Turtle (Scratch with Python)

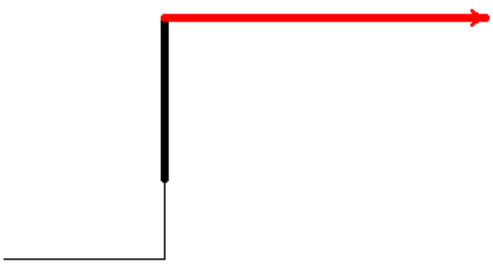
The `turtle` *module allows you to easily make drawings in* `Python`*. It's about giving orders to a turtle with simple instructions like "go ahead", "turn"... It's the same principle as with* Scratch*, but with one difference: you no longer move blocks, instead you write the instructions.*

Lesson 1 (The `Python` turtle)**.**
Turtle is the ancestor of *Scratch*! In a few lines you can make beautiful drawings.

```
from turtle import *

forward(100)    # Move forward
left(90)        # Turn 90 degrees left
forward(50)
width(5)        # Width of the pencil
forward(100)
color('red')
right(90)
forward(200)

exitonclick()
```



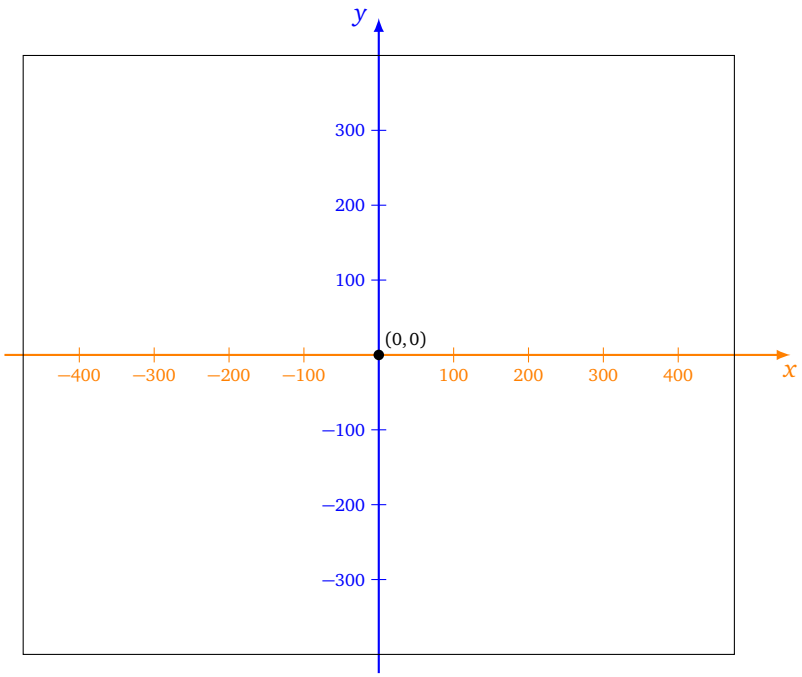
Here is a list of the main commands, accessible after writing:

```
from turtle import *
```

- `forward(length)` advances a number of steps
- `backward(length)` goes backwards
- `right(angle)` turns to the right (without advancing) at a given angle in degrees
- `left(angle)` turns left

- `setheading(direction)` points turtle in a direction (0 = right, 90 = top, −90 = bottom, 180 = left)
- `goto(x,y)` moves to the point (x, y)
- `setx(newx)` changes the value of the abscissa
- `sety(newy)` changes the value of the ordinate
- `down()` sets the pen down
- `up()` sets the pen up
- `width(size)` changes the thickness of the line
- `color(col)` changes the color: `"red"`, `"green"`, `"blue"`, `"orange"`, `"purple"`...
- `position()` returns the (x, y) position of the turtle
- `heading()` returns the direction `angle` to which the turtle is pointing
- `towards(x,y)` returns the angle between the horizontal and the segment starting at the turtle and ending at the point (x, y)
- `exitonclick()` ends the program as soon as you click

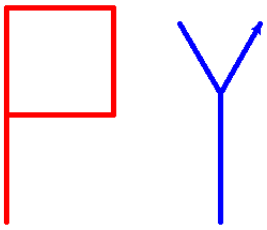
The default screen coordinates range from −475 to +475 for x and from −400 to +400 for y; $(0, 0)$ is in the center of the screen.



