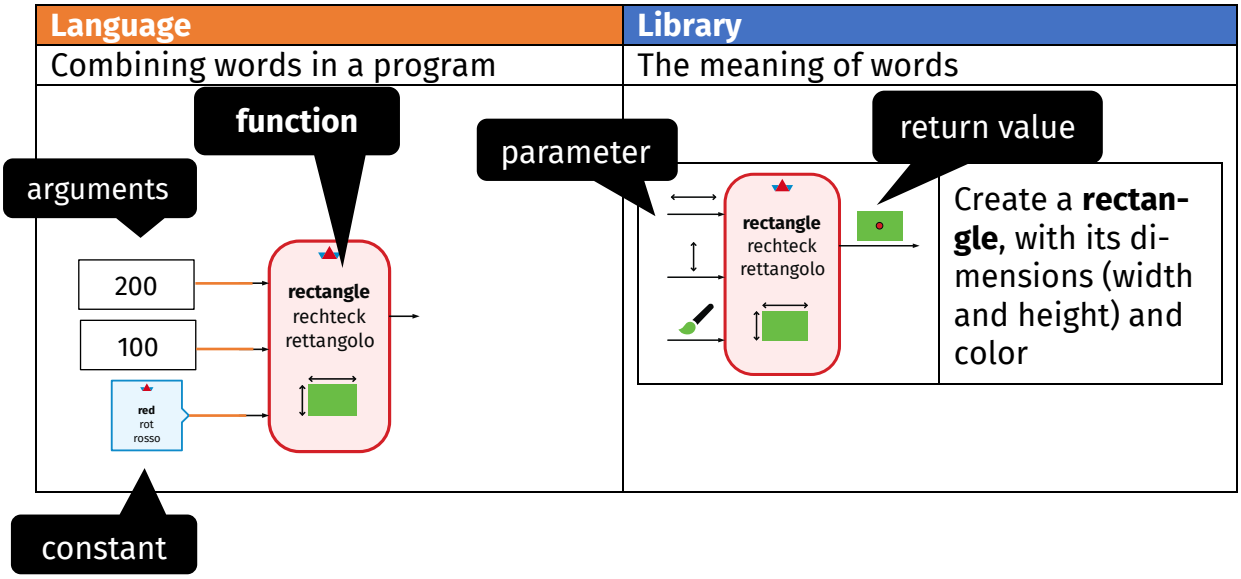
Name:	Date:

From Cards to Python

A brief review...

Tamaro Cards – Creating Graphics



Python Language & PyTamaro Library

Language – Python	**Library** – PyTamaro
Combining words in a program	The meaning of words

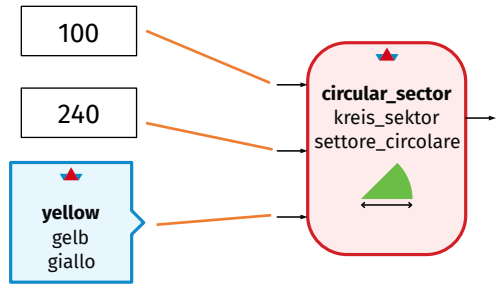
`rectangle(80, 40, red)`
function call arguments
	`rectangle(`
	`  width,`
	`  height,`
	`  color`
	`)` — Create a **rectangle**, with its dimensions (width/height) and color

On your computer, one unit is one pixel.

Your screen (on your computer, on your phone) uses thousands of pixels, tiny colored squares, to show graphics.

