What's New in Python 2.1

Author: A.M. Kuchling

Introduction

This article explains the new features in Python 2.1. While there aren't as many changes in 2.1 as there were in Python 2.0, there are still some pleasant surprises in store. 2.1 is the first release to be steered through the use of Python Enhancement Proposals, or PEPs, so most of the sizable changes have accompanying PEPs that provide more complete documentation and a design rationale for the change. This article doesn't attempt to document the new features completely, but simply provides an overview of the new features for Python programmers. Refer to the Python 2.1 documentation, or to the specific PEP, for more details about any new feature that particularly interests you.

One recent goal of the Python development team has been to accelerate the pace of new releases, with a new release coming every 6 to 9 months. 2.1 is the first release to come out at this faster pace, with the first alpha appearing in January, 3 months after the final version of 2.0 was released.

The final release of Python 2.1 was made on April 17, 2001.

PEP 227: Nested Scopes

The largest change in Python 2.1 is to Python's scoping rules. In Python 2.0, at any given time there are at most three namespaces used to look up variable names: local, module-level, and the built-in namespace. This often surprised people because it didn't match their intuitive expectations. For example, a nested recursive function definition doesn't work:

```
def f():
    ...
    def g(value):
        ...
        return g(value-1) + 1
    ...
```

The function :func:`g` will always raise a :exc:`NameError` exception, because the binding of the name `g` isn't in either its local namespace or in the module-level namespace. This isn't much of a problem in practice (how often do you recursively define interior functions like this?), but this also made using the :keyword:`lambda` expression clumsier, and this was a problem in practice. In code which uses :keyword:`lambda` you can often find local variables being copied by passing them as the default values of arguments.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 51); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 51); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 51); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 51); *backlink*

Unknown interpreted text role "keyword".

```
def find(self, name):
    "Return list of any entries equal to 'name'"
    L = filter(lambda x, name=name: x == name,
               self.list_attribute)
    return L
```

The readability of Python code written in a strongly functional style suffers greatly as a result.

The most significant change to Python 2.1 is that static scoping has been added to the language to fix this problem. As a first effect, the `name=name` default argument is now unnecessary in the above example. Put simply, when a given variable name is not assigned a value within a function (by an assignment, or the :keyword:`def`, :keyword:`class`, or :keyword:`import` statements), references to the variable will be looked up in the local namespace of the enclosing scope. A more detailed explanation of the rules, and a dissection of the implementation, can be found in the PEP.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 68); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 68); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 68); *backlink*

Unknown interpreted text role "keyword".

This change may cause some compatibility problems for code where the same variable name is used both at the module level and as a local variable within a function that contains further function definitions. This seems rather unlikely though, since such code would have been pretty confusing to read in the first place.

One side effect of the change is that the `from module import *` and `exec` statements have been made illegal inside a function scope under certain conditions. The Python reference manual has said all along that `from module import *` is only legal at the top level of a module, but the CPython interpreter has never enforced this before. As part of the implementation of nested scopes, the compiler which turns Python source into bytecodes has to generate different code to access variables in a containing scope. `from module import *` and `exec` make it impossible for the compiler to figure this out, because they add names to the local namespace that are unknowable at compile time. Therefore, if a function contains function definitions or :keyword:`lambda` expressions with free variables, the compiler will flag this by raising a :exc:`SyntaxError` exception.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 83); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 83); *backlink*

Unknown interpreted text role "exc".

To make the preceding explanation a bit clearer, here's an example:

```
x = 1
def f():
    # The next line is a syntax error
    exec 'x=2'
    def g():
        return x
```

Line 4 containing the `exec` statement is a syntax error, since `exec` would define a new local variable named `x` whose value should be accessed by :func:`g`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 105); *backlink*

Unknown interpreted text role "func".

This shouldn't be much of a limitation, since `exec` is rarely used in most Python code (and when it is used, it's often a sign of a poor design anyway).

Compatibility concerns have led to nested scopes being introduced gradually; in Python 2.1, they aren't enabled by default, but can be turned on within a module by using a future statement as described in PEP 236. (See the following section for further discussion of PEP 236.) In Python 2.2, nested scopes will become the default and there will be no way to turn them off, but users will have had all of 2.1's lifetime to fix any breakage resulting from their introduction.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 121)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`227` - Statically Nested Scopes
      Written and implemented by Jeremy Hylton.
```

PEP 236: __future__ Directives

The reaction to nested scopes was widespread concern about the dangers of breaking code with the 2.1 release, and it was strong enough to make the Pythoneers take a more conservative approach. This approach consists of introducing a convention for enabling optional functionality in release N that will become compulsory in release N+1.

The syntax uses a `from...import` statement using the reserved module name :mod:`__future__`. Nested scopes can be enabled by the following statement:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 138); *backlink*

Unknown interpreted text role "mod".

```
from __future__ import nested_scopes
```

While it looks like a normal :keyword:`import` statement, it's not; there are strict rules on where such a future statement can be put. They can only be at the top of a module, and must precede any Python code or regular :keyword:`!import` statements. This is because such statements can affect how the Python bytecode compiler parses code and generates bytecode, so they must precede any statement that will result in bytecodes being produced.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 143); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 143); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 151)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`236` - Back to the :mod:`__future__`
      Written by Tim Peters, and primarily implemented by Jeremy Hylton.
```

PEP 207: Rich Comparisons

In earlier versions, Python's support for implementing comparisons on user-defined classes and extension types was quite simple. Classes could implement a :meth:`__cmp__` method that was given two instances of a class, and could only return 0 if they were equal or +1 or -1 if they weren't; the method couldn't raise an exception or return anything other than a Boolean value. Users of Numeric Python often found this model too weak and restrictive, because in the number-crunching programs that numeric Python is used for, it would be more useful to be able to perform elementwise comparisons of two matrices, returning a matrix containing the results of a given comparison for each element. If the two matrices are of different sizes, then the compare has to be able to raise an exception to signal the error.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 162); *backlink*

Unknown interpreted text role "meth".

In Python 2.1, rich comparisons were added in order to support this need. Python classes can now individually overload each of the <, <=, >, >=, ==, and `!=` operations. The new magic method names are:

Operation	Method name

Operation	Method name
`<`	:meth:`__lt__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 182**); ***backlink*** Unknown interpreted text role "meth".
`<=`	:meth:`__le__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 184**); ***backlink*** Unknown interpreted text role "meth".
`>`	:meth:`__gt__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 186**); ***backlink*** Unknown interpreted text role "meth".
`>=`	:meth:`__ge__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 188**); ***backlink*** Unknown interpreted text role "meth".
`==`	:meth:`__eq__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 190**); ***backlink*** Unknown interpreted text role "meth".
`!=`	:meth:`__ne__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 192**); ***backlink*** Unknown interpreted text role "meth".

(The magic methods are named after the corresponding Fortran operators `.LT.`. `.LE.`, &c. Numeric programmers are almost certainly quite familiar with these names and will find them easy to remember.)

Each of these magic methods is of the form `method(self, other)`, where `self` will be the object on the left-hand side of the operator, while `other` will be the object on the right-hand side. For example, the expression `A < B` will cause `A.__lt__(B)` to be called.

Each of these magic methods can return anything at all: a Boolean, a matrix, a list, or any other Python object. Alternatively they can raise an exception if the comparison is impossible, inconsistent, or otherwise meaningless.

The built-in `cmp(A,B)` function can use the rich comparison machinery, and now accepts an optional argument specifying which comparison operation to use; this is given as one of the strings `"<"`, `"<="`, `">"`, `">="`, `"=="`, or `"!="`. If called without the optional third argument, :func:`cmp` will only return -1, 0, or +1 as in previous versions of Python; otherwise it will call the appropriate method and can return any Python object.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 207); *backlink*

Unknown interpreted text role "func".

There are also corresponding changes of interest to C programmers; there's a new slot `tp_richcmp` in type objects and an API for performing a given rich comparison. I won't cover the C API here, but will refer you to PEP 207, or to 2.1's C API documentation, for the full list of related functions.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 220)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`207` - Rich Comparisons
      Written by Guido van Rossum, heavily based on earlier work by David Ascher, and
      implemented by Guido van Rossum.
```

PEP 230: Warning Framework

Over its 10 years of existence, Python has accumulated a certain number of obsolete modules and features along the way. It's difficult to know when a feature is safe to remove, since there's no way of knowing how much code uses it --- perhaps no programs depend on the feature, or perhaps many do. To enable removing old features in a more structured way, a warning framework was added. When the Python developers want to get rid of a feature, it will first trigger a warning in the next version of Python. The following Python version can then drop the feature, and users will have had a full release cycle to remove uses of the old feature.

Python 2.1 adds the warning framework to be used in this scheme. It adds a :mod:`warnings` module that provide functions to issue warnings, and to filter out warnings that you don't want to be displayed. Third-party modules can also use this framework to deprecate old features that they no longer wish to support.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 242); *backlink*

Unknown interpreted text role "mod".

For example, in Python 2.1 the :mod:`regex` module is deprecated, so importing it causes a warning to be printed:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 248); *backlink*

Unknown interpreted text role "mod".

```
>>> import regex
__main__:1: DeprecationWarning: the regex module
         is deprecated; please use the re module
>>>
```

Warnings can be issued by calling the :func:`warnings.warn` function:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 256); *backlink*

Unknown interpreted text role "func".

```
warnings.warn("feature X no longer supported")
```

The first parameter is the warning message; an additional optional parameters can be used to specify a particular warning category.

Filters can be added to disable certain warnings; a regular expression pattern can be applied to the message or to the module name in order to suppress a warning. For example, you may have a program that uses the :mod:`regex` module and not want to spare the time to convert it to use the :mod:`re` module right now. The warning can be suppressed by calling

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 263); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 263); *backlink*

Unknown interpreted text role "mod".

```
import warnings
warnings.filterwarnings(action = 'ignore',
                        message='.*regex module is deprecated',
                        category=DeprecationWarning,
                        module = '__main__')
```

This adds a filter that will apply only to warnings of the class :class:`DeprecationWarning` triggered in the :mod:`__main__` module, and applies a regular expression to only match the message about the :mod:`regex` module being deprecated, and will cause such warnings to be ignored. Warnings can also be printed only once, printed every time the offending code is executed, or turned into exceptions that will cause the program to stop (unless the exceptions are caught in the usual way, of course).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 275); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 275); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 275); *backlink*

Unknown interpreted text role "mod".

Functions were also added to Python's C API for issuing warnings; refer to PEP 230 or to Python's API documentation for the details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 287)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`5` - Guidelines for Language Evolution
      Written by Paul Prescod, to specify procedures to be followed when removing old
      features from Python.  The policy described in this PEP hasn't been officially
      adopted, but the eventual policy probably won't be too different from Prescod's
      proposal.

   :pep:`230` - Warning Framework
      Written and implemented by Guido van Rossum.
```

PEP 229: New Build System

When compiling Python, the user had to go in and edit the :file:`Modules/Setup` file in order to enable various additional modules; the default set is relatively small and limited to modules that compile on most Unix platforms. This means that on Unix platforms with many more features, most notably Linux, Python installations often don't contain all useful modules they could.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 304); *backlink*

Unknown interpreted text role "file".

Python 2.0 added the Distutils, a set of modules for distributing and installing extensions. In Python 2.1, the Distutils are used to compile much of the standard library of extension modules, autodetecting which ones are supported on the current machine. It's hoped that this will make Python installations easier and more featureful.

Instead of having to edit the :file:`Modules/Setup` file in order to enable modules, a :file:`setup.py` script in the top directory of the Python source distribution is run at build time, and attempts to discover which modules can be enabled by examining the modules and header files on the system. If a module is configured in :file:`Modules/Setup`, the :file:`setup.py` script won't attempt to compile that module and will defer to the :file:`Modules/Setup` file's contents. This provides a way to specific any strange command-line flags or libraries that are required for a specific platform.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 316); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 316); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 316); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 316); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 316); *backlink*

Unknown interpreted text role "file".

In another far-reaching change to the build mechanism, Neil Schemenauer restructured things so Python now uses a single makefile that isn't recursive, instead of makefiles in the top directory and in each of the :file:`Python/`, :file:`Parser/`, :file:`Objects/`, and :file:`Modules/` subdirectories. This makes building Python faster and also makes hacking the Makefiles clearer and simpler.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 325); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 325); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 325); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 325); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 333)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`229` - Using Distutils to Build Python
      Written and implemented by A.M. Kuchling.
```

PEP 205: Weak References

Weak references, available through the :mod:`weakref` module, are a minor but useful new data type in the Python programmer's toolbox.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 344); *backlink*

Unknown interpreted text role "mod".

Storing a reference to an object (say, in a dictionary or a list) has the side effect of keeping that object alive forever. There are a few specific cases where this behaviour is undesirable, object caches being the most common one, and another being circular references in data structures such as trees.

For example, consider a memoizing function that caches the results of another function `f(x)` by storing the function's argument and its result in a dictionary:

```
_cache = {}
def memoize(x):
    if _cache.has_key(x):
        return _cache[x]

    retval = f(x)

    # Cache the returned object
    _cache[x] = retval

    return retval
```

This version works for simple things such as integers, but it has a side effect; the `_cache` dictionary holds a reference to the return values, so they'll never be deallocated until the Python process exits and cleans up. This isn't very noticeable for integers, but if :func:`f` returns an object, or a data structure that takes up a lot of memory, this can be a problem.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 368); *backlink*

Unknown interpreted text role "func".

Weak references provide a way to implement a cache that won't keep objects alive beyond their time. If an object is only accessible through weak references, the object will be deallocated and the weak references will now indicate that the object it referred to no longer exists. A weak reference to an object *obj* is created by calling `wr = weakref.ref(obj)`. The object being referred to is returned by calling the weak reference as if it were a function: `wr()`. It will return the referenced object, or `None` if the object no longer exists.

This makes it possible to write a :func:`memoize` function whose cache doesn't keep objects alive, by storing weak references in the cache.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 382); *backlink*

Unknown interpreted text role "func".

```
_cache = {}
def memoize(x):
    if _cache.has_key(x):
        obj = _cache[x]()
        # If weak reference object still exists,
        # return it
        if obj is not None: return obj

    retval = f(x)

    # Cache a weak reference
    _cache[x] = weakref.ref(retval)

    return retval
```

The :mod:`weakref` module also allows creating proxy objects which behave like weak references --- an object referenced only by proxy objects is deallocated -- but instead of requiring an explicit call to retrieve the object, the proxy transparently forwards all operations to the object as long as the object still exists. If the object is deallocated, attempting to use a proxy will cause a :exc:`weakref.ReferenceError` exception to be raised.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 400); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 400); *backlink*

Unknown interpreted text role "exc".

```
proxy = weakref.proxy(obj)
proxy.attr   # Equivalent to obj.attr
proxy.meth() # Equivalent to obj.meth()
del obj
proxy.attr   # raises weakref.ReferenceError
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 414)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`205` - Weak References
      Written and implemented by Fred L. Drake, Jr.
```

PEP 232: Function Attributes

In Python 2.1, functions can now have arbitrary information attached to them. People were often using docstrings to hold information about functions and methods, because the `__doc__` attribute was the only way of attaching any information to a function. For example, in the Zope web application server, functions are marked as safe for public access by having a docstring, and in John Aycock's SPARK parsing framework, docstrings hold parts of the BNF grammar to be parsed. This overloading is unfortunate, since docstrings are really intended to hold a function's documentation; for example, it means you can't properly document functions intended for private use in Zope.

Arbitrary attributes can now be set and retrieved on functions using the regular Python syntax:

```
def f(): pass

f.publish = 1
f.secure = 1
f.grammar = "A ::= B (C D)*"
```

The dictionary containing attributes can be accessed as the function's :attr:`~object.__dict__`. Unlike the :attr:`~object.__dict__` attribute of class instances, in functions you can actually assign a new dictionary to :attr:`~object.__dict__`, though the new value is restricted to a regular Python dictionary; you *can't* be tricky and set it to a :class:`UserDict` instance, or any other random object that behaves like a mapping.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 444); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 444); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 444); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 444); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 452)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`232` - Function Attributes
      Written and implemented by Barry Warsaw.
```

PEP 235: Importing Modules on Case-Insensitive Platforms

Some operating systems have filesystems that are case-insensitive, MacOS and Windows being the primary examples; on these systems, it's impossible to distinguish the filenames `FILE.PY` and `file.py`, even though they do store the file's name in its original case (they're case-preserving, too).

In Python 2.1, the :keyword:`import` statement will work to simulate case-sensitivity on case-insensitive platforms. Python will now search for the first case-sensitive match by default, raising an :exc:`ImportError` if no such file is found, so `import file` will not import a module named `FILE.PY`. Case-insensitive matching can be requested by setting the :envvar:`PYTHONCASEOK` environment variable before starting the Python interpreter.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 468); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 468); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 468); *backlink*

Unknown interpreted text role "envvar".

PEP 217: Interactive Display Hook

When using the Python interpreter interactively, the output of commands is displayed using the built-in :func:`repr` function. In Python 2.1, the variable :func:`sys.displayhook` can be set to a callable object which will be called instead of :func:`repr`. For example, you can set it to a special pretty-printing function:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 481); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 481); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 481); *backlink*

Unknown interpreted text role "func".

```
>>> # Create a recursive data structure
... L = [1,2,3]
>>> L.append(L)
>>> L # Show Python's default output
[1, 2, 3, [...]]
>>> # Use pprint.pprint() as the display function
... import sys, pprint
>>> sys.displayhook = pprint.pprint
>>> L
[1, 2, 3,  <Recursion on list with id=135143996>]
```

```
>>>
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 500)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`217` - Display Hook for Interactive Use
      Written and implemented by Moshe Zadka.
```

PEP 208: New Coercion Model

How numeric coercion is done at the C level was significantly modified. This will only affect the authors of C extensions to Python, allowing them more flexibility in writing extension types that support numeric operations.

Extension types can now set the type flag `Py_TPFLAGS_CHECKTYPES` in their `PyTypeObject` structure to indicate that they support the new coercion model. In such extension types, the numeric slot functions can no longer assume that they'll be passed two arguments of the same type; instead they may be passed two arguments of differing types, and can then perform their own internal coercion. If the slot function is passed a type it can't handle, it can indicate the failure by returning a reference to the `Py_NotImplemented` singleton value. The numeric functions of the other type will then be tried, and perhaps they can handle the operation; if the other type also returns `Py_NotImplemented`, then a :exc:`TypeError` will be raised. Numeric methods written in Python can also return `Py_NotImplemented`, causing the interpreter to act as if the method did not exist (perhaps raising a :exc:`TypeError`, perhaps trying another object's numeric methods).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 515); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 515); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 530)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`208` - Reworking the Coercion Model
      Written and implemented by Neil Schemenauer, heavily based upon earlier work by
      Marc-AndrÃ© Lemburg.  Read this to understand the fine points of how numeric
      operations will now be processed at the C level.
```

PEP 241: Metadata in Python Packages

A common complaint from Python users is that there's no single catalog of all the Python modules in existence. T. Middleton's Vaults of Parnassus at http://www.vex.net/parnassus/ are the largest catalog of Python modules, but registering software at the Vaults is optional, and many people don't bother.

As a first small step toward fixing the problem, Python software packaged using the Distutils :command:`sdist` command will include a file named :file:`PKG-INFO` containing information about the package such as its name, version, and author (metadata, in cataloguing terminology). PEP 241 contains the full list of fields that can be present in the :file:`PKG-INFO` file. As people began to package their software using Python 2.1, more and more packages will include metadata, making it possible to build automated cataloguing systems and experiment with them. With the result experience, perhaps it'll be possible to design a really good catalog and then build support for it into Python 2.2. For example, the Distutils :command:`sdist` and :command:`bdist_\*` commands could support an `upload` option that would automatically upload your package to a catalog server.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 548); *backlink*

Unknown interpreted text role "command".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 548);** ***backlink***

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 548);** ***backlink***

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 548);** ***backlink***

Unknown interpreted text role "command".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 548);** ***backlink***

Unknown interpreted text role "command".

You can start creating packages containing :file:`PKG-INFO` even if you're not using Python 2.1, since a new release of the Distutils will be made for users of earlier Python versions. Version 1.0.2 of the Distutils includes the changes described in PEP 241, as well as various bugfixes and enhancements. It will be available from the Distutils SIG at https://www.python.org/community/sigs/current/distutils-sig/.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 561);** ***backlink***

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 568)**

Unknown directive type "seealso".

```
.. seealso::

   :pep:`241` - Metadata for Python Software Packages
      Written and implemented by A.M. Kuchling.

   :pep:`243` - Module Repository Upload Mechanism
      Written by Sean Reifschneider, this draft PEP describes a proposed mechanism for
      uploading  Python packages to a central server.
```

New and Improved Modules

- Ka-Ping Yee contributed two new modules: :mod:`inspect.py`, a module for getting information about live Python code, and :mod:`pydoc.py`, a module for interactively converting docstrings to HTML or text. As a bonus, :file:`Tools/scripts/pydoc`, which is now automatically installed, uses :mod:`pydoc.py` to display documentation given a Python module, package, or class name. For example, `pydoc xml.dom` displays the following:

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 583);** ***backlink***

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 583);** ***backlink***

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 583);**

backlink

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 583); *backlink*

Unknown interpreted text role "mod".

```
Python Library Documentation: package xml.dom in xml

NAME
    xml.dom - W3C Document Object Model implementation for Python.

FILE
    /usr/local/lib/python2.1/xml/dom/__init__.pyc

DESCRIPTION
    The Python mapping of the Document Object Model is documented in the
    Python Library Reference in the section on the xml.dom package.

    This package contains the following modules:
      ...
```

:file:`pydoc` also includes a Tk-based interactive help browser. :file:`pydoc` quickly becomes addictive; try it out!

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 605); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 605); *backlink*

Unknown interpreted text role "file".

- Two different modules for unit testing were added to the standard library. The :mod:`doctest` module, contributed by Tim Peters, provides a testing framework based on running embedded examples in docstrings and comparing the results against the expected output. PyUnit, contributed by Steve Purcell, is a unit testing framework inspired by JUnit, which was in turn an adaptation of Kent Beck's Smalltalk testing framework. See http://pyunit.sourceforge.net/ for more information about PyUnit.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 608); *backlink*

  Unknown interpreted text role "mod".

- The :mod:`difflib` module contains a class, :class:`SequenceMatcher`, which compares two sequences and computes the changes required to transform one sequence into the other. For example, this module can be used to write a tool similar to the Unix :program:`diff` program, and in fact the sample program :file:`Tools/scripts/ndiff.py` demonstrates how to write such a script.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 616); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 616); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-`

resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 616); *backlink*

Unknown interpreted text role "program".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 616); *backlink*

Unknown interpreted text role "file".

- :mod:`curses.panel`, a wrapper for the panel library, part of ncurses and of SYSV curses, was contributed by Thomas Gellekum. The panel library provides windows with the additional feature of depth. Windows can be moved higher or lower in the depth ordering, and the panel library figures out where panels overlap and which sections are visible.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 622); *backlink*

  Unknown interpreted text role "mod".

- The PyXML package has gone through a few releases since Python 2.0, and Python 2.1 includes an updated version of the :mod:`xml` package. Some of the noteworthy changes include support for Expat 1.2 and later versions, the ability for Expat parsers to handle files in any encoding supported by Python, and various bugfixes for SAX, DOM, and the :mod:`minidom` module.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 628); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 628); *backlink*

  Unknown interpreted text role "mod".

- Ping also contributed another hook for handling uncaught exceptions. :func:`sys.excepthook` can be set to a callable object. When an exception isn't caught by any :keyword:`try`...:keyword:`except` blocks, the exception will be passed to :func:`sys.excepthook`, which can then do whatever it likes. At the Ninth Python Conference, Ping demonstrated an application for this hook: printing an extended traceback that not only lists the stack frames, but also lists the function arguments and the local variables for each frame.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 634); *backlink*

  Unknown interpreted text role "func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 634); *backlink*

  Unknown interpreted text role "keyword".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 634); *backlink*

  Unknown interpreted text role "keyword".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst, line 634); *backlink*

Unknown interpreted text role "func".

- Various functions in the :mod:`time` module, such as :func:`asctime` and :func:`localtime`, require a floating point argument containing the time in seconds since the epoch. The most common use of these functions is to work with the current time, so the floating point argument has been made optional; when a value isn't provided, the current time will be used. For example, log file entries usually need a string containing the current time; in Python 2.1, `time.asctime()` can be used, instead of the lengthier `time.asctime(time.localtime(time.time()))` that was previously required.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 642); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 642); *backlink*

  Unknown interpreted text role "func".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 642); *backlink*

  Unknown interpreted text role "func".

  This change was proposed and implemented by Thomas Wouters.

- The :mod:`ftplib` module now defaults to retrieving files in passive mode, because passive mode is more likely to work from behind a firewall. This request came from the Debian bug tracking system, since other Debian packages use :mod:`ftplib` to retrieve files and then don't work from behind a firewall. It's deemed unlikely that this will cause problems for anyone, because Netscape defaults to passive mode and few people complain, but if passive mode is unsuitable for your application or network setup, call `set_pasv(0)` on FTP objects to disable passive mode.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 653); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 653); *backlink*

  Unknown interpreted text role "mod".

- Support for raw socket access has been added to the :mod:`socket` module, contributed by Grant Edwards.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 662); *backlink*

  Unknown interpreted text role "mod".

- The :mod:`pstats` module now contains a simple interactive statistics browser for displaying timing profiles for Python programs, invoked when the module is run as a script. Contributed by Eric S. Raymond.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 665); *backlink*

  Unknown interpreted text role "mod".

- A new implementation-dependent function, `sys._getframe([depth])`, has been added to return a given frame object from the current call stack. :func:`sys._getframe` returns the frame at the top of the call stack; if the optional integer argument *depth* is supplied, the function returns the frame that is *depth* calls below the top of the stack. For example, `sys._getframe(1)` returns the caller's frame object.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 669**); ***backlink***

Unknown interpreted text role "func".

This function is only present in CPython, not in Jython or the .NET implementation. Use it for debugging, and resist the temptation to put it into production code.

Other Changes and Fixes

There were relatively few smaller changes made in Python 2.1 due to the shorter release cycle. A search through the CVS change logs turns up 117 patches applied, and 136 bugs fixed; both figures are likely to be underestimates. Some of the more notable changes are:

- A specialized object allocator is now optionally available, that should be faster than the system :func:`malloc` and have less memory overhead. The allocator uses C's :func:`malloc` function to get large pools of memory, and then fulfills smaller memory requests from these pools. It can be enabled by providing the :option:`!--with-pymalloc` option to the :program:`configure` script; see :file:`Objects/obmalloc.c` for the implementation details.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 691**); ***backlink***

  Unknown interpreted text role "func".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 691**); ***backlink***

  Unknown interpreted text role "func".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 691**); ***backlink***

  Unknown interpreted text role "option".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 691**); ***backlink***

  Unknown interpreted text role "program".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 691**); ***backlink***

  Unknown interpreted text role "file".

  Authors of C extension modules should test their code with the object allocator enabled, because some incorrect code may break, causing core dumps at runtime. There are a bunch of memory allocation functions in Python's C API that have previously been just aliases for the C library's :func:`malloc` and :func:`free`, meaning that if you accidentally called mismatched functions, the error wouldn't be noticeable. When the object allocator is enabled, these functions aren't aliases of :func:`malloc` and :func:`free` any more, and calling the wrong function to free memory will get you a core dump. For example, if memory was allocated using :func:`PyMem_New`, it has to be freed using :func:`PyMem_Del`, not :func:`free`. A few modules included with Python fell afoul of this and had to be fixed; doubtless there are more third-party modules that will have the same problem.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, **line 698**); ***backlink***

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 698); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 698); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 698); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 698); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 698); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 698); *backlink*

Unknown interpreted text role "func".

The object allocator was contributed by Vladimir Marangozov.

- The speed of line-oriented file I/O has been improved because people often complain about its lack of speed, and because it's often been used as a naÃ¯ve benchmark. The :meth:`readline` method of file objects has therefore been rewritten to be much faster. The exact amount of the speedup will vary from platform to platform depending on how slow the C library's :func:`getc` was, but is around 66%, and potentially much faster on some particular operating systems. Tim Peters did much of the benchmarking and coding for this change, motivated by a discussion in comp.lang.python.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 713); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main) (Doc) (whatsnew)2.1.rst`, line 713); *backlink*

Unknown interpreted text role "func".

A new module and method for file objects was also added, contributed by Jeff Epler. The new method, :meth:`xreadlines`, is similar to the existing :func:`xrange` built-in. :func:`xreadlines` returns an opaque sequence object that only supports being iterated over, reading a line on every iteration but not reading the entire file into memory as the existing :meth:`readlines` method does. You'd use it like this:

System Message: ERROR/3 (`D:\onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst, line 722); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst, line 722); ***backlink***

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst, line 722); ***backlink***

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst, line 722); ***backlink***

Unknown interpreted text role "meth".

```
for line in sys.stdin.xreadlines():
    # ... do something for each line ...
    ...
```

For a fuller discussion of the line I/O changes, see the python-dev summary for January 1--15, 2001 at https://mail.python.org/pipermail/python-dev/2001-January/.

- A new method, :meth:`popitem`, was added to dictionaries to enable destructively iterating through the contents of a dictionary; this can be faster for large dictionaries because there's no need to construct a list containing all the keys or values. `D.popitem()` removes a random `(key, value)` pair from the dictionary `D` and returns it as a 2-tuple. This was implemented mostly by Tim Peters and Guido van Rossum, after a suggestion and preliminary patch by Moshe Zadka.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst, line 736); ***backlink***

  Unknown interpreted text role "meth".

- Modules can now control which names are imported when `from module import *` is used, by defining an `__all__` attribute containing a list of names that will be imported. One common complaint is that if the module imports other modules such as :mod:`sys` or :mod:`string`, `from module import *` will add them to the importing module's namespace. To fix this, simply list the public names in `__all__`:

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst, line 744); ***backlink***

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst, line 744); ***backlink***

  Unknown interpreted text role "mod".

  ```
  # List public names
  __all__ = ['Database', 'open']
  ```

  A stricter version of this patch was first suggested and implemented by Ben Wolfson, but after some python-dev discussion, a weaker final version was checked in.

- Applying :func:`repr` to strings previously used octal escapes for non-printable characters; for example, a newline was `'\012'`. This was a vestigial trace of Python's C ancestry, but today octal is of very little practical use. Ka-Ping Yee suggested using hex escapes instead of octal ones, and using the `\n`, `\t`, `\r` escapes for the appropriate characters, and implemented this new formatting.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 758);** ***backlink***

Unknown interpreted text role "func".

- Syntax errors detected at compile-time can now raise exceptions containing the filename and line number of the error, a pleasant side effect of the compiler reorganization done by Jeremy Hylton.
- C extensions which import other modules have been changed to use :func:`PyImport_ImportModule`, which means that they will use any import hooks that have been installed. This is also encouraged for third-party extensions that need to import some other module from C code.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\whatsnew\(cpython-main)(Doc)(whatsnew)2.1.rst`, **line 769);** ***backlink***

Unknown interpreted text role "func".

- The size of the Unicode character database was shrunk by another 340K thanks to Fredrik Lundh.
- Some new ports were contributed: MacOS X (by Steven Majewski), Cygwin (by Jason Tishler); RISCOS (by Dietmar Schwertberger); Unixware 7 (by Billy G. Allie).

And there's the usual list of minor bugfixes, minor memory leaks, docstring edits, and other tweaks, too lengthy to be worth itemizing; see the CVS logs for the full details if you want them.

Acknowledgements

The author would like to thank the following people for offering suggestions on various drafts of this article: Graeme Cross, David Goodger, Jay Graves, Michael Hudson, Marc-AndrÃ© Lemburg, Fredrik Lundh, Neil Schemenauer, Thomas Wouters.