca napis (string) zawierający liczbę na typ zmiennoprzecinkowy (float)
print("4. ", str(12.4))  # Funkcja str() konwertuje liczbę (tu: zmiennoprzecinkową) na tekst (string)
print("5. ", bool(0))  # Typ logiczny (True/False); 0 oznacza fałsz (False), a każda inna wartość – prawdę (True)
```

## Wynik programu




## Analiza programu
1. int(3.0) → usuwa część dziesiętną, wynik 3.
2. float(3) → zamienia int na float, wynik 3.0.
3. float("3") → konwersja tekstu na liczbę zmiennoprzecinkową.
4. str(12.4) → zamienia liczbę na tekst '12.4'.
5. bool(0) → 0 w Pythonie to False, każda niezerowa liczba to True.

## Wnioski
- Funkcje konwertujące pozwalają zmieniać typy danych w Pythonie.
- int() obcina część dziesiętną, float() dodaje ją.
- str() zamienia wszystko na tekst.
- bool() traktuje 0, pusty tekst, listy i inne puste kolekcje jako False.

# Zadanie 2

## Cel zadania
Nauka przypisywania wartości tekstowych (stringów) do zmiennych oraz wyświetlania ich w konsoli przy użyciu funkcji print().

## Przebieg zadania
Tworzymy zmienną uczelnia i przypisujemy jej tekst „Studiuję na WSIiZ”. Następnie drukujemy jej zawartość.

## Kod programu
```
uczelnia = "Studiuję na WSIiZ"

print(uczelnia)
```

## Wynik programu
```
Studiuję na WSIiZ
PS C:\Users\patry>
```

## Analiza programu
- uczelnia = "Studiuję na WSIiZ" → przypisuje tekst do zmiennej uczelnia.
- Funkcja print(uczelnia) wyświetla zawartość zmiennej w konsoli.
- Zmienna typu str przechowuje dane tekstowe.

## Wnioski
- Przypisywanie tekstu do zmiennej jest proste i umożliwia późniejsze użycie go w programie.
- Funkcja print() jest podstawowym sposobem wyświetlania informacji w Pythonie.
- Zmienne pozwalają wielokrotnie korzystać z tej samej wartości bez jej ponownego wpisywania.

# Zadanie 3

## Cel zadania
Nauka wyświetlania zmiennych w konsoli wraz z tekstem przy użyciu funkcji print().

## Przebieg zadania
Zdefiniowano zmienne imie, wiek i wzrost. Następnie wyświetlono je w konsoli, łącząc z tekstem za pomocą przecinków w funkcji print().

## Kod programu
```
imie = 'Jan'
wiek = 20
wzrost = 178

print("Nazywam się", imie, "i mam", wiek, "lat.")
print("Mój wzrost to", wzrost, "cm.")
```

## Wynik programu

```
Nazywam się Jan i mam 20 lat.
Mój wzrost to 178 cm.
PS C:\Users\patry>
```

## Analiza programu

- print() pozwala podawać wiele wartości rozdzielonych przecinkami.
- Funkcja automatycznie dodaje spacje między argumentami i konwertuje liczby na tekst.
- Dzięki temu rozwiązaniu kod jest prosty i czytelny, bez konieczności używania str() czy f-stringów.

## Wnioski

- Użycie przecinków w print() jest prostym sposobem łączenia zmiennych i tekstu.
- Jest to idealne rozwiązanie dla początkujących programistów.
- Wynik w konsoli jest czytelny i zgodny z wymaganiami zadania.

# Zadanie 4

## Cel zadania

Nauka przechowywania wartości liczbowych w zmiennych, wykonywania obliczeń matematycznych oraz wyświetlania wyników w konsoli z odpowiednim formatowaniem.

## Przebieg zadania

1. Zmienna cena przechowuje wartość produktu.
2. Zmienna rabat przechowuje wartość rabatu (0.2 = 20%).
3. Obliczamy cenę po rabacie przy pomocy prostego działania matematycznego.
4. Wynik wyświetlamy w konsoli, zaokrąglając go do dwóch miejsc po przecinku za pomocą funkcji round().

## Kod programu