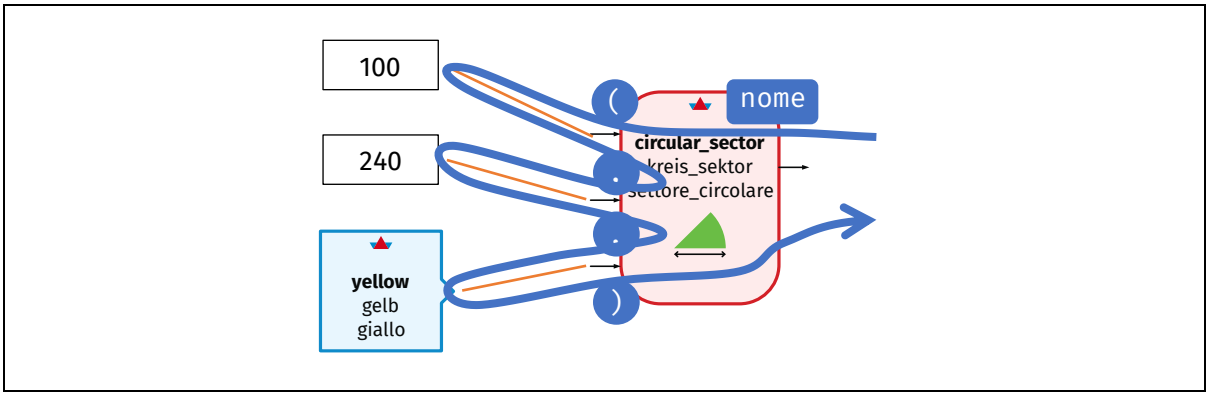
How many pixels does your screen have?

From Cards to Python Code

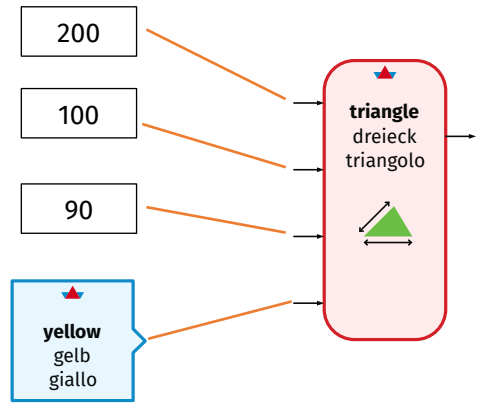
Program with Cards	Program in Python
	`circuar_sector(100, 240, yellow)`

To translate Tamaro Cards into Python code:

- Draw the "street", starting at the result (the rightmost arrow)
- Write the name of the **function** **circular_sector**
- Before passing to the first argument, open a parenthesis **(**
- Write each **argument** from top to bottom
- Separate the arguments with a comma **,**
- After the last argument, close the parenthesis **)**



Translate the following program into Python code (draw the street to do that):

Program with Cards	Program in Python
	

Run Your Python Program

To **execute** your program in Python, we need an **interpreter** that understands code written in Python. The interpreter knows the rules of the grammar (syntax) and the meaning of the various parts (semantics).

The execution of a program is a process where a system (the Python interpreter on your computer) executes, or interprets, the code of the program.

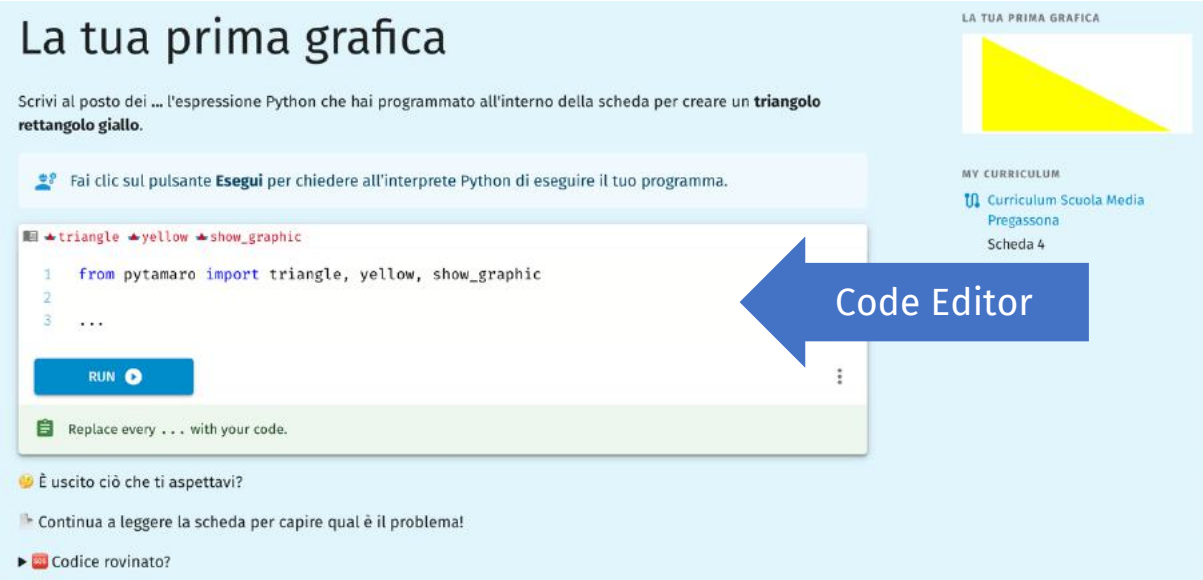
Let's get to work! With your computer, visit the **PyTamaro** web site on the following page, which contains the curriculum with the various activities we will follow:

