Temat: Przynależność punktu do obszaru.

1. Wprowadzenie do problemu

Celem jest sprawdzenie, czy dany punkt o współrzędnych (x, y) należy do zdefiniowanego obszaru. Obszarem może być prostokąt, koło, trójkąt lub inny kształt geometryczny. W tym przykładzie skupimy się na prostokącie oraz kole, ponieważ są to najczęściej używane figury w zadaniach tego typu.

2. Sprawdzanie przynależności do prostokąta

Prostokąt jest zdefiniowany przez współrzędne dwóch przeciwległych wierzchołków, np. $(x1, y1)$ i $(x2, y2)$.

Kod: Sprawdzanie przynależności punktu do prostokąta

```
def punkt_w_prostokacie(x, y, x1, y1, x2, y2):
    # Zakładamy, że (x1, y1) i (x2, y2) są przeciwległymi wierzchołkami
prostokąta
    return (min(x1, x2) <= x <= max(x1, x2)) and (min(y1, y2) <= y <=
max(y1, y2))

# Przykład użycia:
punkt = (3, 4)
wierzcholek1 = (1, 1)
wierzcholek2 = (5, 5)

if punkt_w_prostokacie(punkt[0], punkt[1], wierzcholek1[0],
wierzcholek1[1], wierzcholek2[0], wierzcholek2[1]):
    print("Punkt należy do prostokąta.")
else:
    print("Punkt nie należy do prostokąta.")
```

3. Sprawdzanie przynależności do koła

Koło jest zdefiniowane przez współrzędne środka (xc, yc) oraz promień r. Aby sprawdzić, czy punkt (x, y) należy do koła, obliczamy odległość punktu od środka koła i sprawdzamy, czy jest mniejsza lub równa promieniowi.

Kod: Sprawdzanie przynależności punktu do koła

```
import math

def punkt_w_kole(x, y, xc, yc, r):
    # Obliczanie odległości punktu od środka koła
    odleglosc = math.sqrt((x - xc)**2 + (y - yc)**2)
    return odleglosc <= r

# Przykład użycia:
punkt = (3, 4)
srodek_kola = (2, 3)
promien = 2

if punkt_w_kole(punkt[0], punkt[1], srodek_kola[0], srodek_kola[1],
promien):
    print("Punkt należy do koła.")
else:
    print("Punkt nie należy do koła.")
```

4. Sprawdzanie przynależności do trójkąta

Aby sprawdzić, czy punkt należy do trójkąta zdefiniowanego przez trzy wierzchołki $(x1, y1)$, $(x2, y2)$, $(x3, y3)$, możemy użyć metody opartej na powierzchniach trójkątów.

Kod: Sprawdzanie przynależności punktu do trójkąta

```
def pole_trojkata(x1, y1, x2, y2, x3, y3):
    # Obliczanie pola trójkąta za pomocą wzoru na pole powierzchni
    return abs((x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2)) / 2.0)

def punkt_w_trojkacie(x, y, x1, y1, x2, y2, x3, y3):
    # Obliczamy pole trójkąta bazowego
    pole_podstawowe = pole_trojkata(x1, y1, x2, y2, x3, y3)

    # Obliczamy pola trójkątów z punktem w środku
    pole1 = pole_trojkata(x, y, x2, y2, x3, y3)
    pole2 = pole_trojkata(x1, y1, x, y, x3, y3)
    pole3 = pole_trojkata(x1, y1, x2, y2, x, y)

    # Sprawdzamy, czy suma pól mniejszych trójkątów równa się polu
bazowego trójkąta
    return pole_podstawowe == (pole1 + pole2 + pole3)

# Przykład użycia:
punkt = (3, 4)
wierzcholek1 = (0, 0)
wierzcholek2 = (5, 0)
wierzcholek3 = (0, 5)

if punkt_w_trojkacie(punkt[0], punkt[1], wierzcholek1[0],
wierzcholek1[1], wierzcholek2[0], wierzcholek2[1], wierzcholek3[0],
wierzcholek3[1]):
    print("Punkt należy do trójkąta.")
else:
    print("Punkt nie należy do trójkąta.")
```

5. Podsumowanie

Każda funkcja została zaprojektowana tak, aby można było łatwo dostosować ją do różnych potrzeb:

- Dla prostokąta sprawdzamy, czy współrzędne punktu są w granicach.
- Dla koła sprawdzamy, czy odległość punktu od środka jest mniejsza lub równa promieniowi.
- Dla trójkąta używamy metody powierzchni, aby sprawdzić, czy punkt znajduje się wewnątrz trójkąta.

Te przykłady można rozszerzyć na inne kształty, takie jak wielokąty, używając bardziej zaawansowanych metod (np. algorytm ray-casting lub metoda wnętrza dla wielokątów wypukłych).

ZADANIA DO WYKONANIA

ZAD.1 Punkt w pierścieniu

Napisz program, który sprawdza, czy dany punkt leży w pierścieniu (obszar między dwoma współśrodkowymi okręgami). Program powinien pobierać współrzędne punktu oraz współrzędne środka pierścienia, a także promienie wewnętrzny i zewnętrzny.

Dane wejściowe:

- Współrzędne punktu (x, y).
- Współrzędne środka pierścienia (xc, yc).
- Promień wewnętrzny $r1$.
- Promień zewnętrzny $r2$ (gdzie $r2 > r1$).

Oczekiwany wynik:

- Wyświetl komunikat, czy punkt leży w pierścieniu ("Punkt jest w pierścieniu" lub "Punkt nie jest w pierścieniu").

ZAD.2 Punkt wewnątrz rombu

Napisz program, który sprawdza, czy dany punkt leży wewnątrz rombu. Rombo jest zdefiniowany przez współrzędne jego środka oraz długość przekątnych. Program powinien pobierać współrzędne punktu oraz współrzędne środka rombu i długości przekątnych.

Dane wejściowe:

- Współrzędne punktu (x, y).
- Współrzędne środka rombu (xc, yc).
- Długość pierwszej przekątnej $d1$.
- Długość drugiej przekątnej $d2$.

Oczekiwany wynik:

- Wyświetl komunikat, czy punkt leży wewnątrz rombu ("Punkt jest w rombie" lub "Punkt nie jest w rombie").