```python
cena = 39.99
rabat = 0.2
nowa_cena = cena * (1 - rabat)
print("Cena po obniżce to",round(nowa_cena, 2))
```

## Wynik programu

```
Cena po obniżce to 31.99
PS C:\Users\patry>
```

## Analiza programu

- cena * (1 - rabat) oblicza nową cenę po uwzględnieniu rabatu.
- Funkcja round(nowa_cena, 2) zaokrągla wynik do dwóch miejsc po przecinku.

- Funkcja print() z przecinkami łączy tekst i wynik w prosty, czytelny sposób.

## Wnioski

- Funkcja round() jest wygodnym sposobem na formatowanie wyników liczbowych.
- Przechowywanie wartości w zmiennych pozwala na łatwe wykonywanie obliczeń.
- print() z przecinkami pozwala tworzyć przejrzyste komunikaty w konsoli bez dodatkowej konwersji typów.

# Zadanie 5

## Cel zadania

Nauka pobierania danych od użytkownika, wykonywania obliczeń matematycznych oraz wyświetlania wyników w konsoli.

## Przebieg zadania

1. Program pobiera długości boków prostokąta od użytkownika przy użyciu funkcji input().
2. Oblicza pole prostokąta (a * b).
3. Oblicza obwód prostokąta (2*a + 2*b).
4. Wyświetla wyniki w konsoli w czytelnej formie, łącząc tekst z wartościami liczbowymi.

## Kod programu

```
a = float(input("Podaj długość boku a: "))
b = float(input("Podaj długość boku b: "))

pole = a * b
obwod = (2 * a) + (2 * b)

print("Pole tego prostokąta wynosi:",pole ,"cm kwadratowych a obwód:",obwod,"cm")
```

## Wynik programu

```
Podaj długość boku a: 3
Podaj długość boku b: 8
Pole tego prostokąta wynosi: 24.0 cm kwadratowych a obwód: 22.0 cm
PS C:\Users\patry>
```

## Analiza programu

- float(input(...)) pobiera dane od użytkownika i konwertuje je na liczby zmiennoprzecinkowe.

- pole = a * b oblicza pole prostokąta.
- obwod = 2*a + 2*b oblicza obwód prostokąta.
- Funkcja print() z przecinkami łączy tekst i liczby w czytelny komunikat.

## Wnioski

- Funkcja input() umożliwia interakcję programu z użytkownikiem.
- Przechowywanie danych w zmiennych pozwala na łatwe wykonywanie obliczeń.
- Użycie przecinków w print() daje przejrzysty i czytelny sposób wyświetlania wyników w konsoli.

# Zadanie 6

## Cel zadania

- Nauka pobierania danych od użytkownika i generowania losowych wartości.
- Obliczanie przewidywanego zużycia paliwa oraz kosztów podróży.
- Wykorzystanie f-stringów do czytelnego formatowania wyników.

## Przebieg zadania

1. Program losuje długość przejechanej drogi w zakresie 1–10000 km.
2. Użytkownik podaje średnie spalanie samochodu (L/100 km) oraz cenę paliwa (zł/l).
3. Obliczane jest zużycie paliwa: spalanie = droga * (srednie_spalanie / 100).
4. Obliczany jest koszt podróży: koszt_podrozy = spalanie * cena_paliwa.
5. Wyniki wyświetlane są przy użyciu f-stringów, zaokrąglone do dwóch miejsc po przecinku.

## Kod programu

```
import random

droga = random.randint(1, 10000)
srednie_spalanie = float(input("Podaj średnie spalanie samochodu (L/100km): "))
cena_paliwa = float(input("Podaj cenę paliwa (zł/l): "))
spalanie = round(droga * (srednie_spalanie / 100),2)
koszt_podrozy = round(spalanie * cena_paliwa,2)

print(f'Samochód spali: {spalanie} litrów paliwa a koszt podróży wynosi: {koszt_podrozy} zł')
```

## Wynik programu