https://pytamaro.si.usi.ch/curricula/ehinger/pregassona

Or, shorter: tinyurl.com/pytamaropregassona

Open the page, click on "Start" and open the "La tua prima grafica" activity.

Each activity contains at least one code editor:
a white area where you can write Python code.



Write instead of the ... the Python expression you programmed on the previous page to create a right-angled yellow triangle.

Click on the **"Esegui"** button to ask per the Python interpreter to execute your program.

What gets visualized?

If you haven't made errors, you should not see any response from the Python interpreter.

The interpreter really executed your code and created the right-angled triangle!
But it simply has not received any instruction to make that triangle visible!
A really fussy interpreter!

It gives you a hint to use `show_graphic`!

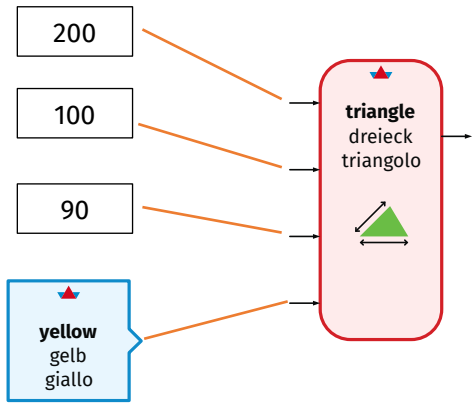
To show a graphic, we can use a new function. Here is its card:

There are **two** key differences between this card and those you have seen before. Which?

This function does not have an arrow on the right: it does not produce a return value! We can thus not "attach" it to other cards.

The symbol indicates that this function causes an **effect**.
For the function `show_graphic`, the effect is to show the graphic it receives via its parameter on the screen.

Add to the cards below the function `show_graphic` and then **translate** the cards to Python by carefully using the "street"!

Program with Cards	Program in Python
	

Write the new Python expression in the code editor instead of the old one, then try to **execute it again**.

Do you see the yellow triangle? Great! The Python interpreter successfully executed your first program!

The Python Interpreter is a Strict Teacher

If your code contains an **error** because you did not respect the rules of the Python languages, you are a lucky person! The Python interpreter will show you your error. It tells you what kind of error there is in your code, like your English teacher does in an essay or a grammar test.

```
rectangle(200 100 reed)
```

SyntaxError: invalid syntax. Perhaps you forgot a comma?

Even if English is not your native language, the Python interpreter likes to show you error messages in English.

Do you think using English is cooler? Or would you prefer your native language?

In this example Python indicates that a **syntax** error was made. One needs to write a **comma** between arguments!

Let's correct the error:

```
rectangle(200, 100, reed)
```

NameError: name 'reed' is not defined

Ops... after fixing the indicated error, we get another error message...
The interpreter does not know the definition of the **name** `reed` ...

Unlike a language teacher, the Python interpreter only shows you the **first** error made, then it stops.
You have to fix that error and then try to execute the program again.

