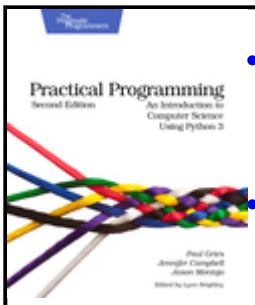
Built-in Functions



- Chaper 3.1. Functions That Python Provides
- Chaper 3.2. Memory Addresses: How Python Keeps Track of Values

Optional reading

Function Call

The general form of a function call:

```
function_name(arguments)
```

The rules for executing a function call:

1. Evaluate the arguments.
2. Call the function, passing in the argument values.

Terminology:

- *Argument*: a value given to a function
- *Pass*: to provide to a function
- *Call*: ask Python to evaluate a function
- *Return*: pass back a value

Example function calls:

```
>>> abs(-23)
23
>>> abs(56.24)
56.24
```

Function `dir`

Python has a set of built-in functions. To see the list of built-in functions, run `dir(__builtins__)`:

```
>>> dir(__builtins__)< br/> ['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException',
'BufferError', 'BytesWarning', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError',
'Exception', 'False', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError',
'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'KeyError', 'KeyboardInterrupt',
'LookupError', 'MemoryError', 'NameError', 'None', 'NotImplemented', 'NotImplementedError',
'OSError', 'OverflowError', 'PendingDeprecationWarning', 'ReferenceError', 'ResourceWarning',
'RuntimeError', 'RuntimeWarning', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError',
'SystemExit', 'TabError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError',
'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning',
'ValueError', 'Warning', 'ZeroDivisionError', '_', '__build_class__', '__debug__', '__doc__',
'__import__', '__name__', '__package__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray',
'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr',
'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format',
'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int',
'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview',
'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr',
'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super',
'tuple', 'type', 'vars', 'zip']
```

Function `help`

To get information about a particular function, call `help` and pass the function as the argument. For example:

```
>>> help(abs)
Help on built-in function abs in module builtins:
abs(...)
    abs(number) -> number

    Return the absolute value of the argument.
```

Optional arguments

In the description of function `pow` below, the square brackets around `[, z]` indicate that the third argument is optional:

```
>>> help(pow)
Help on built-in function pow in module builtins:

pow(...)
    pow(x, y[, z]) -> number

    With two arguments, equivalent to x**y.  With three arguments,
    equivalent to (x**y) % z, but may be more efficient (e.g. for longs).
```

Function `pow` can be called with either two or three arguments:

```
>>> pow(2, 5)
32
>>> pow(2, 5, 3)
2
```

Jennifer Campbell • Paul Gries
University of Toronto
