

APS106 – Lab #4

Preamble

This week you will practice using conditional statements by writing a function to determine whether two rectangles overlap.

Use appropriate variable names and place comments throughout your program.

The name of the source file must be “lab4.py”.

Deliverables

For this lab, you must submit a function `rectangle_overlap` within a single file named ‘lab4.py’ to MarkUS by the posted deadline.

Five test cases are provided on MarkUs to help you prepare your solution. **Passing all these test cases does not guarantee your code is correct.** You will need to develop your own test cases to verify your solution works correctly. Your programs will be graded using ten secret test cases. These test cases will be released after the assignment deadline.

IMPORTANT:

- Do not change the file name or function names
- Do not use `input()` inside your program

Problem

While developing solutions or new technologies, engineers are faced with balancing trade-offs in their design. For example, in designing a car, reducing the size can increase fuel efficiency but result in lower crash safety ratings. Computer scientists and software engineers typically need to balance the speed of their programs with the amount of memory they require to execute. One way to approach these trade-offs is to define some constraints and then optimize the design within those constraints. In our car design example, we could define a minimum safety rating and then design the car to maximize fuel efficiency while still meeting our minimum safety rating.

In this week's lab, you will be writing a function that will check whether a design meets a particular constraint. For our problem, we will imagine we are designing the layout of a wind farm and are trying to identify where to place hundreds of turbines¹. We want to optimize turbine placement to maximize energy generation while adhering to land use constraints. These constraints define areas where turbines cannot be placed. These constraints come from land rights, proximity to other turbines, and regulations regarding turbine proximity to human housing and natural habitats.

You will write a function named `rectangle_overlap` which will analyze whether two rectangles overlap in two-dimensional space. In cases where the rectangles overlap, your function will determine the nature of the overlap (details below). In the context of our problem, these rectangles can be thought of as a restricted area and a proposed turbine site. The output of the function will tell us whether the proposed location meets the constraints.

Each of the rectangles input to the function will be defined by **two points**: the bottom left corner and the top right corner (see figure 1).

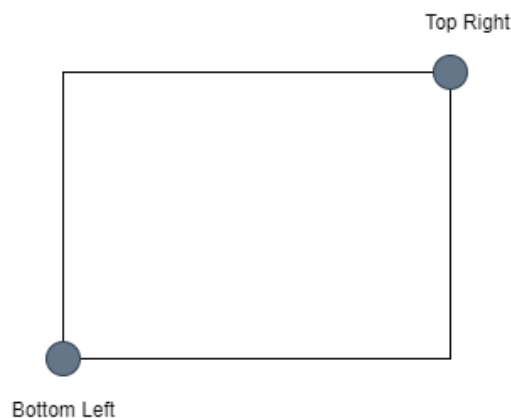


Figure 1. A rectangle can be defined with two non-adjacent corners. In this case, we are given the bottom left and top right corners. Because the angles at each corner are 90°, the other two points can be calculated using the given points.

¹ This is a real problem that has been previously investigated by Prof. Beck! See <https://www.sciencedirect.com/science/article/abs/pii/S0960148118303641?via=ihub>

The `rectangle_overlap` function will accept the following inputs:

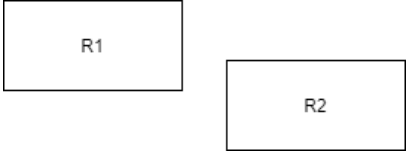

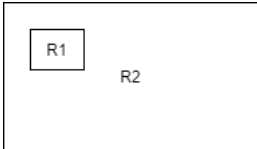
Input Parameter Name	Description
<code>rect1_bl_x</code>	x coordinate of the bottom left corner of rectangle 1
<code>rect1_bl_y</code>	y coordinate of the bottom left corner of rectangle 1
<code>rect1_tr_x</code>	x coordinate of the top right corner of rectangle 1
<code>rect1_tr_y</code>	y coordinate of the top right corner of rectangle 1
<code>rect2_bl_x</code>	x coordinate of the bottom left corner of rectangle 2
<code>rect2_bl_y</code>	y coordinate of the bottom left corner of rectangle 2
<code>rect2_tr_x</code>	x coordinate of the top right corner of rectangle 2
<code>rect2_tr_y</code>	y coordinate of the top right corner of rectangle 2