Fix the last error and write the correct code:

What happens if we write:

```
show_graphic(rectangle(200, 100, yellow)
```

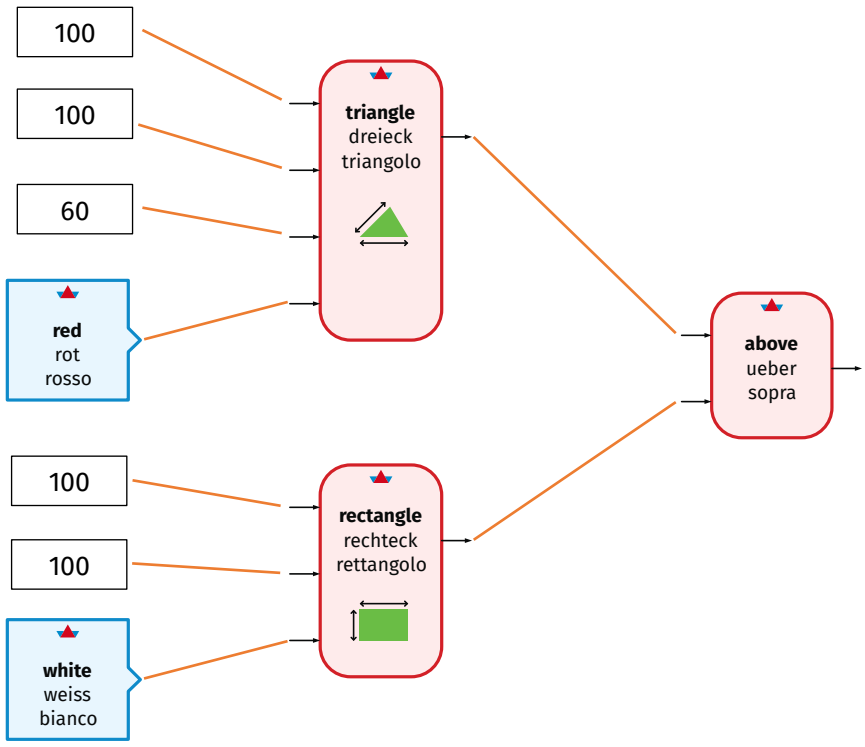
SyntaxError: '(' was never closed

Explain this error:

Program a White House and a Red Roof

Let's try to translate a more complex program... draw the "street" and translate the cards into the Python language:

Program with Cards:



Program in Python:

On the PyTamaro web site, go to the "*Casa*" activity and **replace** the `...` with the Python expression you wrote above. **Execute the program.**

The call to the function `show_graphic` is still missing! **Add** the card above and translate it to Python code using the "street" strategy:

Program in Python:

Execute the program on the platform and you should see the house!

Importing Names from a Library

We can change our house a bit. For example, we can create a ground floor of height 150 instead of 100.

Look carefully at the cards.
Which number do we have to change to grow the ground floor of the house?

Change the corresponding number in the Python program and **execute it**.
Is the result what you expected?

Let's try to change the **color** of our ground floor from white to yellow.
What do you have to change in the program?

Change the corresponding color in the Python program and **execute it**.
Is the result what you expected?

There is no magic in the world of programming. The interpreter is a "faithful" but "dumb" executor: we have to tell him really everything explicitly!

We already met the `NameError`: it happens when the interpreter does not know the meaning of a certain name. It knows names like `rectangle` and `white` only because at the beginning there is an instruction that tells it where to find their meaning:

```
from pytamaro import rectangle, white
```

This instruction tells the Python interpreter to use the names `rectangle` and `white` from the `pytamaro` library.

Attention! Once we import too many names, some of them may not be visible in the editor anymore. Don't worry: the names are still there, and you can use the horizontal scrollbar to see them. Or you can split the imports into multiple lines:

```
from pytamaro import rectangle, triangle
from pytamaro import white
```

Add to the list of these names also the name `yellow` (separated by a comma).
Execute the program. Is the result what you expected?