```
Podaj średnie spalanie samochodu (L/100km): 385
Podaj cenę paliwa (zł/l): 2.69
Samochód spali: 5478.55 litrów paliwa a koszt podróży wynosi: 14737.3 zł
PS C:\Users\patry>
```

## Analiza programu

- random.randint(1, 10000) generuje losową długość drogi.

- float(input(...)) pobiera od użytkownika wartości liczbowe.
- Obliczenia spalania i kosztu podróży są proste dzięki operacjom matematycznym w Pythonie.
- Funkcja round() zaokrągla wyniki do dwóch miejsc po przecinku.
- F-string (f'...') umożliwia wstawienie wartości zmiennych bez dodatkowej konwersji typów.

## Wnioski

- Losowanie danych i pobieranie wartości od użytkownika pozwala na tworzenie dynamicznych programów.
- F-stringi są wygodnym i czytelnym sposobem wyświetlania wyników.
- Program pokazuje praktyczne zastosowanie zmiennych, obliczeń matematycznych i formatowania wyjścia w Pythonie.

# Zadanie 7

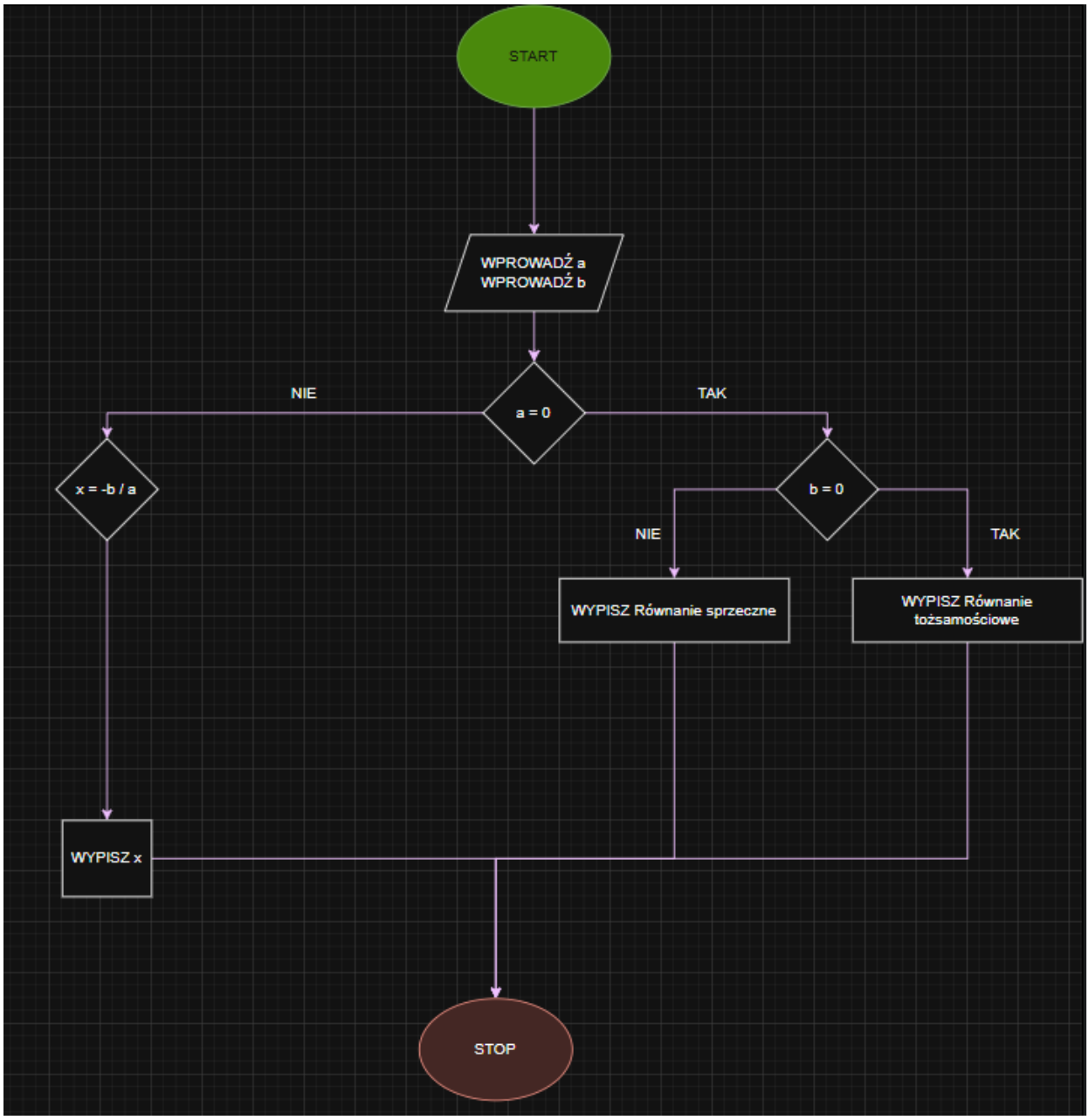
## Cel zadania

- Nauka pobierania danych od użytkownika.
- Tworzenie algorytmu rozwiązującego równanie liniowe ax + b = 0.
- Obsługa różnych przypadków równań: sprzeczne, tożsamościowe i standardowe.

## Przebieg zadania

1) Program pobiera współczynniki a i b od użytkownika.
2) Sprawdza przypadek a = 0:
    a) jeśli b = 0, równanie jest tożsamościowe (nieskończenie wiele rozwiązań),
    b) jeśli b != 0, równanie jest sprzeczne (brak rozwiązań).
3) Jeśli a != 0, oblicza rozwiązanie standardowe: x = -b / a.
4) Wynik wyświetlany jest w konsoli.

## Schemat blokowy




## Kod programu

