STEP IT ACADEMY

Programming with python

Lesson # 3

Python Turtle Graphics. The for Loop

CONTENTS

Variables

Python is a **dynamically-typed** language, which means that the **data type is determined already during the execution of the program** and it is not necessary to write it for each variable.

Let's look at this example and complete it:

```
a = 8
b = 0.8
c = "string"
d = True
print(type(a), type(b), type(c), type(d))
```

Here we created four different variables and defined their types. To define a type in Python, there is a special function, `type()`. The code will result in the following line:

```
<class 'int'> <class 'float'> <class 'str'>
<class 'bool'>
```

These are the four basic types of data that you are already familiar with. Correspondingly, `int (integer)` is used for integers (-765, 0, 20), `float` for floating-point numbers (-15.0, 0.1, 225.5), `str (string)` for a string ("Tom", "Hello World"), and `bool` for a logical data type, i. e. one that can answer the question either yes or no (`True`/`False`) (Figure 1).

Figure 1

In one form or another, variables will be needed in almost every program, especially for **loops** and **conditions**.

Loops in Programming

People constantly repeat some actions, and programmers are no exception. They especially do not like to do extra work, so **loops** are used to repeat actions.

A loop is a repetition of a specific set of commands. For example, if a person does push-ups 10 times, they perform the same action 10 times.

Loops can be of several types:

- with a known number of repetitions (**with a parameter**);
- with an unknown number of repetitions (**conditional**).

If you return to the example with push-ups, the phrase "do 40 push-ups" means that the action "push-ups" will be performed in a loop, which should be repeated 40 times. If you say: "Do push-ups until you get tired", this will be a conditional loop when we initially do not know how many times it will be repeated.

The for Loop

The for loop is a loop with a parameter (a specified number of repetitions). This loop should be used in such cases:

- repetition of certain actions for a set number of times;
- getting a number that will change with each repetition.

The **for** loop syntax is as follows:

```
for i in sequence_generator:
   instructions
```

Let's use the turtle to draw not one figure but three (see Figure 2 on page 7):

```
import turtle

window = turtle.Screen()
turtle.reset()


turtle.shape("turtle")
turtle.bgcolor("dark slate gray")
turtle.color("alice blue")

turtle.speed(2)
turtle.pensize(3)

turtle.left(20)

turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)

turtle.left(20)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)

turtle.left(20)

turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)

window.exitonclick()
```

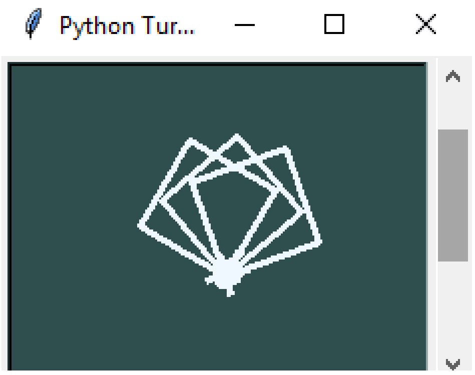

Figure 2

As you can see, we have to write lots of code. And what if you need about a hundred of such squares?

It's not the best decision to re-write everything all the time, right? Moreover, this approach is not suitable for programmers. Therefore, the `for` loop comes to rescue.

Look how much more compact and understandable this code is:

```
import turtle

window = turtle.Screen()
turtle.reset()

turtle.shape("turtle")
turtle.bgcolor("dark slate gray")
turtle.color("alice blue")
turtle.speed(4)
turtle.pensize(3)
for i in 1,2,3,4,5,6:
  turtle.left(30)

  turtle.forward(40)
  turtle.left(90)
  turtle.forward(40)
  turtle.left(90)
  turtle.forward(40)
  turtle.left(90)
  turtle.forward(40)
  turtle.left(90)

window.exitonclick()
```

In our code, we used:

```
for i in 1,2,3,4,5,6
```

This loop will repeat drawing of the square 6 times.

However, if you want to repeat an action for, say, 100 or 200 times, then this kind of notation will be inconvenient. Just imagine how much time you will spend on it! Therefore, it is better to use the `range` function.

Range Function

The `for` loop works according to the following principle: it iterates through all the values in turn and executes the code after the colon.

The `range` function returns a series of numbers with given constraints. The number in brackets determines the number of repetitions, for example, `range(10)` indicates that our turtle will draw ten figures. The value in brackets indicates that the loop will repeat 10 times. The `range` function can take more than one parameter, for example, `range(0, 10)` or `range(10, 20)`. In the second case, the parameter will take a value from 10 to 19 and will be repeated 10 times, as in all the examples considered.

