Beispiellösung: Skyline

Das Wahrzeichen von Boomtown ist die Needle, ein schlanker, 100m hoher Turm. Nach ihrem Bau wurde einst beschlossen, dass kein Gebäude in Boomtown höher als die Needle sein darf. Nun sollen viele neue Gebäude in Boomtown errichtet werden, und die Bauherren wollen höher hinaus.

Der Rat von Boomtown überlegt, dass Gebäude, die weiter von der Needle weg sind, höher sein dürfen. So können die Bauherren befriedigt werden, und gleichzeitig bleibt die Needle als markantes Wahrzeichen der Stadt erhalten. Der Rat erlässt also die Vorschrift, dass mit jeden 100m Abstand von der Needle 1m höher gebaut werden darf. Also ist die maximal erlaubte Höhe bei einem Abstand von 0m bis 99m: 100m, bei einem Abstand von 100m bis 199m: 101m usw.

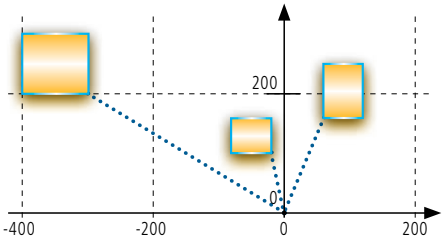
Nun soll für die vielen Bauanfragen entschieden werden, wie hoch das Gebäude jeweils sein darf. Zum Glück hatten die Stadtväter von Boomtown einst ein Koordinatensystem mit der Einheitslänge 1m eingeführt, in dem die Needle die Koordinaten (0,0) hat. Für die Grundrisse aller Gebäude gelten folgende Regeln:

> sie sind rechteckig;
> ihre Eckpunkte haben ganzzahlige Koordinaten;
> ihre Seiten liegen parallel zu den Achsen des Koordinatensystems.

Das Bild unten zeigt drei Gebäude mit den Grundrissen (angegeben durch die Eckpunkte)
(–80, 100); (–20, 100); (–20, 160); (–80, 160)
(60, 160); (120, 160); (120, 250); (60, 250)
(–400, 200); (–300, 200); (–300, 300); (–400, 300)

Aufgabe

Schreibe ein Programm, das für eine Liste von Grundrissen die erlaubten Höhen der Gebäude berechnet.
Für das Beispiel im Bild lautet die richtige Ausgabe:
101
101
103



Lösungsidee

Ein Gebäudegrundriss ist ein Rechteck. Ein Rechteck ist durch zwei diagonal gegenüberliegende Eckpunkte genau festgelegt. Wir gehen also davon aus, dass für jedes Rechteck die Koordinaten des Eckpunkts unten links (*x_ul, y_ul*) und des Eckpunkts oben rechts (*x_or, y_or*) bekannt sind. Wir nennen diese Eckpunkte auch Haupteckpunkte.

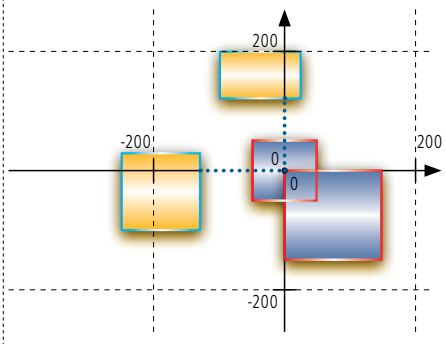
Wir suchen den Punkt auf dem Rand des Rechtecks, der dem Ursprung bzw. Nullpunkt des Koordinatensystems am nächsten liegt. Haben wir diesen Punkt gefunden, können wir den Abstand *d* des Punktes zum Nullpunkt berechnen und dann die erlaubte Höhe so ermitteln: 100 + abrunden(*d*/100).

Bei der Bestimmung des nächsten Punktes müssen wir verschiedene Fälle unterscheiden:

Fall 1: Das Rechteck liegt in genau einem Quadranten des Koordinatensystems; weder schneidet noch berührt es eine der Achsen (wie die Rechtecke im Beispiel aus der Aufgabenstellung). Dann kann der Abstand zum nächstgelegenen Eckpunkt (*x, y*) gemessen werden, und zwar mit der Formel $\sqrt{x^2+y^2}$ (vergleiche den Satz von Pythagoras).

Fall 2: Das Rechteck schneidet oder berührt eine Koordinatenachse (siehe die beiden Rechtecke links und oben im folgenden Bild). Dann muss der Abstand zum näheren Schnittpunkt mit der jeweiligen Achse berechnet werden, wie die blau gepunkteten Linien im Bild zeigen. Dafür kann folgende Formel verwendet werden („min" liefere den kleinsten von zwei Werten): min(|x_ul|, |x_or|) bzw. min(|y_ul|, |y_or|). Die beiden Rechtecke im Bild haben die Haupteckpunkte (-250, -100) und (-130, 30) bzw. (-100, 120) und (25, 200). Die Abstände sind also 130 bzw. 120, die erlaubten Höhen jeweils 101.

Fall 3: Das Rechteck schneidet oder berührt beide Koordinatenachsen (siehe die roten Rechtecke im Bild unten). Dann enthält es den Nullpunkt, evtl. als Randpunkt. In diesem Fall liegt ein Fehler vor, denn das Gebäude soll nicht auf die Needle gebaut werden.



Umsetzung

Die Lösungsidee wird in ein Programm in der Sprache Python umgesetzt. Die Unterscheidung der drei Fälle erledigt die Funktion **berechne_abstand**, die genau die vier Hauptkoordinaten x_ul, y_ul, x_or und y_or als Parameter erwartet. Der Fehlerfall 3 liegt vor, wenn x_ul <= 0 <= x_or und y_ul <= 0 <= y_or. Gilt nur eine dieser beiden Ungleichungen, dann liegt Fall 2 vor, ansonsten der „Normalfall" 1.

