A Short and Incomplete Introduction to Python

Part 2: Functions

Riccardo Murri <riccardo.murri@uzh.ch>, Sergio Maffioletti <sergio.maffioletti@uzh.ch> S3IT: Services and Support for Science IT, University of Zurich

Functions

Functions, I

Functions are called by postfixing the function name with a parenthesized argument list.

```
>>> int("42")
42
>>> int(4.2)
4
>>> float(42)
42.0
>>> str(42)
'42'
>>> str()
```

Functions, II

Some functions can take a variable number of arguments. For instance:

```
\operatorname{sum}(x_0, \ldots, x_n) Return x_0 + \cdots + x_n.

\operatorname{max}(x_0, \ldots, x_n) Return the maximum of \{x_0, \ldots, x_n\}

\operatorname{min}(x_0, \ldots, x_n) Return the minimum of \{x_0, \ldots, x_n\}
```

Examples:

```
In [1]: min(1,2,3)
Out[1]: 1
In [2]: max(1,2)
Out[2]: 2
```

The most important function of all

help(fn) Display help on the function named fn

Exercise 2.A: What happens if you type these at the prompt?

- ▶ help(abs)
- ▶ help(help)

How to define new functions

The **def** statement starts a function definition.

```
def greet(name):
    """
    A friendly function.
    """
    print ("Hello, " + name + "!")
# the customary greeting
greet("world")
```

```
def greet (name):
    """
    A friendly function.
    """
    print ("Hello, " + name + "!")

# the customary greeting
greet("world")
in Python: it is used to
delimit blocks of code, like
    '{' and '}' in Java and C.
```

Indentation is significant

(This is a comment. It is

This calls the function just defined.

```
def greet (name):
    """
    A friendly function.
    """
    print ("Hello, " + name + "!")
# the customary greeting
    greet ("world")
```

What is this? The answer in the next exercise!

```
def greet (name):
    """
    A friendly function.
    """
    print ("Hello, " + name + "!")
# the customary greeting
greet("world")
```

Exercise 2.B: Type and run the code on the previous page at the interactive prompt. (Pay attention to indentation!)

What's the result of evaluating the function greet ("world")?

What does help(greet) output?

Default values

Function arguments can have default values.

```
>>> def hello(name='world'):
... print ("Hello, " + name)
...
>>> hello()
'Hello, world'
```

Named arguments

Python allows calling a function with named arguments:

```
hello(name="Alice")
```

When passing arguments by name, they can be passed in any order:

```
>>> from fractions import Fraction
>>> Fraction(numerator=1, denominator=2)
Fraction(1, 2)
>>> Fraction(denominator=2, numerator=1)
Fraction(1, 2)
```

The 'return' statement

```
def double(x):
    return x+x

double(3) == 6
```

The result of a function evaluation is set by the *return* statement.

If no return is present, the function returns the special value None.

```
def double(x):
    return x+x
# the following line
# is never exec'd
    print('Hello')
```

After executing return the control flow leaves the function.

Basic control flow

Conditionals

Conditional execution uses the if statement:

if expr:

indented block

elif other-expr:

indented block

else:

executed if none of the above matched

The elif can be repeated, with different conditions, or left out entirely.

Also the else clause is optional.

Q: Where's the 'end if'?

Conditionals

Conditional execution uses the if statement:

if expr:

indented block

elif other-expr:

indented block

else:

executed if none of the above matched

The elif can be repeated, with different conditions, or left out entirely.

Also the else clause is optional.

9: Where's the 'end if'? There's no 'end if': indentation delimits blocks!

while-Loops

Conditional looping uses the while statement:

while expr:

indented block

To break out of a while loop, use the break statement.

Use the continue statement anywhere in the indented block to jump back to the while statement.

Exercise 2.C: Modify the greet() function to print "Welcome back!" if the argument name is the string 'Python'.

Modules

Modules, I

The import statement reads a .py file, executes it, and makes its contents available to the current program.

```
>>> import hello Hello, world!
```

Modules are only read once, no matter how many times an import statement is issued.

```
>>> import hello
Hello, world!
>>> import hello
>>> import hello
```

Modules, II

Modules are *namespaces*: functions and variables defined in a module must be prefixed with the module name when used in other modules:

```
>>> hello.greet("Python")
Hello, Python!
```

To import definitions into the current namespace, use the 'from x import y' form:

```
>>> from hello import greet
>>> greet("Python")
Hello, Python!
```