Pár poznámok k štvrtému cviku:

1.) načítaniavanie zo súborov prevádzame pomocou funkcií **open()** a **read()**, netreba zabudnúť súbory zavrieť pomocou funkcie **close()**. Trik ktorým sa dá vynúť nutnosti zatvárať súbor ste viacerí použili ;) pre ostatných:

```
with open('polygon.txt', 'r') as f:
    read_data = f.read()
```

Súbor sa takto zavrie sám, viac o **with**:
https://docs.python.org/2/reference/compound_stmts.html#the-with-statement

Polygón sa načíta ako string, niektrorí ste ho parsovali, niekto využil externý modul na konverziu do JSONu, najjednoduchšie je asi použiť:

```
read_data = eval(read_data)
```

eval() vyhodnotí akýkoľvek výraz, je preto vhodný len pre spracovanie dôveryhodných zdrojov, bezpečnejšia alternatíva je **ast.literal_eval()**
https://docs.python.org/2/library/ast.html#ast.literal_eval (import ast)

2.) Pomocou objektu si de facto môžete vytvárať vlastné knihovny, triedy môžeme načítať aj z iných súborov, napr. ak máme v nejakej zložke súbor Polygon.py s týmto obsahom:

```
class Polygon:
    def __init__(self, geometrie = 0):
        self.geometrie = geometrie

    def mbr(self):
        lat=[]
        lon=[]
        for bod in self.geometrie:
            lat.append(bod[1])
            lon.append(bod[0])
        return (min(lat),min(lon),max(lat),max(lon))

    def area(self):
        a = self.mbr()
        return (a[2]-a[0])*(a[3]-a[1])
```

Môžeme si triedu Polygon importovať v inom skripte, napr. (subor so skriptom je v rovnakej zlozke ako Polygon):

```
import ast
from Polygon import Polygon

with open('polygon.txt', 'r') as f:
    read_data = f.read()

read_data = ast.literal_eval(read_data)

p = Polygon(read_data) #nova instancia triedy polygon
print p.geometrie
print p.mbr()
print p.area()
```

Pre tých čo sa trápili s objektami, treba si pamätať, že objekt je mix dát a procedúr (funkcií), funkcie voláme s použitím zátvoriek (p.geometrie vs p.mbr()). Ak už raz deklarujeme inštanciu triedy (p = Polygon(read_data)), voláme ju menom ktoré sme vybrali (p.mbr() miesto Polygon.mbr()). Snáď je to trochu jesnejšie.

3.) Lepší spôsob hľadania prieniku MBR (diki Václav)
http://gamemath.com/2011/09/detecting-whether-two-boxes-overlap/:

```
if (mbr1[3] < mbr2[1]): return False
elif (mbr1[1] > mbr2[3]): return False
elif (mbr1[2] < mbr2[0]): return False
elif (mbr1[0] > mbr2[2]): return False
else: return True;
```