Activity 1 (First steps)**.**

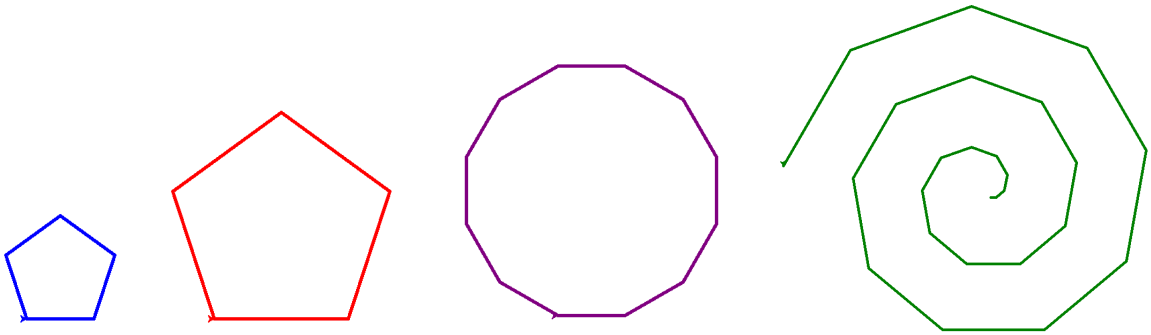
Goal: create your first drawings.

Trace the first letters of `Python`, for example as below.



Activity 2 (Figures).

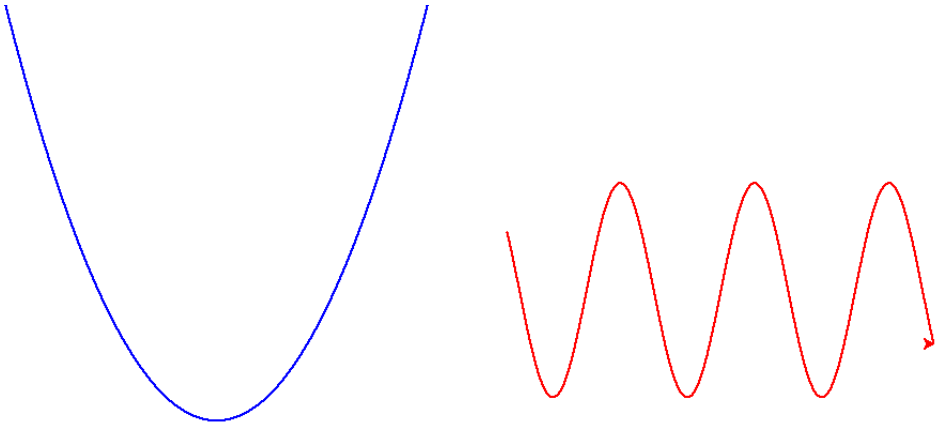
Goal: drawing geometric shapes.



1. **Pentagon.** Draw a first pentagon (in blue). You have to repeat 5 times: advance 100 steps, turn 72 degrees.
   Hint. To build a loop, use
   ```
   for i in range(5):
   ```
   (even if you do not use the variable `i`).
2. **Pentagon (bis).** Define a variable `length` which is equal to 200 and a variable `angle` which is equal to 72 degrees. Draw a second pentagon (in red), this time advancing by `length` and turning by `angle`.
3. **Dodecagon.** Draw a polygon having 12 sides (in purple).
   Hint. To draw a polygon with n sides, it is necessary to turn an angle of $360/n$ degrees.
4. **Spiral.** Draw a spiral (in green).
   Hint. Build a loop, in which you always turn at the same angle, but you move forward by a length that increases with each step.

Activity 3 (Function graph).

Goal: draw the graph of a function.



Plot the graph of the square function and the sine function.
In order to get a curve in the turtle window, repeat for x varying from -200 to $+200$:

- set $y = \frac{1}{100}x^2$,
- go to (x, y).

For the sinusoid, you can use the formula

$$y = 100 \sin\left(\frac{1}{20}x\right).$$

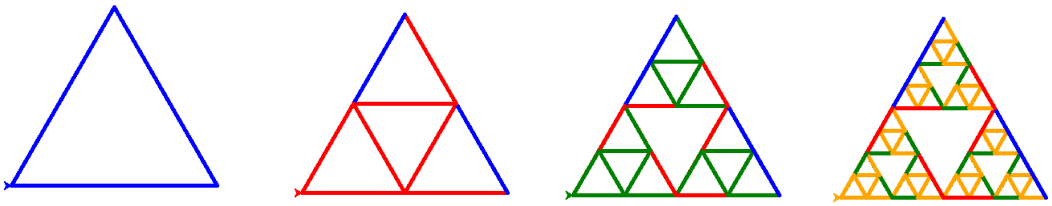
By default `Python` does not know the sine function, to use `sin()` you must first import the `math` module:

```
from math import *
```

To make the turtle move faster, you can use the command `speed("fastest")`.