```
a = float(input("Podaj liczbe a: "))
b = float(input("Podaj liczbe b: "))

if a == 0:
    if b == 0:
        print("Równanie tożsamościowe")
    else:
        print("Równanie sprzeczne")
else:
    x = (b * -1)/a
    print(x)
```

## Wynik programu

```
Podaj liczbe a: 4
Podaj liczbe b: 5
-1.25
PS C:\Users\patry>
```

## Analiza programu

- Program obsługuje wszystkie możliwe przypadki równania liniowego.
- Funkcja input() pobiera dane od użytkownika i konwertuje je na float.
- Instrukcje warunkowe if pozwalają na logiczne rozgałęzienie i obsługę różnych sytuacji.
- Obliczenie x = -b / a daje poprawne rozwiązanie dla równania standardowego.

## Wnioski

- Instrukcje warunkowe są kluczowe w obsłudze różnych przypadków w programach.
- Pobieranie danych od użytkownika pozwala tworzyć dynamiczne i interaktywne skrypty.
- Tworzenie schematów blokowych pomaga wizualnie zrozumieć algorytm i jego logikę.

# Zadanie 8

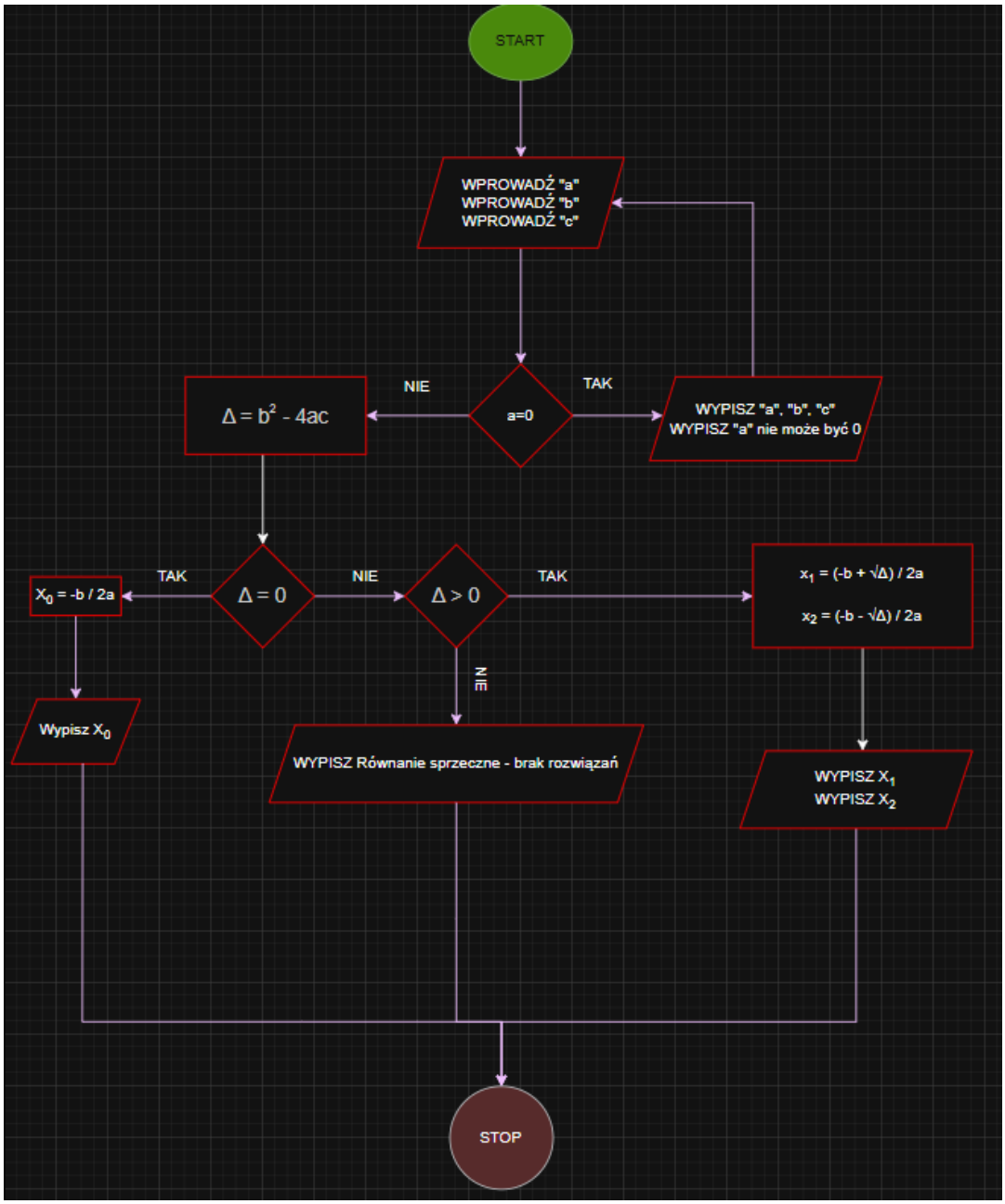
## Cel zadania

- Nauka pobierania współczynników od użytkownika.
- Obliczanie delt i rozwiązań równań kwadratowych.
- Obsługa różnych przypadków w zależności od wartości delty.
