FUNCTION

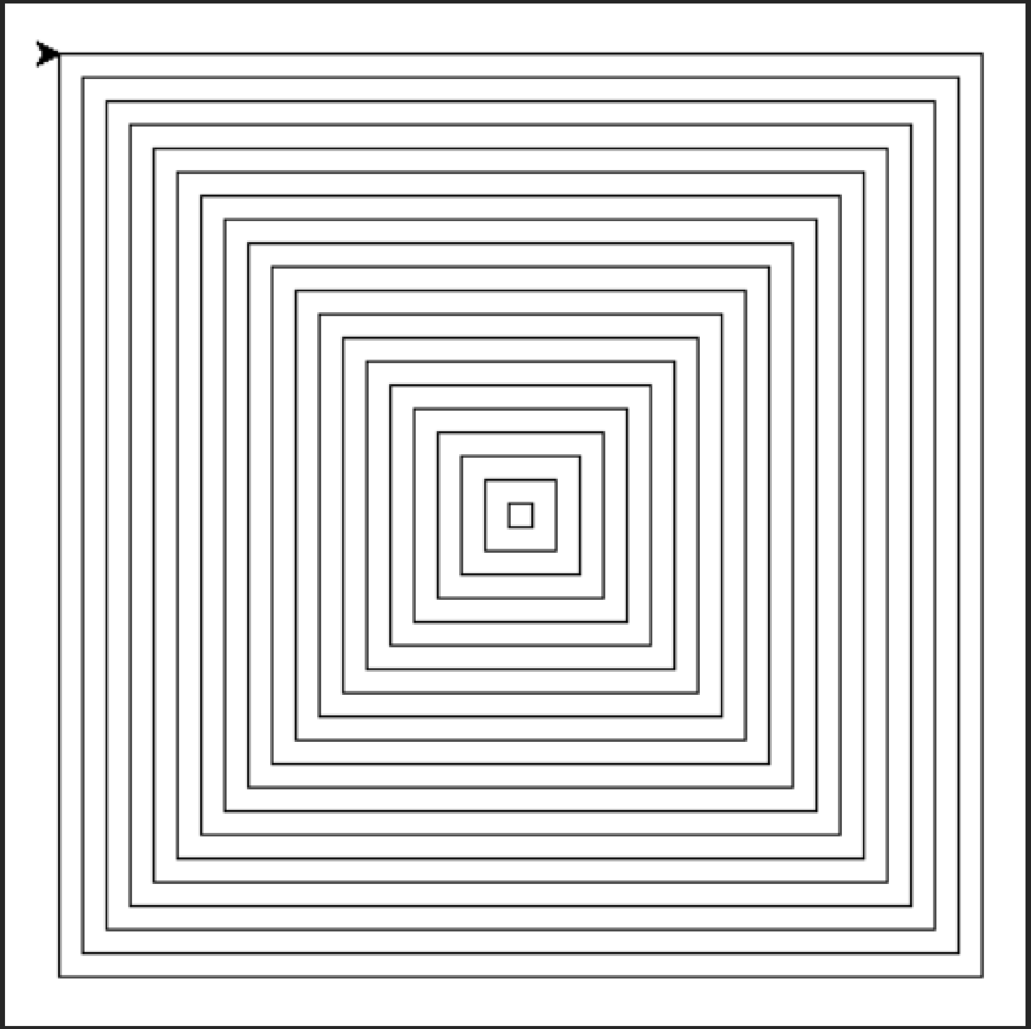
by

Lilian Blot

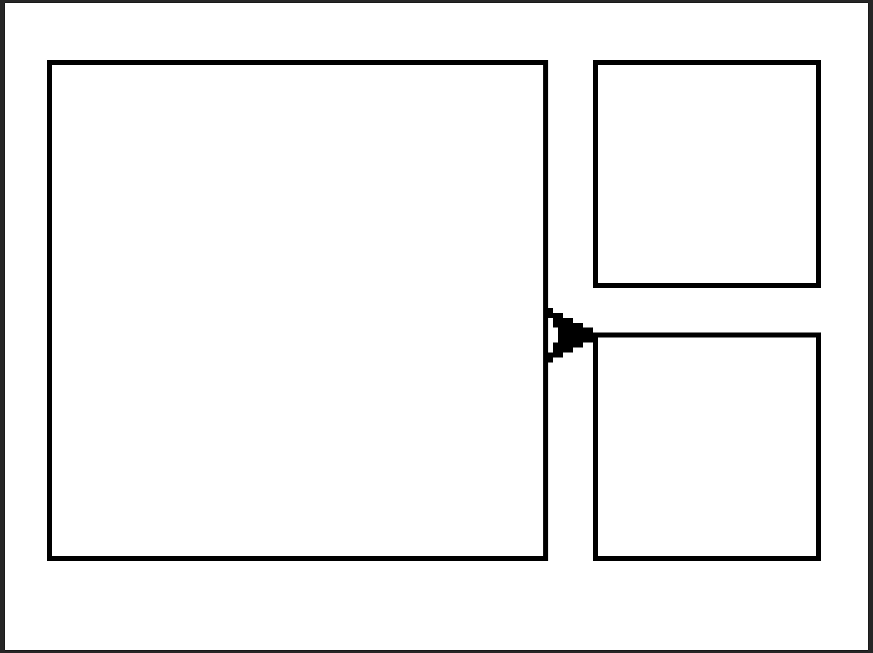
Code

```
for n in range(20):
    # move cursor (square’s top-left
    # corner) to the correct position
    turtle.penup()
    turtle.goto(-length/2, length/2)
    turtle.pendown()
    # draw a square
    for i in range(4):
        turtle.forward(length)
        turtle.right(90)

    # increase length of next square
    length = length + 10
```



Code