Note: You can also specify step size.
So, `range(0, 12, 2)` *will result in the following:* `0,2,4,6,8,10`, *and step size can even be negative:* `range(10, 0, -1)`:

```
for i in range(10, 0, -1):
  print(i)
```

Let’s write the code where we use the `for` loop and the `range()` function. Let’s teach the turtle to draw a beautiful figure using regular squares (see Figure 3):

```
import turtle

window = turtle.Screen()
turtle.reset()

turtle.shape("turtle")
turtle.bgcolor("dark slate gray")
turtle.color("alice blue")

turtle.speed(4)
turtle.pensize(3)
for i in range(12):
  turtle.left(30)


  turtle.forward(40)
  turtle.left(90)
  turtle.forward(40)
  turtle.left(90)
  turtle.forward(40)
  turtle.left(90)
  turtle.forward(40)
  turtle.left(90)


window.exitonclick()
```

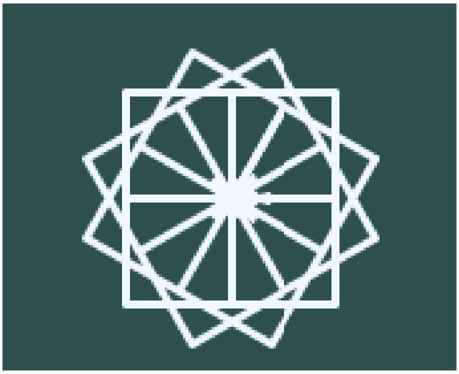

Figure 3

Drawing Various Shapes

It’s good if you understand the material and can make changes to the code yourself.

And now let's use the loop and the turtle to create a shape consisting of circles (see Figure 4 on page 14):

```
import turtle

window = turtle.Screen()
turtle.reset()

turtle.shape("turtle")
turtle.bgcolor("black")
turtle.pencolor("purple")


turtle.speed(10)
turtle.pensize(2)

for i in range(30):
  turtle.circle(5 * i)
  turtle.circle(-5 * i)
  turtle.left(i)

turtle.exitonclick()
```

Or spiral shape (see Figure 5 on page 14):

```
import turtle

window = turtle.Screen()
turtle.reset()

turtle.shape("turtle")
turtle.bgcolor("black")
turtle.pencolor("yellow")

turtle.speed(10)
turtle.pensize(2)

for i in range(360):
  turtle.pensize(i/100 + 1)
  turtle.forward(i)
  turtle.left(59)

turtle.exitonclick()
```

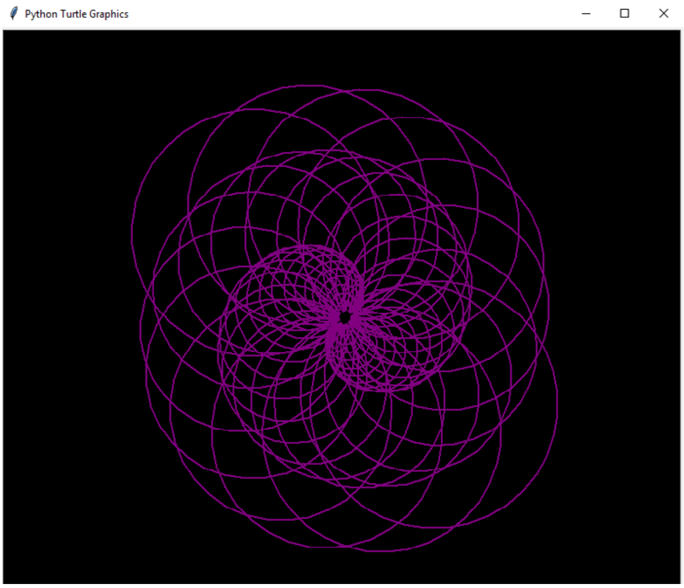

Figure 4

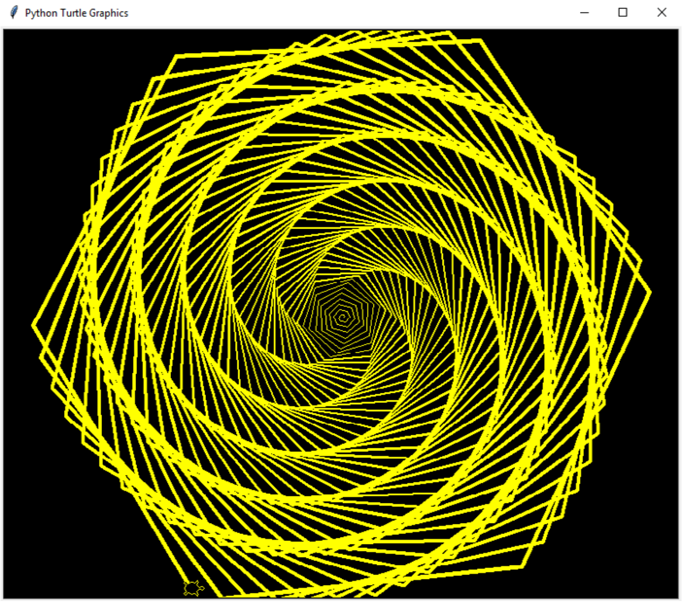

Figure 5

The obtained knowledge will allow you to create various figures and shapes and set all kinds of parameters.

Lesson #3

Python Turtle Graphics. The for Loop