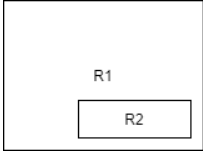
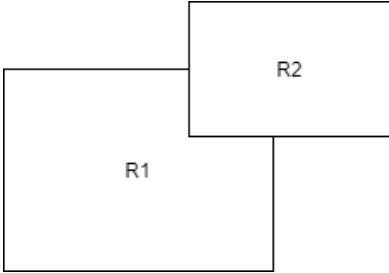
The function will output one of the five following strings describing the overlap of the two rectangles:

Function Output	Scenario
<code>"no overlap"</code>	The two rectangles do not have any overlapping area ²
<code>"identical coordinates"</code>	The two rectangles have the same set of corner coordinates
<code>"rectangle 1 is contained within rectangle 2"</code>	The entire area of rectangle 1 is contained within the area of rectangle 2 AND the two rectangles do not have the same set of corner coordinates
<code>"rectangle 2 is contained within rectangle 1"</code>	The entire area of rectangle 2 is contained within the area of rectangle 1 AND the two rectangles do not have the same set of corner coordinates
<code>"rectangles overlap"</code>	The rectangles share some overlapping area, but neither is contained completely within the other

² For this problem we will define “overlap” as a non-zero area that is within both rectangles. This means rectangles can share common borders without being considered as overlapping.

Each of these scenarios is presented below with a visual and sample inputs to help understanding. Note images are not to scale.

Scenario/Function Output	Visual Representation	Function Input Parameters
"no overlap"		rect1_bl_x: -1 rect1_bl_y: 1 rect1_tr_x: 3 rect1_tr_y: 5 rect2_bl_x: 6 rect2_bl_y: 0 rect2_tr_x: 9 rect2_tr_y: 2
"identical coordinates"	 <p>*R1 is hidden by R2, due to perfect overlap</p>	rect1_bl_x: -1 rect1_bl_y: 0 rect1_tr_x: 3 rect1_tr_y: 4 rect2_bl_x: -1 rect2_bl_y: 0 rect2_tr_x: 3 rect2_tr_y: 4
"rectangle 1 is contained within rectangle 2"		rect1_bl_x: 2 rect1_bl_y: 1 rect1_tr_x: 3 rect1_tr_y: 2 rect2_bl_x: 1 rect2_bl_y: -5 rect2_tr_x: 10 rect2_tr_y: 6

<p>"rectangle 2 is contained within rectangle 1"</p>	 <p>A diagram showing a large rectangle labeled R1. Inside R1, there is a smaller rectangle labeled R2. R2 is positioned in the lower-right area of R1, demonstrating that R2 is contained within R1.</p>	<pre> rect1_bl_x: 2 rect1_bl_y: 1 rect1_tr_x: 13 rect1_tr_y: 20 rect2_bl_x: 10 rect2_bl_y: 2 rect2_tr_x: 11 rect2_tr_y: 6 </pre>
<p>"rectangles overlap"</p>	 <p>A diagram showing two rectangles, R1 and R2, that overlap. R1 is on the left and R2 is on the right. They share a common vertical border, illustrating an overlap between the two rectangles.</p>	<pre> rect1_bl_x: 1 rect1_bl_y: 1 rect1_tr_x: 10 rect1_tr_y: 20 rect2_bl_x: 7 rect2_bl_y: 18 rect2_tr_x: 30 rect2_tr_y: 33 </pre>

Additional notes and assumptions:

- We define “overlap” as a non-zero area that is within both rectangles. This means rectangles can share common borders without being considered as overlapping.
- Rectangle coordinates will always be integers.
- Rectangle coordinates can be from any of the four quadrants of the two-dimensional plane. That is, x and y coordinates may be zero, positive, or negative.