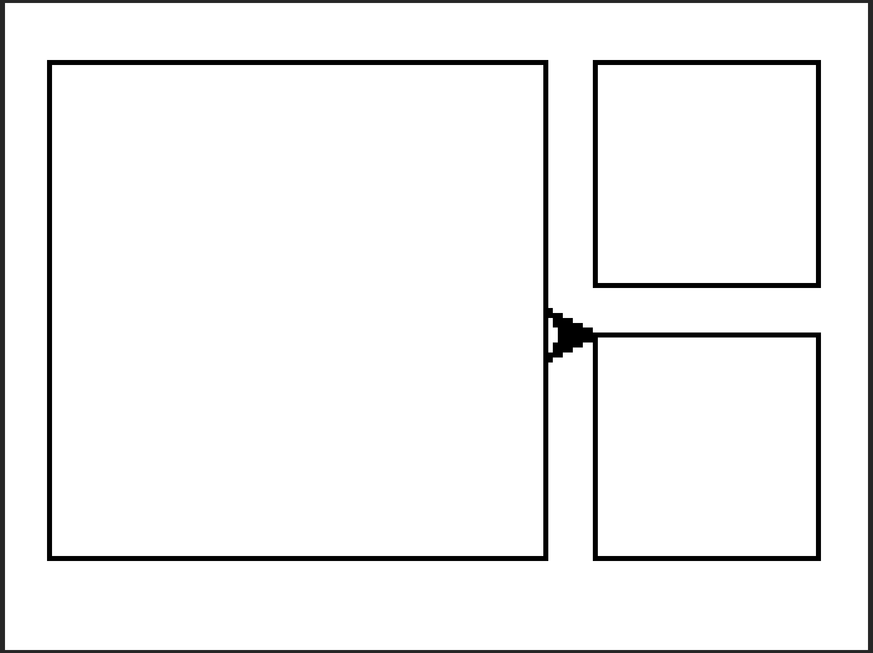


Code

```
my_turtle.penup()
my_turtle.goto(     ,     )
my_turtle.pendown()
for i in range(4):
    my_turtle.forward(    )
    my_turtle.right(90)
```

```
my_turtle.penup()
my_turtle.goto      ,     
my_turtle.pendown()
for i in range(4):
    my_turtle.forward    
    my_turtle.right(90)
```

```
my_turtle.penup()
my_turtle.goto      ,     
my_turtle.pendown()
for i in range(4):
    my_turtle.forward    )
    my_turtle.right(90)
```



