

**intro** Introduction: What is Python?  
**python3** Python 2.7 and Python 3.x  
**starting** Starting Python  
   **starting-windows** Using Python in Windows  
   **starting-linux** Using Python in Linux  
**line-syntax** Line syntax  
**names** Names and keywords  
**types** Basic types  
**numeric-types** Numeric types  
   **int-type** Type `int`: Integers  
   **long-type** Type `long`: Extended-precision integers  
   **bool-type** Type `bool`: Boolean truth values  
   **float-type** Type `float`: Floating-point numbers  
   **complex-type** Type `complex`: Imaginary numbers  
**sequence-types** Sequence types  
   **sequence-common** Operations common to all the sequence types  
**str-type** Type `str`: Strings of 8-bit characters  
   **str-constants** String constants  
   **whitespace** Definition of "whitespace"  
   **str-methods** Methods on `str` values  
   **new-str-format** The string `.format()` method  
     **general-format-form** General form of a format code  
     **format-name** The *name* part  
     **format-conversion** The *conversion* part  
     **format-spec** The *spec* part  
     **format-var-length** Formatting a field of variable length  
   **old-str-format** The older string format operator  
**unicode-type** Type `unicode`: Strings of 32-bit characters  
   **utf-8** The UTF-8 encoding  
**list-type** Type `list`: Mutable sequences  
   **list-methods** Methods on lists  
   **list-comprehensions** List comprehensions  
**tuple-type** Type `tuple`: Immutable sequences  
**bytes-type** The `bytes` type  
   **bytes-to-3x** Using the `bytes` type in 3.x conversion  
**bytearray-type** The `bytearray` type  
**set-types** Types `set` and `frozenset`: Set types  
   **set-immutable-operations** Operations on mutable and immutable sets  
   **set-mutable-operations** Operations on mutable sets  
**dict-type** Type `dict`: Dictionaries  
   **dict-methods** Operations on dictionaries  
   **dict-comprehensions** Dictionary comprehensions  
**file-type** Type `file`: Input and output files  
   **file-methods** Methods on `file` objects  
**None-type** `None`: The special placeholder value

**expressions** Operators and expressions  
**predicates** What is a predicate?  
**iterable** What is an iterable?  
**interface** Duck typing, or: what is an interface?  
**locale** What is the locale?  
**basic-functions** Basic functions  
**abs-function** `abs()`: Absolute value  
**all-function** `all()`: Are all the elements of an iterable true?  
**any-function** `any()`: Are any of the members of an iterable true?  
**bin-function** `bin()`: Convert to binary  
**bool-function** `bool()`: Convert to Boolean  
**bytearray-function** `bytearray()`: Create a byte array  
**chr-function** `chr()`: Get the character with a given code  
**cmp-function** `cmp()`: Compare two values  
**complex-function** `complex()`: Convert to `complex` type  
**dict-function** `dict()`: Convert to a dictionary  
**divmod-function** `divmod()`: Quotient and remainder  
**enumerate-function** `enumerate()`: Step through indices and values of an iterable  
**file-function** `file()`: Open a file  
**filter-function** `filter()`: Extract qualifying elements from an iterable  
**float-function** `float()`: Convert to `float` type  
**format-function** `format()`: Format a value  
**frozenset-function** `frozenset()`: Create a frozen set  
**hex-function** `hex()`: Convert to base 16  
**int-function** `int()`: Convert to `int` type  
**input-function** `input()`: Read an expression from the user  
**iter-function** `iter()`: Produce an iterator over a sequence  
**len-function** `len()`: Number of elements  
**list-function** `list()`: Convert to a list  
**long-function** `long()`: Convert to `long` type  
**map-function** `map()`: Apply a function to each element of an iterable  
**max-function** `max()`: Largest element of an iterable  
**min-function** `min()`: Smallest element of an iterable  
**next-function** `next()`: Call an iterator  
**oct-function** `oct()`: Convert to base 8  
**open-function** `open()`: Open a file  
**ord-function** `ord()`: Find the numeric code for a character  
**pow-function** `pow()`: Exponentiation  
**range-function** `range()`: Generate an arithmetic progression as a list  
**raw\_input-function** `raw_input()`: Prompt and read a string from the user  
**reduce-function** `reduce()`: Sequence reduction  
**reversed-function** `reversed()`: Produce a reverse iterator  
**round-function** `round()`: Round to the nearest integral value  
**set-function** `set()`: Create an algebraic set

**sorted-function** `sorted()`: Sort a sequence

**str-function** `str()`: Convert to `str` type

**sum-function** `sum()`: Total the elements of a sequence

**tuple-function** `tuple()`: Convert to a tuple

**type-function** `type()`: Return a value's type

**unichr-function** `unichr()`: Convert a numeric code to a Unicode character

**unicode-function** `unicode()`: Convert to a Unicode string

**xrange-function** `xrange()`: Arithmetic progression generator

**zip-function** `zip()`: Combine multiple sequences

**advanced-functions** Advanced functions

**basestring-function** `basestring`: The string base class

**callable-function** `callable()`: Is this thing callable?

**classmethod-function** `classmethod()`: Create a class method

**delattr-function** `delattr()`: Delete a named attribute

**dir-function** `dir()`: Display a namespace's names

**eval-function** `eval()`: Evaluate an expression in source form

**execfile-function** `execfile()`: Execute a Python source file

**getattr-function** `getattr()`: Retrieve an attribute of a given name

**globals-function** `globals()`: Dictionary of global name bindings

**hasattr-function** `hasattr()`: Does a value have an attribute of a given name?

**id-function** `id()`: Unique identifier

**isinstance-function** `isinstance()`: Is a value an instance of some class or type?

**issubclass-function** `issubclass()`: Is a class a subclass of some other class?

**locals-function** `locals()`: Dictionary of local name bindings

**property-function** `property()`: Create an access-controlled attribute

**reload-function** `reload()`: Reload a module

**repr-function** `repr()`: Representation

**setattr-function** `setattr()`: Set an attribute

**slice-function** `slice()`: Create a slice instance

**staticmethod-function** `staticmethod()`: Create a static method

**super-function** `super()`: Superclass

**vars-function** `vars()`: Local variables

**simple-statements** Simple statements

**assignment-statement** The assignment statement: *name = expression*

**assert-statement** The `assert` statement: Verify preconditions

**del-statement** The `del` statement: Delete a name or part of a value

**exec-statement** The `exec` statement: Execute Python source code

**global-statement** The `global` statement: Declare access to a global name

**import-statement** The `import` statement: Use a module

**pass-statement** The `pass` statement: Do nothing

**print-statement** The `print` statement: Display output values

**print-as-function** The `print()` function

**compound-statements** Compound statements

**blocks** Python's block structure

**break-statement** The `break` statement: Exit a `for` or `while` loop

**continue-statement** The `continue` statement: Jump to the next cycle of a `for` or `while`

**for-statement** The `for` statement: Iteration over a sequence

**if-statement** The `if` statement: Conditional execution

**raise-statement** The `raise` statement: Cause an exception

**return-statement** The `return` statement: Exit a function or method

**try-statement** The `try` statement: Anticipate exceptions

**with-statement** The `with` statement and context managers

**yield-statement** The `yield` statement: Generate one result from a generator

**def** `def()`: Defining your own functions

**function-locals** A function's local namespace

**iterators** Iterators: Values that can produce a sequence of values

**generators** Generators: Functions that can produce a sequence of values

**decorators** Decorators

**exceptions** Exceptions: Error signaling and handling

**exception-terms** Definitions of exception terms

**exception-lifecycle** Life cycle of an exception

**exception-hierarchy** Built-in exceptions

**classes** Classes: Defining your own types

**old-classes-intro** Old-style classes

**old-class-def** Defining an old-style class

**old-constructor** Instantiation of an old-style class: The constructor, `__init__()`

**old-attribute-ref** Attribute references in old-style classes

**old-method-call** Method calls in an old-style class

**old-del** Instance deletion: the destructor, `__del__()`

**new-classes-intro** Life cycle of a new-style class

**new-new-method** `__new__()`: New instance creation

**new-attr-access** Attribute access control in new-style classes

**new-property** Properties in new-style classes: Fine-grained attribute access control

**new-slots** Conserving memory with `__slots__`

**special-methods** Special method names

**rich-compare-methods** Rich comparison methods

**binary-operator-methods** Special methods for binary operators

**unary-operator-methods** Unary operator special methods

**builtin-function-methods** Special methods to emulate built-in functions

**call-method** `__call__()`: What to do when someone calls an instance

**cmp-method** `__cmp__()`: Generalized comparison

**contains-method** `__contains__()`: The "in" and "not in" operators

**del-method** `__del__()`: Destructor

**delattr-method** `__delattr__()`: Delete an attribute

**delitem-method** `__delitem__()`: Delete one item of a sequence

**enter-method** `__enter__`: Context manager initialization

**exit-method** `__exit__`: Context manager cleanup  
**format-method** `__format__`: Implement the `format()` function  
**getattr-method** `__getattr__()`: Handle a reference to an unknown attribute  
**getattribute-method** `__getattribute__()`: Intercept all attribute references  
**getitem-method** `__getitem__()`: Get one item from a sequence or mapping  
**iter-method** `__iter__()`: Create an iterator  
**nonzero-method** `__nonzero__()`: True/false evaluation  
**repr-method** `__repr__()`: String representation  
**reversed-method** `__reversed__()`: Implement the `reversed()` function  
**setattr-method** `__setattr__()`: Intercept all attribute changes  
**setitem-method** `__setitem__()`: Assign a value to one item of a sequence  
**static-methods** Static methods  
**class-methods** Class methods  
**pdb** `pdb`: The Python interactive debugger  
**pdb-startup** Starting up `pdb`  
**pdb-exports** Functions exported by `pdb`  
**pdb-commands** Commands available in `pdb`  
**common-modules** Commonly used modules  
**math-module** `math`: Common mathematical operations  
**string-module** `string`: Utility functions for strings  
**random-module** `random`: Random number generation  
**time-module** `time`: Clock and calendar functions  
**re-module** `re`: Regular expression pattern-matching  
**regex-chars** Characters in regular expressions  
**regex-functions** Functions in the `re` module  
**compiled-re** Compiled regular expression objects  
**match-object** Methods on a `MatchObject`  
**sys-module** `sys`: Universal system interface  
**os-module** `os`: The operating system interface  
**os-stat-module** `stat`: Interpretation of file status  
**os.path-module** `os.path`: File and directory interface  
**argparse** `argparse`: Processing command line arguments  
**argparse-defs** Types of command line arguments  
**argparse-flow** Overall flow of argument processing  
**argparse-init** The `ArgumentParser()` constructor  
**argparse-add\_argument** The `ArgumentParser.add_argument()` method  
**argparse-parse\_args** The `ArgumentParser.parse_args()` method  
**argparse-utilities** Other useful `ArgumentParser` methods