- Tworzenie schematów blokowych dla wizualizacji algorytmu.

## Przebieg zadania

1. Program pobiera współczynniki a, b, c od użytkownika.
2. Sprawdza, czy a != 0 – równanie kwadratowe wymaga, aby a było różne od zera.
3. Oblicza deltę: delta = b**2 - 4*a*c.
4. W zależności od wartości delty:
    a. delta < 0: brak rozwiązań, równanie sprzeczne,
    b. delta = 0: jedno rozwiązanie x0 = -b / (2*a),
    c. delta > 0: dwa rozwiązania x1 i x2.
5. Wyniki wyświetlane są w konsoli.

## Schemat blokowy




# Kod programu

```python
from math import sqrt

a = float(input("Podaj a: "))
b = float(input("Podaj b: "))
c = float(input("Podaj c: "))

if a == 0:
    print("a nie może być 0")
else:
    delta = (b ** 2) - (4 * a * c)
    if delta == 0:
        x_0 = (b * -1) / (2 * a)
        print(f"X0 = {x_0}")
    else:
        if delta > 0:
            pierwiastek_z_delty = sqrt(delta)
            x_1 = round(((b * -1) + pierwiastek_z_delty) / (2 * a),2)
            x_2 = round(((b * -1) - pierwiastek_z_delty) / (2 * a),2)
            print(f"x1 = {x_1} x2 = {x_2}")
        else:
            print("Równanie sprzeczne - Brak rozwiązań")
```

## Wynik programu