Code

```
my_turtle.penup()
my_turtle.goto(-100, 100)
my_turtle.pendown()
for i in range(4):
    my_turtle.forward(100)
    my_turtle.right(90)
```

```
my_turtle.penup()
my_turtle.goto(10, 100)
my_turtle.pendown()
for i in range(4):
    my_turtle.forward(45)
    my_turtle.right(90)
```

```
my_turtle.penup()
my_turtle.goto(10, 45)
my_turtle.pendown()
for i in range(4):
    my_turtle.forward(45)
    my_turtle.right(90)
```

Monolithic code:

- huge collection of statement
- no modularisation
- no code reuse (copy & paste is not code reuse!)
- no parallel implementation

Monolithic Programs cannot be maintained, nor reused easily.

Modularisation

Modularisation

- Decomposition of complex problem into smaller sub-problem
- Separation of concerns
- Semantically coherent

Creation of Modules and declaration of Functions can help.

Why Function?

- having similar code in two places has some drawbacks
- failing to keep related part of the code in sync is a common problem in program maintenance
- function can be used to reduce code duplication

What is a Function?

- We are familiar with maths' function like $f(x) = 2 * x +1$
- In programming it is a kind of a sub-program
- A function definition describes what the function does
- Calling or invoking a function executes the instructions from the function's definition.

Function Declaration

```
def <name> (<parameters>):
    <body>
```

- `<name>`: an identifier
- `<parameters>`: comma-separated identifiers
- `<body>`: any number of indented statements.

Function Declaration

```
def <name> (<parameters>):
    <body>
```

- **<name>**: an identifier
- **<parameters>**: comma-separated identifiers
- **<body>**: any number of indented statements.

Code

```
def square(x,y,length):
    my_turtle.penup()
    my_turtle.goto(x, y)
    my_turtle.pendown()
    for i in range(4):
        my_turtle.forward(length)
        my_turtle.right(90)
```

Function Call

Code

```
def square(x,y,length):
    my_turtle.penup()
    my_turtle.goto(x, y)
    my_turtle.pendown()
    for i in range(4):
        my_turtle.forward(length)
        my_turtle.right(90)


square(-100, 100, 100)
square(10, 100, 45)
square(10, 45, 45)
```

Function Call

1 function declaration

3 Function calls/invocations

Code

```
def square(x,y,length):
    my_turtle.penup()
    my_turtle.goto(x, y)
    my_turtle.pendown()
    for i in range(4):
        my_turtle.forward(length)
        my_turtle.right(90)
```

```
square(-100, 100, 100)
square(10, 100, 45)
square(10, 45, 45)
```

“Write once,
Use many times.”

Function Design

- Key point 1: Separation of concerns
- Key point 2: A function should do one and only one thing (most of the time)
- Key point 3: Decide which information is needed by the function to do its computation… (parameters)
- Key point 4: Does the function need to return zero, one or multiple values.

Preconditions & Postconditions

Frequently a programmer must communicate precisely what a function accomplishes, without any indication of how the function does it.

The precondition statement indicates what must be true before the function is called.

The postcondition statement indicates what will be true when the function finishes its work.

⚠ The postcondition is only guaranteed if the precondition is satisfied

The programmer who calls the function is responsible for ensuring that the precondition is valid when the function is called.

The programmer who implements the function counts on the precondition being valid, and ensures that the postcondition is true at the function’s end.

Advantages

Succinctly describes the behaviour of a function (the what) without cluttering up your thinking with details implementation (the how).

“Code refactoring is the process of restructuring existing computer code without changing its external behaviour.”

You should have a better understanding of the advantages of modularisation, and in particular the use of functions.

You have seen how to refactor your code by declaring and calling functions, resulting in the removal of code duplication.