Activity 4 (Sierpinski triangle)**.**

Goal: trace the beginning of Sierpinski's fractal by nesting loops.



Here is how to picture the second drawing. Analyze the nesting of the loops and draw the next pictures.

```
for i in range(3):
    color("blue")
    forward(256)
    left(120)

    for i in range(3):
        color("red")
        forward(128)
        left(120)
```

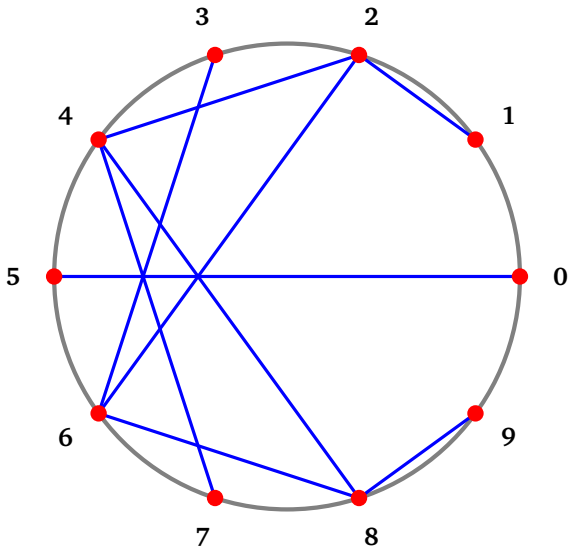
Activity 5 (The heart of multiplication tables)**.**

Goal: draw the multiplication tables.

We set an integer n. We are studying the 2 table, that is to say we calculate 2×0, 2×1, 2×2, up to $2 \times (n-1)$. In addition, the calculations will be modulo n. We therefore calculate

$$2 \times k \pmod{n} \qquad \text{for } k = 0, 1, \ldots, n-1$$

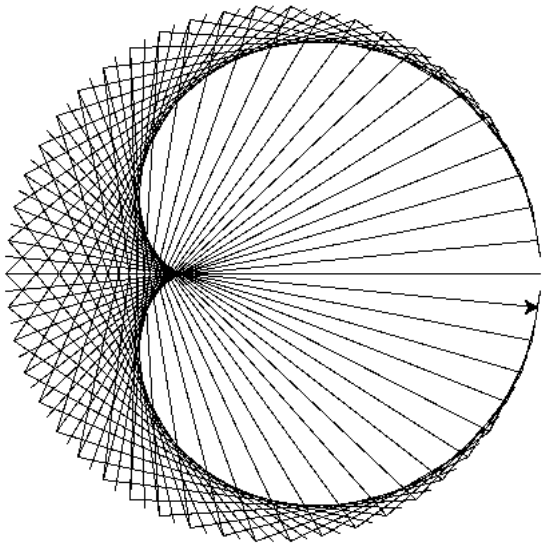
How do we draw this table?
We place n points on a circle, numbered from 0 to $n-1$. For each $k \in \{0, \ldots, n-1\}$, we connect the point number k with the point number $2 \times k \pmod{n}$ by a segment.
Here is the layout, from the table of 2, modulo $n = 10$.



For example:

- the 3 point is linked to the 6 point, because $2 \times 3 = 6$;
- the 4 point is linked to the 8 point, because $2 \times 4 = 8$;
- the 7 point is linked to the 4 point, because $2 \times 7 = 14 = 4 \pmod{10}$.

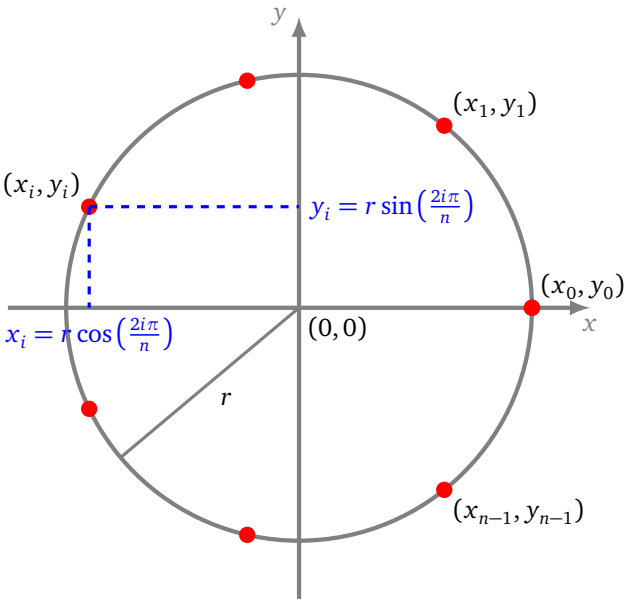
Draw the table of 2 modulo n, for different values of n.
Here is what it gives for $n = 100$.



