12.2. Function Definition

The syntax for creating a named function, a **function definition**, is:

```
def name( parameters ):
    statements
```

You can make up any names you want for the functions you create, except that you can't use a name that is a Python keyword, and the names must follow the rules for legal identifiers that were given previously. The parameters specify what information, if any, you have to provide in order to use the new function. Another way to say this is that the parameters specify what the function needs to do its work.

There can be any number of statements inside the function, but they have to be indented from the `def`. In the examples in this book, we will use the standard indentation of four spaces. Function definitions are the third of several **compound statements** we will see, all of which have the same pattern:

1. A header line which begins with a keyword and ends with a colon.
2. A **body** consisting of one or more Python statements, each indented the same amount – *4 spaces is the Python standard* – from the header line.

We've already seen the `for` statement which has the same structure, with an indented block of code, and the `if`, `elif`, and `else` statements that do so as well.

In a function definition, the keyword in the header is `def`, which is followed by the name of the function and some *parameter names* enclosed in parentheses. The parameter list may be empty, or it may contain any number of parameters separated from one another by commas. In either case, the parentheses are required.

We will come back to the parameters in a little while, but first let's see what happens when a function is executed, using a function without any parameters to illustrate.

Here's the definition of a simple function, hello.

Save & Run Original - 1 of 1 Show in CodeLens

```
def hello():
    """This function says hello and greets you"""
    print("Hello")
    print("Glad to meet you")

```

Activity: 1 -- ActiveCode (ac11_1_1)

docstrings

If the first thing after the function header is a string (some tools insist that it must be a triple-quoted string), it is called a **docstring** and gets special treatment in Python and in some of the programming tools.

Another way to retrieve this information is to use the interactive interpreter, and enter the expression `<function_name>.__doc__`, which will retrieve the docstring for the function. So the string you write as documentation at the start of a function is retrievable by python tools *at runtime*. This is different from comments in your code, which are completely eliminated when the program is parsed.

By convention, Python programmers use docstrings for the key documentation of their functions.

We can apply functions to the turtle drawings we've done in the past as well.

Save & Run Original - 1 of 1

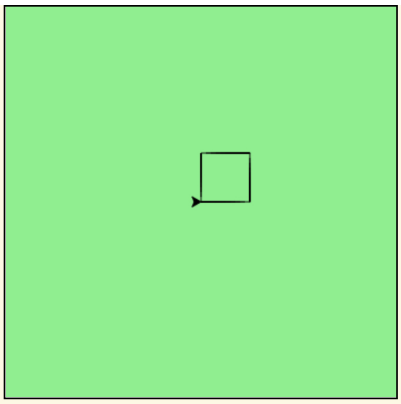
```
import turtle

def drawSquare(t, sz):
    """Make turtle t draw a square of with side sz."""

    for i in range(4):
        t.forward(sz)
        t.left(90)


wn = turtle.Screen()      # Set up the window and its attributes
wn.bgcolor("lightgreen")

alex = turtle.Turtle()    # create alex
drawSquare(alex, 50)      # Call the function to draw the square passing the actua
```



Activity: 2 -- ActiveCode (ac11_1_2)

This function is named `drawSquare`. It has two parameters — one to tell the function which turtle to move around and the other to tell it the size of the square we want drawn. In the function definition they are called `t` and `sz` respectively. Make sure you know where the body of the function ends — it depends on the indentation and the blank lines don't count for this purpose!

You have attempted 3 of 2 activities on this page

12.1. Introduction to Functions">

troduction to Functions">

12.3. Function Invocation">

✔ Completed. Well Done!

12.3. Function Invocation">Next Section - 12.3. Function Invocation

| Back to top