# The garden

**This will be an incremental project. First the petals of a flower, then the blossom of the flower, then the stem and leaves, then multiple flowers, etc.**

**The goal is that we practice all that we have learnt and learn a few new things to have under our sleeve:**

- **Define functions and call them**
- **Use random numbers**
- **Parametrization (we'll learn what this means when we get to it)**

## First step: drawing a petal of a flower. Code to get it:

```
from turtle import *

pencolor('yellow') # Set the pen to draw yellow lines

for i in range(3):
    forward(35)
    right(120)

exitonclick()
```

Great so now we'll have our petal! - It'll be just a yellow triangle on the screen, but it will take us far.

## Second step: drawing the blossom of a flower.

- Can you think of a way to you what you already wrote to get the whole flower? Try thinking how you would do it before looking at the code below - Hint: If you think about a flower, you know it has multiple petals. We know how to draw a petal!

```
from turtle import *

speed(0)

def petal():
    pencolor('yellow')

    for i in xrange(3):
        forward(35)
        right(120)

for i in xrange(24):
    petal()
    right(15)

exitonclick()
```

- We define a function call petal, which is in charge of drawing a single petal each time is called. - Note: It's the same code we had before, just that now is inside a function.
- We write functions starting with the following format "def name_we_want_for_our_function(name_of_arguments_it_needs):" . - Note: This is called the *signature* of our function
- We write what the function needs to do following the signature. We can use inside the function any of the arguments we had pass it. - Note: This is called the *body* of our function and also note that all the body needs to be more to the right than the signature.
- But the function needs to be call to do its task. We call functions looking at their signature in the following way: "name_of_function(value_for_the_arguments_it_needs)" . - Note: It's very similar to the signature but the "def" and the ":" are no longer there. Also, instead of name for arguments, we need values. For example if the signature of the function has colour as argument we need to call it with "blue", or "red"
- In our case, the function petal has the following signature, body and how to call it:
    - Signature: `def petal():`
    - Body:

```
    pencolor('yellow')

    for i in xrange(3):
        forward(35)
        right(120)
```

    - How to call it: `petal()`

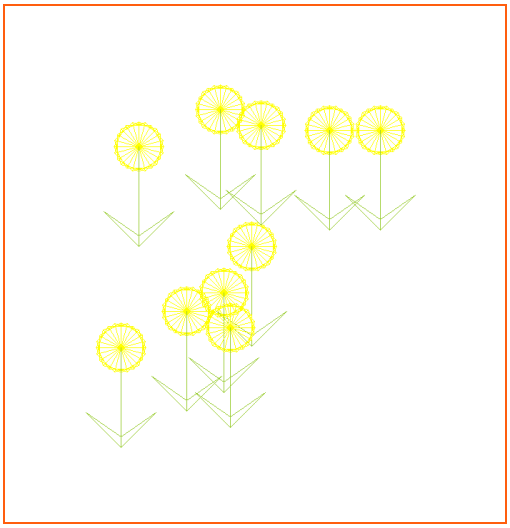
## Third step: drawing the whole flower. Code to get it:

```
from turtle import *

def petal():
    pencolor('yellow')

    for i in xrange(3):
        forward(35)
        right(120)


def blossom():
    for i in xrange(24):
        petal()
        right(15)


def stem_and_leaves():
    pencolor('Yellow Green')

    # the stem
    left(90)
    back(140)

    # the left leaf
    left(45)
    forward(70)
    left(10)
    back(60)

    # the right leaf
    right(110)
    forward(60)
    left(10)
    back(71)

speed(0) # this is optional, it makes the drawing faster

# Our full flower
setheading(0) # Set the orientation of the turtle to the number you give it
blossom() # calling this function will draw the blossom of the flower
stem_and_leaves() # calling this fucntion will draw the steam and leaves
hideturtle() # this makes hides the drawing arrow

exitonclick()
```

- We already know how to define the blossom so we now define another fuction to draw the stem and we put them together!
- The setheading is maybe the most subtle thing in this piece of code, it changes “where the turtle is looking”.

## Fourth step: drawing a garden.

- We'll be introducing a couple of new things here, don't get intimidated you'll learn them little by little by using them. And once you have learnt them you'll discover how much you can do with them!



## Code to get it (next page)

- goto(x,y)

Moves turtle to an absolute position. If the pen is down, draw line. It does not change the turtle's orientation. You pass them a set of coordinates.

- randint(x,y)

Returns a random (this means unknown or unspecified) number *N* such that x <= N <= y. This function is something we borrow from the library random, which we are importing at the beginning of the file. Being able to borrow functions that other people has written is very useful.

```
from turtle import *
from random import randint

def petal():
    pencolor('yellow')

    for i in xrange(3):
        forward(35)
        right(120)


def blossom():
    for i in xrange(24):
        petal()
        right(15)


def stem_and_leaves():
    pencolor('Yellow Green')

    # the stem
    left(90)
    back(140)

    # the left leaf
    left(45)
    forward(70)
    left(10)
    back(60)

    # the right leaf
    right(110)
    forward(60)
    left(10)
    back(71)


def flower():
    setheading(0)
    blossom()
    stem_and_leaves()
    hideturtle()

# Our (a little monotone) garden
speed(0)
for i in xrange(10):
    #generate random coordinates to
    # draw the next flower
    x = randint(-200,200)
    y = randint(-200,200)

    flower()
    penup()
    goto(x,y)
    pendown()

exitonclick()
```