Defining Functions

Function Definitions

The general form of a function definition:

```
def function_name(parameters):
    body
```

- `def`: a keyword indicating a function definition
- function_name: the function name
- parameters:
    - the parameter(s) of the function, 0 or more and are separated by a comma
    - a parameter is a variable whose value will be supplied when the function is called
- body:
    - 1 or more statements, often ending with a `return` statement

Example of a function definition:

```
def f(x):
    return x ** 2
```

`return` statement

The general form of a `return` statement:

```
return expression
```

The rules for executing a `return` statement:

1. Evaluate the expression. This produces a memory address.
2. Pass back that memory address to the caller. Exit the function.

Function Calls

Function calls are expressions and the result can be stored in a variable.

The general form of a function call:

```
function_name(arguments)
```

The rules for executing a function call:

1. Evaluate the arguments to produce memory addresses.
2. Store those memory addresses in the corresponding parameters.
3. Execute the body of the function.

Example of a function definition and function calls:

```
>>> def area(base, height):
    return base * height / 2
>>> area(3, 4)
6.0
>>> result = area(10, 7.45)
>>> result
37.25
```

Saving your programs to ".py" files

We usually save our Python programs in ".py" files. A file can contain multiple function definitions and other statements. Before calling a function from a ".py" file in the shell in IDLE, you need to first execute Run -> Run Module, or else the shell will not recognize the function call.

Extra readings:

Chaper 3.3. Defining Our Own Functions

Chaper 3.4. Using Local Variables for Temporary Storage

Chaper 3.7. Writing and Running a Program

Optional reading

Jennifer Campbell • Paul Gries
University of Toronto