Das Programm erwartet die Eingabedaten in einer Datei. Diese enthält in der ersten Zeile die Anzahl der in der Datei beschriebenen Grundrisse. Jede weitere Zeile beschreibt dann einen Grundriss und enthält die Koordinaten aller seiner Eckpunkte, allerdings in vereinfachter Form: x1 y1 x2 y2 x3 y3 x4 y4

Die Funktion **hauptkoordinaten** nimmt genau so eine Zeile und gibt eine Liste mit den vier Hauptkoordinaten zurück: [x_ul, y_ul, x_or, y_or].

Die zentrale Funktion des Programms, **skyline**, liest die Zeilen der Eingabedatei nacheinander ein, besorgt sich für jede Koordinatenzeile die **hauptkoordinaten** und ruft damit **berechne_abstand** auf. Je nach Rückgabewert wird eine Fehlermeldung (Fall 3) oder die erlaubte Höhe (Fälle 1 und 2) ausgegeben.

Beispiele

Zunächst wird das Programm mit den Beispieldaten aus der Aufgabenstellung getestet. Die Eingabedatei **bwinf.in** sieht so aus:
3
-80 100 -20 100 -20 160 -80 160
60 160 120 160 120 250 60 250
-400 200 -300 200 -300 300 -400 300

Nun wird **skyline** mit dieser Datei aufgerufen:
>>> skyline('bwinf.in')
Erlaubte Höhe: 101
Erlaubte Höhe: 101
Erlaubte Höhe: 103

Ein zweiter Beispiellauf zeigt, dass das Programm auch die Fälle 2 und 3 richtig bearbeitet. Die folgende Eingabedatei **sonderfaelle.in** entspricht dem obigen Bild: zuerst kommen die beiden Fehlerfälle, dann das Rechteck über der x-Achse und zuletzt das über der y-Achse:
4
-50 -50 50 -50 50 50 -50 50
0 -150 150 -150 150 0 0 0
-250 -100 -130 -100 -130 30 -250 30
-100 120 25 120 25 200 -100 200

Als Ausgaben sind also zwei Fehlermeldungen und zweimal der Höhenwert 101 zu erwarten.

>>> skyline('sonderfaelle.in')
Fehler: Gebäude liegt über der Needle!
Fehler: Gebäude liegt über der Needle!
Erlaubte Höhe: 101
Erlaubte Höhe: 101

Quelltext

```
import math

def hauptkoordinaten(zeile):
    """Eingabe: Zeile der Eingabedatei, also: 'x1 y1 ... x4 y4'.
    Rückgabe: Koordinaten der Eckpunkte unten links und oben
    rechts als Liste [x_ul, y_ul, x_or, y_or]."""
    x_koordinaten = [] # an den Positionen 0, 2, ... der Zeile
    y_koordinaten = [] # an den Positionen 1, 3, ... der Zeile
    for i, k in enumerate(zeile.split()):
            # "enumerate" ermöglicht zwei Laufvariablen:
            # i als Zähler, k für die Listenelemente
        if i%2 == 0:
            x_koordinaten.append(int(k))
        else:
            y_koordinaten.append(int(k))
    return [min(x_koordinaten), min(y_koordinaten),
            max(x_koordinaten), max(y_koordinaten)]

def abstand(x,y):
    """Berechnet Abstand des Punktes (x,y) zum Ursprung (0,0)."""
    return math.sqrt(x*x + y*y)

def berechne_abstand(x_ul, y_ul, x_or, y_or):
    """Der kleinste Abstand eines Rechtecks zum Nullpunkt
    wird berechnet. Gegeben sind die Koordinaten der Eckpunkte
    unten links und oben rechts: x_ul, y_ul, x_or, y_or"""
    if (x_ul <= 0 <= x_or):
        # senkrechte Seiten auf beiden Seiten der y-Achse und ...
        if (y_ul <= 0 <= y_or):
            # ... waagerechte Seiten auf beiden Seiten der x-Achse:
            # Fall 3, Fehler!
            return -1 # negativer Wert als Fehlercode
        else: # Fall 2: nur senkrechte Seiten
              # auf beiden Seiten der y-Achse
            return min(abs(y_ul), abs(y_or))
    elif (y_ul <= 0 <= y_or):
        # Fall 2: nur waagerechte Seiten
        # auf beiden Seiten der x-Achse
        return min(abs(x_ul), abs(x_or))
    else: # Fall 1: Rechteck liegt komplett in einem Quadranten
        return min(abstand(x_ul,y_ul),
                   abstand(x_ul,y_or),
                   abstand(x_or,y_ul),
                   abstand(x_or,y_or))

def hoehe(distanz):
    """Berechnet die erlaubte Höhe für eine (minimale) Distanz."""
    return (100 + int(distanz/100))

def skyline(dateiname):
    """Hauptfunktion"""
    with open(dateiname) as eingabe:
        anz_gebaeude = int(eingabe.readline())
        for i in range(0, anz_gebaeude):
            # lies die nächste Zeile und berechne die Hauptkoordinaten
            geb_koord = hauptkoordinaten(eingabe.readline())
            # berechne den Abstand
            abstand = berechne_abstand(geb_koord[0], geb_koord[1],
                                       geb_koord[2], geb_koord[3])
            # Ausgabe
            if abstand < 0:
                print("Fehler: Gebäude liegt über der Needle!")
            else:
                print("Erlaubte Höhe:", hoehe(abstand))
```