```
Podaj a: 10
Podaj b: 9
Podaj c: -5
x1 = 0.39 x2 = -1.29
PS C:\Users\patry>
```

## Analiza programu

- Program poprawnie obsługuje wszystkie przypadki równań kwadratowych.
- Funkcja input() pobiera dane od użytkownika i konwertuje je na float.
- Obliczenia delty i pierwiastków wykonywane są przy użyciu funkcji sqrt() i operatorów matematycznych.
- Zaokrąglanie wyników (round()) poprawia czytelność wyświetlanych rozwiązań.
- F-stringi umożliwiają łatwe i przejrzyste wyświetlanie wyników w konsoli.

## Wnioski

- Programowanie równań kwadratowych wymaga uwzględnienia różnych przypadków rozwiązań.
- Schematy blokowe pomagają w wizualizacji algorytmu i logicznego przebiegu programu.
- Zaokrąglanie i formatowanie wyników ułatwia prezentację danych użytkownikowi.

# Zadanie 9

## Cel zadania

- Nauka pobierania danych od użytkownika.

- Wykonywanie podstawowych operacji arytmetycznych: dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie i dzielenie całkowite.
- Obsługa błędów (dzielenie przez 0).

## Przebieg zadania

1. Program pobiera od użytkownika dwie liczby (pierwsza_liczba i druga_liczba).
2. Sprawdza, czy druga liczba jest różna od 0, aby uniknąć dzielenia przez 0.
3. Oblicza wyniki operacji arytmetycznych:
    a. dodawanie,
    b. odejmowanie,
    c. dzielenie z resztą,
    d. mnożenie,
    e. potęgowanie,
    f. dzielenie całkowite.
4. Wyświetla wyniki w czytelnej formie.

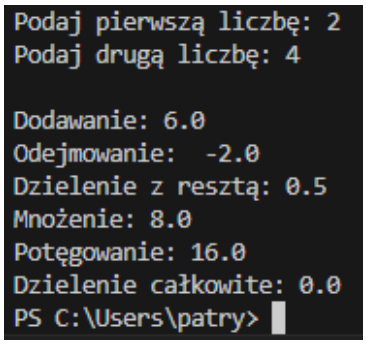
## Kod programu

```python
pierwsza_liczba = float(input("Podaj pierwszą liczbę: "))
druga_liczba = float(input("Podaj drugą liczbę: "))

if druga_liczba == 0:
    print("Nie można dzielić przez 0")
else:
    (variable) odejmowanie: float
    odejmowanie = pierwsza_liczba - druga_liczba
    dzielenie = pierwsza_liczba / druga_liczba
    mnozenie = pierwsza_liczba * druga_liczba
    potegowanie = pierwsza_liczba ** druga_liczba
    dzielenie_calkowite = pierwsza_liczba // druga_liczba

    print()
    print("Dodawanie:", dodawanie)
    print("Odejmowanie: ", odejmowanie)
    print("Dzielenie z resztą:", dzielenie)
    print("Mnożenie:", mnozenie)
    print("Potęgowanie:", potegowanie)
    print("Dzielenie całkowite:", dzielenie_calkowite)
```

## Wynik programu




# Analiza programu

- Funkcja input() pobiera liczby od użytkownika i konwertuje je na float.
- Instrukcja if zabezpiecza przed dzieleniem przez 0.
- Operacje arytmetyczne w Pythonie:
  - + dodawanie,
  - - odejmowanie,
  - / dzielenie z resztą (float),
  - * mnożenie,
  - ** potęgowanie,
  - // dzielenie całkowite.
- Funkcja print() wyświetla wyniki w czytelny sposób.

# Wnioski

- Prosty kalkulator umożliwia wykonywanie wielu operacji matematycznych na dwóch liczbach.
- Obsługa wyjątków, takich jak dzielenie przez 0, zwiększa bezpieczeństwo programu.
- Wykorzystanie zmiennych pozwala wielokrotnie używać wyników w różnych obliczeniach.