Hints. For calculations modulo n, use the expression `(2*k) % n`.
Here's how to get the coordinates of the vertices. This is done with the sine and cosine functions (available from `math` module). The coordinates (x_i, y_i) of the vertex number i, can be calculated by the formula :

$$x_i = r\cos\left(\frac{2i\pi}{n}\right) \qquad \text{et} \qquad y_i = r\sin\left(\frac{2i\pi}{n}\right)$$

These points will be located on a circle of radius r, centered at $(0,0)$. You will have to choose r rather large (for example $r = 200$).



Lesson 2 (Several turtles)**.**
Several turtles can be defined and move independently. Here's how to define two turtles (one red and one blue) and move them.

```
turtle1 = Turtle()   # with capital 'T'!
turtle2 = Turtle()
```

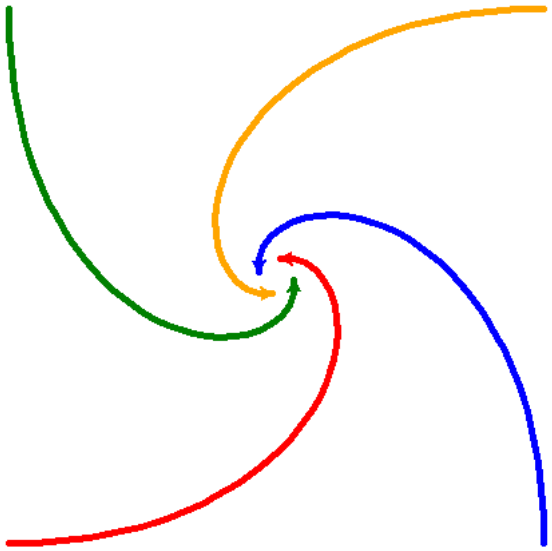
```
turtle1.color('red')
turtle2.color('blue')

turtle1.forward(100)
turtle2.left(90)
turtle2.forward(100)
```

Activity 6 (The pursuit of turtles)**.**

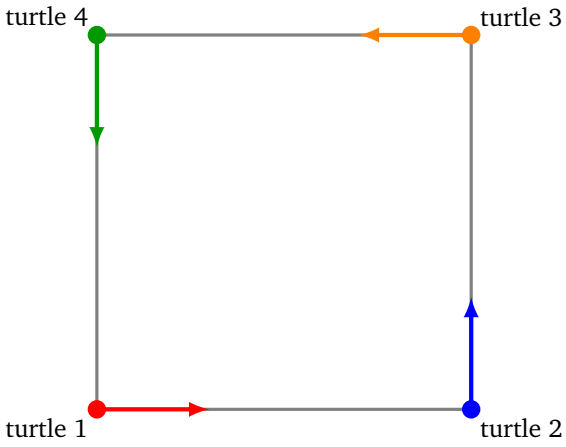
Goal: draw tracking curves.

Program four turtles running one after the other:



- turtle 1 runs after turtle 2,
- turtle 2 runs after turtle 3,
- turtle 3 runs after turtle 4,
- turtle 4 runs after turtle 1.

Here are the starting positions and orientations:



Hints. Use the following piece of code:

```
position1 = turtle1.position()
position2 = turtle2.position()
angle1 = turtle1.towards(position2)
turtle1.setheading(angle1)
```

- You place turtles at the four corners of a square, for example at $(-200, -200)$, $(200, -200)$, $(200, 200)$ and $(-200, 200)$.
- You get the position of the first turtle by using `position1 = turtle1.position()`. Same for the other turtles.
- You calculate the angle between turtle 1 and turtle 2 by the command `angle1 = turtle1.towards(position2)`.
- You orient the first turtle according to this angle: `turtle1.setheading(angle1)`.
- You advance the first turtle by 10 steps.

Improve your program by drawing a segment between the chasing turtle and the chased turtle each time.