cpython

edj

Table of Contents

Lib/test/test email/data/msg 11
Lib/test/test email/data/msg 12
Lib/test/test email/data/msg 12a
Lib/test/test email/data/msg 13
Lib/test/test email/data/msg 14
Lib/test/test email/data/msg 15
Lib/test/test email/data/msg 16
Lib/test/test email/data/msg 17
Lib/test/test email/data/msg 18
Lib/test/test email/data/msg 19
Lib/test/test email/data/msg 20
Lib/test/test email/data/msg 21
Lib/test/test email/data/msg 22
Lib/test/test email/data/msg 23
Lib/test/test email/data/msg 24
Lib/test/test email/data/msg 25
Lib/test/test email/data/msg 26
Lib/test/test email/data/msg 27
Lib/test/test email/data/msg 28
Lib/test/test email/data/msg 29
Lib/test/test email/data/msg 30
Lib/test/test email/data/msg 31
Lib/test/test email/data/msg 32
Lib/test/test email/data/msg 33
Lib/test/test email/data/msg 34
Lib/test/test email/data/msg 35
Lib/test/test email/data/msg 36
Lib/test/test email/data/msg 37
Lib/test/test email/data/msg 38
Lib/test/test email/data/msg 39
Lib/test/test email/data/msg 40
Lib/test/test email/data/msg 41
Lib/test/test email/data/msg 42
Lib/test/test email/data/msg 43
Lib/test/test email/data/msg 44
Lib/test/test email/data/msg 45
Lib/test/test email/data/msg 46
Lib/test/tokenize tests-latin1-coding-cookie-and-utf8-bom-sig
Lib/test/tokenize tests-no-coding-cookie-and-utf8-bom-sig-only
Lib/test/tokenize tests-utf8-coding-cookie-and-no-utf8-bom-sig
Lib/test/tokenize tests-utf8-coding-cookie-and-utf8-bom-sig
Lib/test/tokenize tests
Mac/BuildScript/README
Mac/Icons/ReadMe
Misc/SpecialBuilds
Misc/svnmap
Modules/ decimal/libmpdec/literature/bignum
Modules/ decimal/libmpdec/literature/matrix-transform
Modules/ decimal/libmpdec/literature/mulmod-64
Modules/ decimal/libmpdec/literature/mulmod-ppro
Modules/ decimal/libmpdec/literature/REFERENCES
Modules/ decimal/libmpdec/literature/six-step
Modules/ decimal/libmpdec/README
Modules/ decimal/README
Modules/ decimal/tests/README
Modules/gc weakref
Modules/zlib/algorithm
Objects/dictnotes
Objects/lnotab notes

Objects/stringlib/README
PC/bdist wininst/README
PC/dllbase nt
PC/example nt/readme
PC/readme
PCbuild/readme
Tools/msi/bundle/bootstrap/LICENSE
Tools/msi/crt/crtlicense
Tools/msi/exe/crtlicense
Tools/msi/README
Tools/pynche/html40colors
Tools/pynche/namedcolors
Tools/pynche/webcolors
Tools/pynche/websafe
Tools/pynche/X/rgb
Tools/pynche/X/xlicense
Tools/unittestgui/README

Python Documentation README ~

This directory contains the reStructuredText (reST) sources to the Python documentation. You don't need to build them yourself, prebuilt versions are available at https://docs.python.org/dev/download.html.

Documentation on authoring Python documentation, including information about both style and markup, is available in the "Documenting Python" chapter of the developers guide https://docs.python.org/devguide/documenting.html.

Building the docs

You need to have Sphinx http://sphinx-doc.org/ installed; it is the toolset used to build the docs. It is not included in this tree, but maintained separately and available from PyPI https://pypi.python.org/pypi/Sphinx.

Using make

A Makefile has been prepared so that on Unix, provided you have installed Sphinx, you can just run ::

make html

to build the HTML output files.

On Windows, we try to emulate the Makefile as closely as possible with a `make.bat` file.

To use a Python interpreter that's not called `python`, use the standard way to set Makefile variables, using e.g. ::

make html PYTHON=python3

On Windows, set the PYTHON environment variable instead.

To use a specific sphinx-build (something other than `sphinx-build`), set the SPHINXBUILD variable.

Available make targets are:

- "clean", which removes all build files.
- "html", which builds standalone HTML files for offline viewing.
- "htmlview", which re-uses the "html" builder, but then opens the main page in your default web browser.
- "htmlhelp", which builds HTML files and a HTML Help project file usable to convert them into a single Compiled HTML (.chm) file -- these are popular under Microsoft Windows, but very handy on every platform.

To create the CHM file, you need to run the Microsoft HTML Help Workshop over the generated project (.hhp) file. The make.bat script does this for you on Windows.

- "latex", which builds LaTeX source files as input to "pdflatex" to produce PDF documents.
- "text", which builds a plain text file for each source file.
- "epub", which builds an EPUB document, suitable to be viewed on e-book readers.
- "linkcheck", which checks all external references to see whether they are broken, redirected or malformed, and outputs this information to stdout as well as a plain-text (.txt) file.
- "changes", which builds an overview over all versionadded/versionchanged/ deprecated items in the current version. This is meant as a help for the writer of the "What's New" document.
- "coverage", which builds a coverage overview for standard library modules and C API.
- "pydoc-topics", which builds a Python module containing a dictionary with plain text documentation for the labels defined in `tools/pyspecific.py` -- pydoc needs these to show topic and keyword help.
- "suspicious", which checks the parsed markup for text that looks like malformed and thus unconverted reST.
- "check", which checks for frequent markup errors.
- "serve", which serves the build/html directory on port 8000.

- "dist", (Unix only) which creates distributable archives of HTML, text, PDF, and EPUB builds.

Without make

Install the Sphinx package and its dependencies from PyPI.

Then, from the `Doc` directory, run ::

sphinx-build -b . build/

where `<builder>` is one of html, text, latex, or htmlhelp (for explanations see the make targets above).

Contributing

Bugs in the content should be reported to the Python bug tracker at https://bugs.python.org.

Bugs in the toolset should be reported in the Sphinx bug tracker at https://www.bitbucket.org/birkenfeld/sphinx/issues/.

You can also send a mail to the Python Documentation Team at docs@python.org, and we will process your request as soon as possible.

If you want to help the Documentation Team, you are always welcome. Just send a mail to docs@python.org.

Copyright notice

The Python source is copyrighted, but you can freely use and copy it as long as you don't change or remove the copyright notice:

Writing an IDLE extension

An IDLE extension can define new key bindings and menu entries for IDLE edit windows. There is a simple mechanism to load extensions when IDLE starts up and to attach them to each edit window. (It is also possible to make other changes to IDLE, but this must be done by editing the IDLE source code.)

The list of extensions loaded at startup time is configured by editing the file config-extensions.def. See below for details.

An IDLE extension is defined by a class. Methods of the class define actions that are invoked by event bindings or menu entries. Class (or instance) variables define the bindings and menu additions; these are automatically applied by IDLE when the extension is linked to an edit window.

An IDLE extension class is instantiated with a single argument, `editwin', an EditorWindow instance. The extension cannot assume much about this argument, but it is guaranteed to have the following instance variables:

```
text    a Text instance (a widget)
io      an IOBinding instance (more about this later)
flist   the FileList instance (shared by all edit windows)
```

(There are a few more, but they are rarely useful.)

The extension class must not directly bind Window Manager (e.g. X) events. Rather, it must define one or more virtual events, e.g. <<zoom-height>>, and corresponding methods, e.g. zoom_height_event(). The virtual events will be bound to the corresponding methods, and Window Manager events can then be bound to the virtual events. (This indirection is done so that the key bindings can easily be changed, and so that other sources of virtual events can exist, such as menu entries.)

An extension can define menu entries. This is done with a class or instance variable named menudefs; it should be a list of pairs, where each pair is a menu name (lowercase) and a list of menu entries. Each menu entry is either None (to insert a separator entry) or a pair of strings (menu_label, virtual_event). Here, menu_label is the label of the menu entry, and virtual_event is the virtual event to be generated when the entry is selected. An underscore in the menu label is removed; the character following the underscore is displayed underlined, to indicate the shortcut character (for Windows).

At the moment, extensions cannot define whole new menus; they must define entries in existing menus. Some menus are not present on some windows; such entry definitions are then ignored, but key bindings are still applied. (This should probably be refined in the future.)

Extensions are not required to define menu entries for all the events they implement. (They are also not required to create keybindings, but in that case there must be empty bindings in cofig-extensions.def)

Here is a complete example:

class ZoomHeight:

```
menudefs = [
    ('edit', [
        None, # Separator
        ('_Zoom Height', '<<zoom-height>>'),
     ])
]

def __init__(self, editwin):
    self.editwin = editwin

def zoom_height_event(self, event):
    "...Do what you want here..."
```

The final piece of the puzzle is the file "config-extensions.def", which is used to configure the loading of extensions and to establish key (or, more generally, event) bindings to the virtual events defined in the extensions.

See the comments at the top of config-extensions.def for information. It's currently necessary to manually modify that file to change IDLE's extension loading or extension key bindings.

For further information on binding refer to the Tkinter Resources web page at python.org and to the Tk Command "bind" man page.

[See the end of this file for ** TIPS ** on using IDLE !!]

IDLE is the Python IDE built with the tkinter GUI toolkit.

IDLE has the following features: -coded in 100% pure Python, using the tkinter GUI toolkit -cross-platform: works on Windows, Unix, and OS X -multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, and many other features -Python shell window (a.k.a interactive interpreter) -debugger (not complete, but you can set breakpoints, view and step)

Menus:

IDLE has two window types the Shell window and the Editor window. It is possible to have multiple editor windows simultaneously. IDLE's menus dynamically change based on which window is currently selected. Each menu documented below indicates which window type it is associated with. Click on the dotted line at the top of a menu to "tear it off": a separate window containing the menu is created (for Unix and Windows only).

File Menu (Shell and Editor):

```
New File         -- Create a new file editing window
Open...          -- Open an existing file
Open Module...   -- Open an existing module (searches sys.path)
Recent Files...  -- Open a list of recent files
Class Browser    -- Show classes and methods in current file
Path Browser     -- Show sys.path directories, modules, classes,
                    and methods
---
Save             -- Save current window to the associated file (unsaved
                    windows have a * before and after the window title)

Save As...       -- Save current window to new file, which becomes
                    the associated file
Save Copy As...  -- Save current window to different file
                    without changing the associated file
---
Print Window     -- Print the current window
---
Close            -- Close current window (asks to save if unsaved)
Exit             -- Close all windows, quit (asks to save if unsaved)
```

Edit Menu (Shell and Editor):

```
Undo             -- Undo last change to current window
                    (a maximum of 1000 changes may be undone)
Redo             -- Redo last undone change to current window
---
Cut              -- Copy a selection into system-wide clipboard,
                    then delete the selection
Copy             -- Copy selection into system-wide clipboard
Paste            -- Insert system-wide clipboard into window
Select All       -- Select the entire contents of the edit buffer
---
Find...          -- Open a search dialog box with many options
Find Again       -- Repeat last search
Find Selection   -- Search for the string in the selection
Find in Files... -- Open a search dialog box for searching files
Replace...       -- Open a search-and-replace dialog box
Go to Line       -- Ask for a line number and show that line
Expand Word      -- Expand the word you have typed to match another
```

```
                      word in the same buffer; repeat to get a
                      different expansion
    Show Calltip     -- After an unclosed parenthesis for a function, open
                      a small window with function parameter hints
    Show Parens      -- Highlight the surrounding parenthesis
    Show Completions -- Open a scroll window allowing selection keywords
                      and attributes. (see '*TIPS*', below)
```

Format Menu (Editor window only):

```
    Indent Region       -- Shift selected lines right by the indent width
                           (default 4 spaces)
    Dedent Region       -- Shift selected lines left by the indent width
                           (default 4 spaces)
    Comment Out Region  -- Insert ## in front of selected lines
    Uncomment Region    -- Remove leading # or ## from selected lines
    Tabify Region       -- Turns *leading* stretches of spaces into tabs.
            (Note: We recommend using 4 space blocks to indent Python code.)
    Untabify Region     -- Turn *all* tabs into the corrent number of spaces
    Toggle tabs         -- Open a dialog to switch between indenting with
                           spaces and tabs.
    New Indent Width... -- Open a dialog to change indent width.  The
                           accepted default by the Python community is 4
                           spaces.
    Format Paragraph    -- Reformat the current blank-line-separated
                           paragraph. All lines in the paragraph will be
                           formatted to less than 80 columns.
    ---
    Strip trailing whitespace -- Removed any space characters after the end
                                 of the last non-space character
```

Run Menu (Editor window only):

```
    Python Shell -- Open or wake up the Python shell window
    ---
    Check Module -- Check the syntax of the module currently open in the
                    Editor window.  If the module has not been saved IDLE
                    will prompt the user to save the code.
    Run Module   -- Restart the shell to clean the environment, then
                    execute the currently open module. If the module has
                    not been saved IDLE will prompt the user to save the
                    code.
```

Shell Menu (Shell window only):

```
    View Last Restart -- Scroll the shell window to the last Shell restart
    Restart Shell     -- Restart the shell to clean the environment
```

Debug Menu (Shell window only):

```
    Go to File/Line   -- Look around the insert point for a filename
                         and line number, open the file, and show the line.
                         Useful to view the source lines referenced in an
                         exception traceback.  Available in the context
                         menu of the Shell window.
    Debugger (toggle) -- This feature is not complete and considered
                         experimental. Run commands in the shell under the
                         debugger.
    Stack Viewer      -- Show the stack traceback of the last exception
    Auto-open Stack Viewer (toggle) -- Toggle automatically opening the
```

```
                                    stack viewer on unhandled
                                    exception
```

Options Menu (Shell and Editor):

```
    Configure IDLE -- Open a configuration dialog.  Fonts, indentation,
                      keybindings, and color themes may be altered.
                      Startup Preferences may be set, and additional Help
                      sources can be specified.

    ---
    Code Context (toggle) -- Open a pane at the top of the edit window
                             which shows the block context of the section
                             of code which is scrolling off the top or the
                             window. This is not present in the Shell
                             window only the Editor window.
```

Window Menu (Shell and Editor):

```
    Zoom Height -- Toggles the window between normal size (40x80 initial
    setting) and maximum height.  The initial size is in the Configure
    IDLE dialog under the general tab.
    ---
    The rest of this menu lists the names of all open windows;
    select one to bring it to the foreground (deiconifying it if
    necessary).
```

Help Menu:

```
    About IDLE  -- Version, copyright, license, credits
    ---
    IDLE Help   -- Display this file which is a help file for IDLE
                   detailing the menu options, basic editing and navigation,
                   and other tips.
    Python Docs -- Access local Python documentation, if
                   installed.  Or will start a web browser and open
                   docs.python.org showing the latest Python documentation.
    ---
    Additional help sources may be added here with the Configure IDLE
    dialog under the General tab.
```

Editor context menu (Right-click / Control-click on OS X in Edit window):

```
    Cut              -- Copy a selection into system-wide clipboard,
                        then delete the selection
    Copy             -- Copy selection into system-wide clipboard
    Paste            -- Insert system-wide clipboard into window
    Set Breakpoint   -- Sets a breakpoint. Breakpoints are only enabled
                        when the debugger is open.
    Clear Breakpoint -- Clears the breakpoint on that line
```

Shell context menu (Right-click / Control-click on OS X in Shell window):

```
    Cut              -- Copy a selection into system-wide clipboard,
                        then delete the selection
    Copy             -- Copy selection into system-wide clipboard
    Paste            -- Insert system-wide clipboard into window
    ---
    Go to file/line  -- Same as in Debug menu
```

** TIPS ** ==========

Additional Help Sources:

Windows users can Google on zopeshelf.chm to access Zope help files in the Windows help format. The Additional Help Sources feature of the configuration GUI supports .chm, along with any other filetypes supported by your browser. Supply a Menu Item title, and enter the location in the Help File Path slot of the New Help Source dialog. Use http:// and/or www. to identify external URLs, or download the file and browse for its path on your machine using the Browse button.

All users can access the extensive sources of help, including tutorials, available at docs.python.org. Selected URLs can be added or removed from the Help menu at any time using Configure IDLE.

Basic editing and navigation:

Backspace deletes char to the left; DEL deletes char to the right.
Control-backspace deletes word left, Control-DEL deletes word right.
Arrow keys and Page Up/Down move around.
Control-left/right Arrow moves by words in a strange but useful way.
Home/End go to begin/end of line.
Control-Home/End go to begin/end of file.
Some useful Emacs bindings are inherited from Tcl/Tk:

Control-a	beginning of line
Control-e	end of line
Control-k	kill line (but doesn't put it in clipboard)
Control-l	center window around the insertion point

Standard keybindings (like Control-c to copy and Control-v to paste) may work. Keybindings are selected in the Configure IDLE dialog.

Automatic indentation:

After a block-opening statement, the next line is indented by 4 spaces (in the Python Shell window by one tab). After certain keywords (break, return etc.) the next line is dedented. In leading indentation, Backspace deletes up to 4 spaces if they are there. Tab inserts spaces (in the Python Shell window one tab), number depends on Indent Width. Currently tabs are restricted to four spaces due to Tcl/Tk limitations.

See also the indent/dedent region commands in the edit menu.

Completions:

Completions are supplied for functions, classes, and attributes of classes, both built-in and user-defined. Completions are also provided for filenames.

The AutoCompleteWindow (ACW) will open after a predefined delay (default is two seconds) after a '.' or (in a string) an os.sep is typed. If after one of those characters (plus zero or more other characters) a tab is typed the ACW will open immediately if a possible continuation is found.

If there is only one possible completion for the characters entered, a tab will supply that completion without opening the ACW.

'Show Completions' will force open a completions window, by default the

Control-space keys will open a completions window. In an empty string, this will contain the files in the current directory. On a blank line, it will contain the built-in and user-defined functions and classes in the current name spaces, plus any modules imported. If some characters have been entered, the ACW will attempt to be more specific.

If string of characters is typed, the ACW selection will jump to the entry most closely matching those characters. Entering a tab will cause the longest non-ambiguous match to be entered in the Edit window or Shell. Two tabs in a row will supply the current ACW selection, as will return or a double click. Cursor keys, Page Up/Down, mouse selection, and the scroll wheel all operate on the ACW.

"Hidden" attributes can be accessed by typing the beginning of hidden name after a '.', e.g. '_'. This allows access to modules with '__all__' set, or to class-private attributes.

Completions and the 'Expand Word' facility can save a lot of typing!

Completions are currently limited to those in the namespaces. Names in an Editor window which are not via __main__ or sys.modules will not be found. Run the module once with your imports to correct this situation. Note that IDLE itself places quite a few modules in sys.modules, so much can be found by default, e.g. the re module.

If you don't like the ACW popping up unbidden, simply make the delay longer or disable the extension. Or another option is the delay could be set to zero. Another alternative to preventing ACW popups is to disable the call tips extension.

Python Shell window:

Control-c interrupts executing command.
Control-d sends end-of-file; closes window if typed at >>> prompt.
Alt-/ expand word is also useful to reduce typing.

Command history:

Alt-p retrieves previous command matching what you have typed. On OS X use Control-p.
Alt-n retrieves next. On OS X use Control-n.
Return while cursor is on a previous command retrieves that command.

Syntax colors:

The coloring is applied in a background "thread", so you may occasionally see uncolorized text. To change the color scheme, use the Configure IDLE / Highlighting dialog.

Python default syntax colors:

Keywords	orange
Builtins	royal purple
Strings	green
Comments	red
Definitions	blue

Shell default colors:

```
    Console output  brown
    stdout          blue
    stderr          red
    stdin           black
```

Other preferences:

```
    The font preferences, highlighting, keys, and general preferences can
    be changed via the Configure IDLE menu option.  Be sure to note that
    keys can be user defined, IDLE ships with four built in key sets. In
    addition a user can create a custom key set in the Configure IDLE
    dialog under the keys tab.
```

Command line usage:

```
    Enter idle -h at the command prompt to get a usage message.

    idle.py [-c command] [-d] [-e] [-s] [-t title] [arg] ...

    -c command  run this command
    -d          enable debugger
    -e          edit mode; arguments are files to be edited
    -s          run $IDLESTARTUP or $PYTHONSTARTUP first
    -t title    set title of shell window

    If there are arguments:
    1. If -e is used, arguments are files opened for editing and sys.argv
       reflects the arguments passed to IDLE itself.
    2. Otherwise, if -c is used, all arguments are placed in
       sys.argv[1:...], with sys.argv[0] set to -c.
    3. Otherwise, if neither -e nor -c is used, the first argument is a
       script which is executed with the remaining arguments in
       sys.argv[1:...]  and sys.argv[0] set to the script name.  If the
       script name is -, no script is executed but an interactive Python
       session is started; the arguments are still available in sys.argv.
```

Running without a subprocess: (DEPRECATED in Python 3.4 see Issue 16123)

```
    If IDLE is started with the -n command line switch it will run in a
    single process and will not create the subprocess which runs the RPC
    Python execution server.  This can be useful if Python cannot create
    the subprocess or the RPC socket interface on your platform.  However,
    in this mode user code is not isolated from IDLE itself.  Also, the
    environment is not restarted when Run/Run Module (F5) is selected.  If
    your code has been modified, you must reload() the affected modules and
    re-import any specific items (e.g. from foo import baz) if the changes
    are to take effect.  For these reasons, it is preferable to run IDLE
    with the default subprocess if at all possible.
```

Extensions:

```
    IDLE contains an extension facility.  See the beginning of
    config-extensions.def in the idlelib directory for further information.
    The default extensions are currently:

            FormatParagraph
            AutoExpand
            ZoomHeight
            ScriptBinding
            CallTips
```

```
ParenMatch
AutoComplete
CodeContext
```

IDLE History

This file contains the release messages for previous IDLE releases. As you read on you go back to the dark ages of IDLE's history.

What's New in IDLEfork 0.8.1?

Release date: 22-Jul-2001

- New tarball released as a result of the 'revitalisation' of the IDLEfork project.
- This release requires python 2.1 or better. Compatibility with earlier versions of python (especially ancient ones like 1.5x) is no longer a priority in IDLEfork development.
- This release is based on a merging of the earlier IDLE fork work with current cvs IDLE (post IDLE version 0.8), with some minor additional coding by Kurt B. Kaiser and Stephen M. Gava.
- This release is basically functional but also contains some known breakages, particularly with running things from the shell window. Also the debugger is not working, but I believe this was the case with the previous IDLE fork release (0.7.1) as well.
- This release is being made now to mark the point at which IDLEfork is launching into a new stage of development.
- IDLEfork CVS will now be branched to enable further development and exploration of the two "execution in a remote process" patches submitted by David Scherer (David's is currently in IDLEfork) and GvR, while stabilisation and development of less heavyweight improvements (like user customisation) can continue on the trunk.

What's New in IDLEfork 0.7.1?

Release date: 15-Aug-2000

- First project tarball released.
- This was the first release of IDLE fork, which at this stage was a combination of IDLE 0.5 and the VPython idle fork, with additional changes coded by David Scherer, Peter Schneider-Kamp and Nicholas Riley.

IDLEfork 0.7.1 - 29 May 2000

David Scherer dscherer at cmu dot edu

- This is a modification of the CVS version of IDLE 0.5, updated as of 2000-03-09. It is alpha software and might be unstable. If it breaks, you get to keep both pieces.
- If you have problems or suggestions, you should either contact me or post to the list at http://www.python.org/mailman/listinfo/idle-dev (making it clear that you are using this modified version of IDLE).
- Changes:
- The ExecBinding module, a replacement for ScriptBinding, executes programs in a separate process, piping standard I/O through an RPC mechanism to an OnDemandOutputWindow in IDLE. It supports executing unnamed programs (through a temporary file). It does not yet support debugging.
- When running programs with ExecBinding, tracebacks will be clipped to exclude system modules. If, however, a system module calls back into the user program, that part of the traceback will be shown.
- The OnDemandOutputWindow class has been improved. In particular, it now supports a readline() function used to implement user input, and a scroll_clear() operation which is used to hide the output of a previous run by scrolling it out of the window.
- Startup behavior has been changed. By default IDLE starts up with just a blank editor window, rather than an interactive window. Opening a file in such a blank window replaces the (nonexistent) contents of that window instead of creating another window. Because of the need to have a well-known port for the ExecBinding protocol, only one copy of IDLE can be running. Additional invocations use the RPC mechanism to report their command line arguments to the copy already running.
- The menus have been reorganized. In particular, the excessively large 'edit' menu has been split up into 'edit', 'format', and 'run'.
- 'Python Documentation' now works on Windows, if the win32api module is present.
- A few key bindings have been changed: F1 now loads Python Documentation instead of the IDLE help; shift-TAB is now a synonym for unindent.
- New modules:

ExecBinding.py Executes program through loader loader.py Bootstraps user program protocol.py RPC protocol Remote.py User-process interpreter spawn.py OS-specific code to start programs

- Files modified:

autoindent.py (bindings tweaked) bindings.py (menus reorganized) config.txt (execbinding enabled) editorwindow.py (new menus, fixed 'Python Documentation') filelist.py (hook for "open in same window") formatparagraph.py (bindings tweaked) idle.bat (removed absolute pathname) idle.pyw (weird bug due to import with same name?) iobinding.py (open in same window, EOL convention) keydefs.py (bindings tweaked) outputwindow.py (readline, scroll_clear, etc) pyshell.py (changed startup behavior) readme.txt ( )

IDLE 0.5 - February 2000 - Release Notes

This is an early release of IDLE, my own attempt at a Tkinter-based IDE for Python.

(For a more detailed change log, see the file ChangeLog.)

FEATURES

IDLE has the following features:

- coded in 100% pure Python, using the Tkinter GUI toolkit (i.e. Tcl/Tk)
- cross-platform: works on Windows and Unix (on the Mac, there are currently problems with Tcl/Tk)
- multi-window text editor with multiple undo, Python colorizing and many other features, e.g. smart indent and call tips
- Python shell window (a.k.a. interactive interpreter)
- debugger (not complete, but you can set breakpoints, view and step)

USAGE

The main program is in the file "idle.py"; on Unix, you should be able to run it by typing "./idle.py" to your shell. On Windows, you can run it by double-clicking it; you can use idle.pyw to avoid popping up a DOS console. If you want to pass command line arguments on Windows, use the batch file idle.bat.

Command line arguments: files passed on the command line are executed, not opened for editing, unless you give the -e command line option. Try "./idle.py -h" to see other command line options.

IDLE requires Python 1.5.2, so it is currently only usable with a Python 1.5.2 distribution. (An older version of IDLE is distributed with Python 1.5.2; you can drop this version on top of it.)

COPYRIGHT

New in IDLE 0.5 (2/15/2000)

Tons of stuff, much of it contributed by Tim Peters and Mark Hammond:

- Status bar, displaying current line/column (Moshe Zadka).
- Better stack viewer, using tree widget. (XXX Only used by Stack Viewer menu, not by the debugger.)
- Format paragraph now recognizes Python block comments and reformats them correctly (MH)
- New version of pyclbr.py parses top-level functions and understands much more of Python's syntax; this is reflected in the class and path browsers (TP)
- Much better auto-indent; knows how to indent the insides of multi-line statements (TP)
- Call tip window pops up when you type the name of a known function followed by an open parenthesis. Hit ESC or click elsewhere in the window to close the tip window (MH)
- Comment out region now inserts ## to make it stand out more (TP)
- New path and class browsers based on a tree widget that looks familiar to Windows users
- Reworked script running commands to be more intuitive: I/O now always goes to the *Python Shell* window, and raw_input() works correctly. You use F5 to import/reload a module: this adds the module name to the **main** namespace. You use Control-F5 to run a script: this runs the script *in* the **main** namespace. The latter also sets sys.argv[] to the script name

New in IDLE 0.4 (4/7/99)

Most important change: a new menu entry "File -> Path browser", shows a 4-column hierarchical browser which lets you browse sys.path, directories, modules, and classes. Yes, it's a superset of the Class browser menu entry. There's also a new internal module, MultiScrolledLists.py, which provides the framework for this dialog.

New in IDLE 0.3 (2/17/99)

Most important changes:

- Enabled support for running a module, with or without the debugger. Output goes to a new window. Pressing F5 in a module is effectively a reload of that module; Control-F5 loads it under the debugger.
- Re-enable tearing off the Windows menu, and make a torn-off Windows menu update itself whenever a window is opened or closed.
- Menu items can now be have a checkbox (when the menu label starts with "!"); use this for the Debugger and "Auto-open stack viewer" (was: JIT stack viewer) menu items.
- Added a Quit button to the Debugger API.
- The current directory is explicitly inserted into sys.path.
- Fix the debugger (when using Python 1.5.2b2) to use canonical filenames for breakpoints, so these actually work. (There's still a lot of work to be done to the management of breakpoints in the debugger though.)
- Closing a window that is still colorizing now actually works.
- Allow dragging of the separator between the two list boxes in the class browser.
- Bind ESC to "close window" of the debugger, stack viewer and class browser. It removes the selection highlighting in regular text windows. (These are standard Windows conventions.)

New in IDLE 0.2 (1/8/99)

Lots of changes; here are the highlights:

General:

- You can now write and configure your own IDLE extension modules; see extend.txt.

File menu:

The command to open the Python shell window is now in the File menu.

Edit menu:

New Find dialog with more options; replace dialog; find in files dialog.

Commands to tabify or untabify a region.

Command to format a paragraph.

Debug menu:

JIT (Just-In-Time) stack viewer toggle -- if set, the stack viewer automaticall pops up when you get a traceback.

Windows menu:

Zoom height -- make the window full height.

Help menu:

The help text now show up in a regular window so you can search and even edit it if you like.

IDLE 0.1 was distributed with the Python 1.5.2b1 release on 12/22/98.

==========

README FOR IDLE TESTS IN IDLELIB.IDLE_TEST

1. Test Files

The idle directory, idlelib, has over 60 xyz.py files. The idle_test subdirectory should contain a test_xyy.py for each. (For test modules, make 'xyz' lower case, and possibly shorten it.) Each file should start with the something like the following template, with the blanks after after '.' and 'as', and before and after '_' filled in. --- import unittest from test.support import requires import idlelib. as

class _Test(unittest.TestCase):

```
def test_(self):
```

if **name** == '**main**': unittest.main(verbosity=2, exit=2) --- Idle tests are run with unittest; do not use regrtest's test_main.

Once test_xyy is written, the following should go at the end of xyy.py, with xyz (lowercased) added after 'test_'. --- if **name** == "**main**": import unittest unittest.main('idlelib.idle_test.test_', verbosity=2, exit=False) ---

2. Gui Tests

Gui tests need 'requires' from test.support (test.test_support in 2.7). A test is a gui test if it creates a Tk root or master object either directly or indirectly by instantiating a tkinter or idle class. For the benefit of test processes that either have no graphical environment available or are not allowed to use it, gui tests must be 'guarded' by "requires('gui')" in a setUp function or method. This will typically be setUpClass.

To avoid interfering with other gui tests, all gui objects must be destroyed and deleted by the end of the test. If a widget, such as a Tk root, is created in a setUpX function, destroy it in the corresponding tearDownX. For module and class attributes, also delete the widget. --- @classmethod def setUpClass(cls): requires('gui') cls.root = tk.Tk()

```
@classmethod
def tearDownClass(cls):
    cls.root.destroy()
    del cls.root
```

Support.requires('gui') causes the test(s) it guards to be skipped if any of a few conditions are met: - The tests are being run by regrtest.py, and it was started without enabling the "gui" resource with the "-u" command line option. - The tests are being run on Windows by a service that is not allowed to interact with the graphical environment. - The tests are being run on Mac OSX in a process that cannot make a window manager connection. - tkinter.Tk cannot be successfully instantiated for some reason. - test.support.use_resources has been set by something other than regrtest.py and does not contain "gui".

Since non-gui tests always run, but gui tests only sometimes, tests of non-gui operations should best avoid needing a gui. Methods that make incidental use of tkinter (tk) variables and messageboxes can do this by using the mock classes in idle_test/mock_tk.py. There is also a mock text that will handle some uses of the tk Text widget.

3. Running Tests

Assume that xyz.py and test_xyz.py end with the "if **name**" statements given above. In Idle, pressing F5 in an editor window with either loaded will run all tests in the test_xyz file with the version of Python running Idle. The test report and any tracebacks will appear in the Shell window. The options in these "if **name**" statements are appropriate for developers running (as opposed to importing) either of the files during development: verbosity=2 lists all test methods in the file; exit=False avoids a spurious sys.exit traceback that would otherwise occur when running in Idle. The following command lines also run all test methods, including gui tests, in test_xyz.py. (The exceptions are that idlelib and idlelib.idle start Idle and idlelib.PyShell

should (issue 18330).)

python -m idlelib.xyz # With the capitalization of the xyz module python -m idlelib.idle_test.test_xyz

To run all idle_test/test_*.py tests, either interactively ('>>>', with unittest imported) or from a command line, use one of the following. (Notes: in 2.7, 'test ' (with the space) is 'test.regrtest '; where present, -v and -ugui can be omitted.)

> unittest.main('idlelib.idle_test', verbosity=2, exit=False) python -m unittest -v idlelib.idle_test python -m test -v -ugui test_idle python -m test.test_idle

The idle tests are 'discovered' by idlelib.idle_test.**init**.load_tests, which is also imported into test.test_idle. Normally, neither file should be changed when working on individual test modules. The third command runs unittest indirectly through regrtest. The same happens when the entire test suite is run with 'python -m test'. So that command must work for buildbots to stay green. Idle tests must not disturb the environment in a way that makes other tests fail (issue 18081).

To run an individual Testcase or test method, extend the dotted name given to unittest on the command line.

python -m unittest -v idlelib.idle_test.test_xyz.Test_case.test_meth

What's New in IDLE 3.5.0?

- Issue #16893: Update Idle doc chapter to match current Idle and add new information.
- Issue #3068: Add Idle extension configuration dialog to Options menu. Changes are written to HOME/.idlerc/config-extensions.cfg. Original patch by Tal Einat.
- Issue #16233: A module browser (File : Class Browser, Alt+C) requires a editor window with a filename. When Class Browser is requested otherwise, from a shell, output window, or 'Untitled' editor, