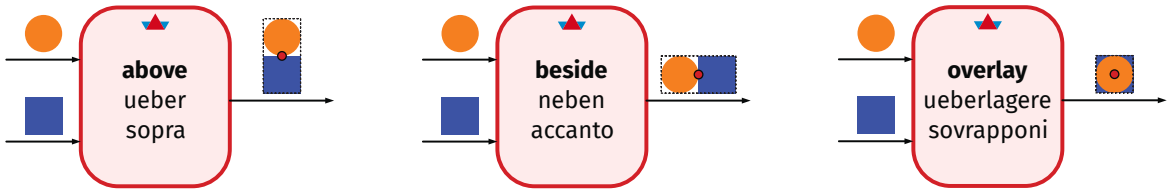
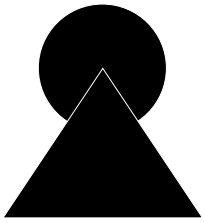
Name:	Date:

More Advanced Composition

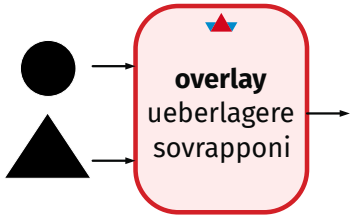
In previous lessons we learned about three functions that combine two graphics. They take two graphics as parameters and create a new one all together:

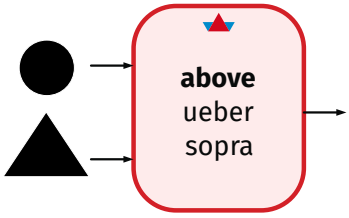


Could you use them to compose this graphic representing a keyhole?



Draw the results of the following function calls:

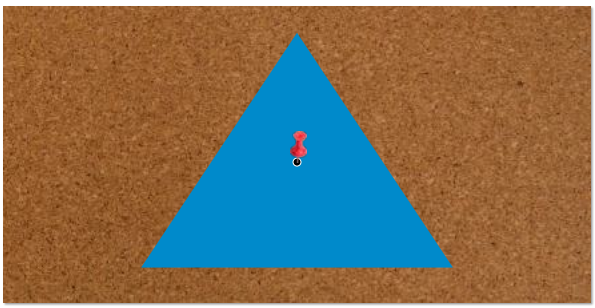




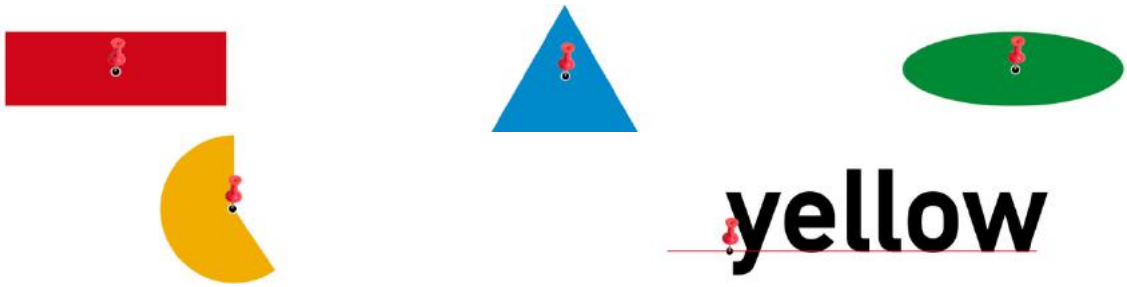
Describe: How should the triangle and circle be composed to get the keyhole?

Pin and Compose

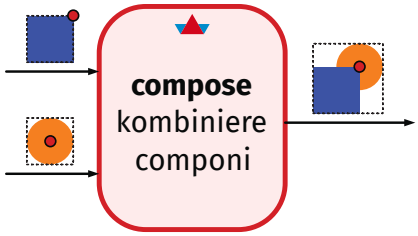
Wouldn't it be nice to be able to align graphics before composing them? With PyTamaro you can! Each graphic has a **pinning position**. You can imagine the action as fixing a thumbtack on a bulletin board:



Observe where the pinning position is in the following basic graphics (rectangle, triangle, ellipse, circular sector, or text) when you create them:



PyTamaro has a function named `compose` with two parameters:



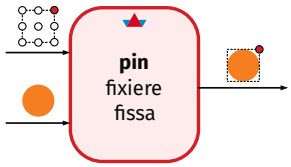
The function aligns two graphics so that their **pinning positions** overlay exactly, keeping the graphic of the first parameter in the foreground. The function produces a new composed graphic.

Draw the result of the following function calls:

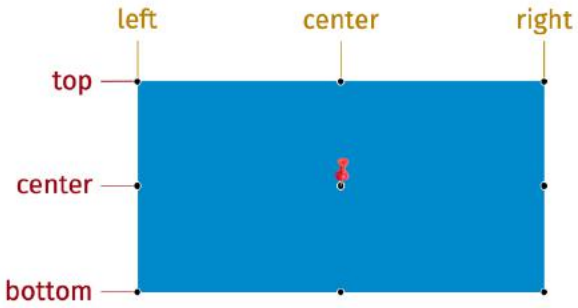
`compose(`
`  circular_sector(50, 90, blue),`
`  rectangle(100, 100, red)`
`)`	

What happens if we exchange the two arguments of the function `compose`?

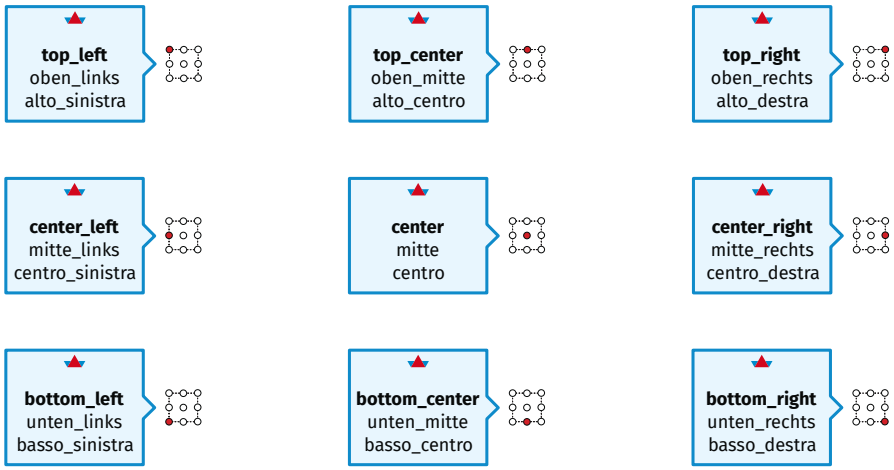
PyTamaro also has another function named `pin` that produces a new graphic with the pinning position at the specified point:



Each graphic has nine different points where you can place the thumbtack:



Here are nine constants that represent the nine points:



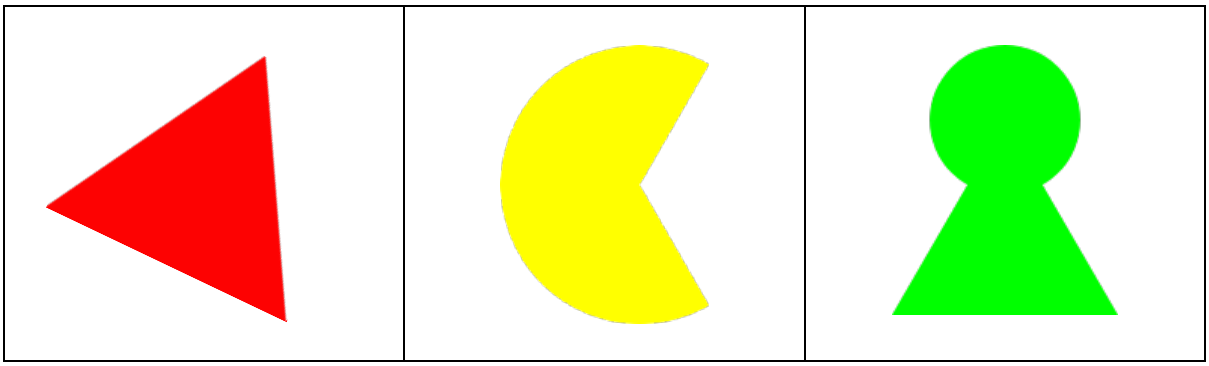
Draw the result of the following function calls (including the thumbtack):

`pin(`
`  top_center,`
`  triangle(60, 60, 60, black)`
`)`	

Tight Axis-Aligned Bounding Box

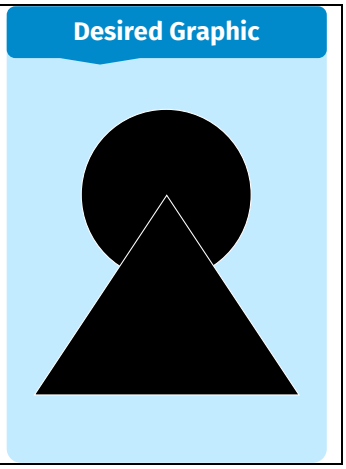
Imagine a **bounding box** that surrounds a graphic as **tightly** as possible. The box must be aligned with the axes (horizontal and vertical). On that bounding box are the nine points.

Around each graphic, draw the bounding box and the nine points:



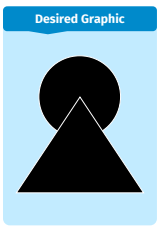
Keyhole

Use the Tamaro Cards (`ellipse`, `triangle`, `pin` and `compose`) to create the graphic of the keyhole:



How many times did you call the `pin` function? Why?

On the PyTamaro web site, go to the "*Serratura*" activity and write your program!



Program in Python:

```
show_graphic(
  compose(
```

Love!

Compose on your table a heart **using the Tamaro Cards**.
Afterwards, translate from cards to **Python code** below:

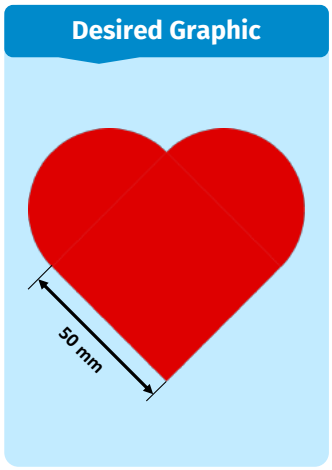
Write the expression for the constant `heart` that will then be shown.
Use the color `love_red`.

```
love_red = rgb_color(222, 0, 0)

heart =




show_graphic(heart)
```

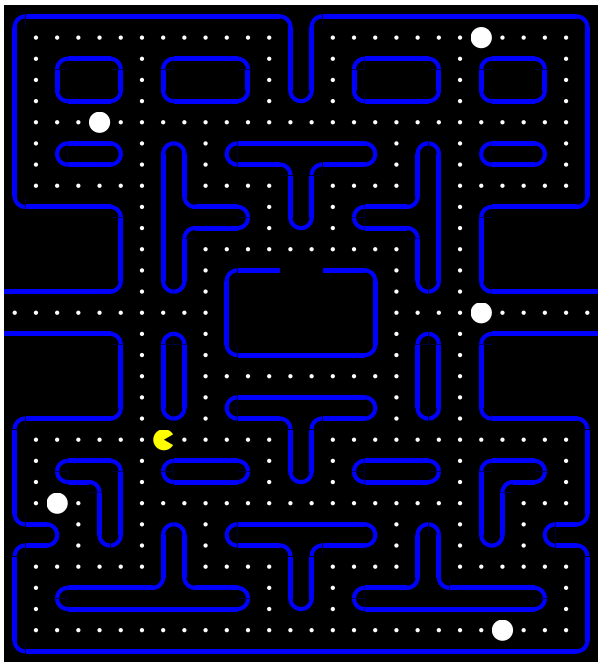


It can be helpful to first compose the graphic using Tamaro Cards:

On the PyTamaro web site, go to the "*Cuore*" activity and test whether the program you wrote works correctly!

A Tile from the Pac-Man Maze

In Unit 2 we saw hot to compose a Pac-Man maze from basic tiles:



How many different tiles did you find?

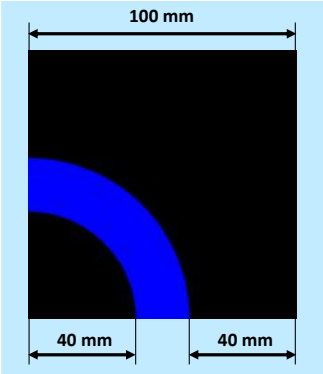
Draw all the necessary tiles:

Let's try to program the “low left corner” tile: use `pin` and `compose` to **program** this tile from the Pac-Man game.

Lower left corner with Tamaro Cards:

Lower left corner with Python code:

```
lower_left_corner =
```



Try your code on the PyTamaro web site in the "*Angolo di Pac-Man*" activity!

If you used `overlay` instead of `compose` with the two circular sectors, what would be the result?

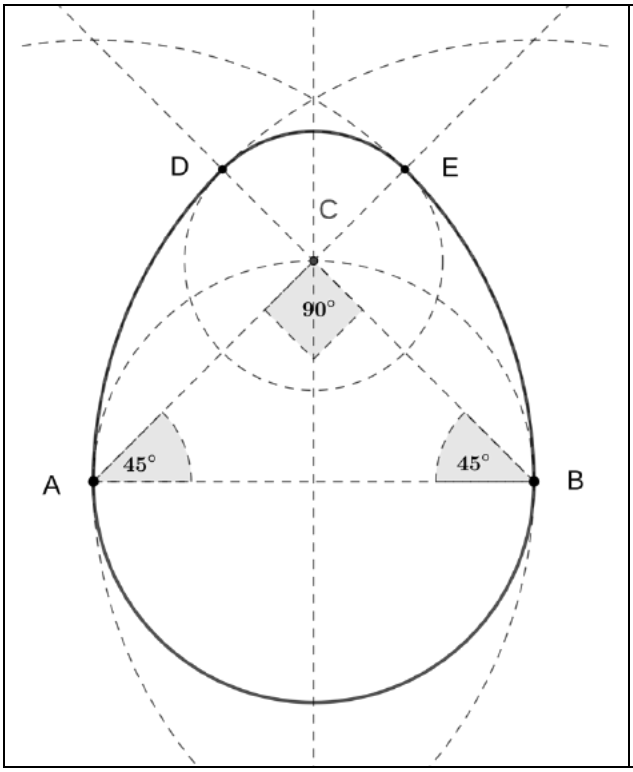
Draw the two versions:

How could you most easily create the other corner tiles, given the tile you just programmed?

Moss's Egg

Here is an egg in the shell, one shelled and one cut in half:

This geometric construction is called a "Moss's egg."



Describe in a few words how Moss's egg is composed.
All you need are circular sectors with different radii.
The circular sectors are stacked on top of each other to create the egg.

Compose the egg using Tamaro Cards first, then write its code and test if it works on the PyTamaro site in the "*Uovo di Moss*" activity.

Recipe:

- Define the color of the shell:
  ```
  shell_color = rgb_color(223, 151, 112)
  ```
- Define the necessary lengths AB, BC, and CE:
  ```
  AB = 600
  BC = AB / sqrt(2)          (from math import sqrt)
  CE = AB – BC
  ```
- Compose the egg, working with the constants:

1st stage: *2nd stage:* *3rd stage:* *egg:*

Implement your Moss's Egg project with the Tamaro Cards:

First stage:

with Tamaro Cards:

in Python code:

```
first_stage =
```

Second stage:

with Tamaro Cards:

in Python code:

```
second_stage = compose(
            pin(bottom_right, first_stage),
```

Third stage:

with Tamaro Cards:

in Python code:

```
third_stage = compose(
            pin(bottom_left, second_stage),
```

Complete egg:

with Tamaro Cards:

in Python code:

```
egg = above(
            third_stage,




show_graphic(egg)
```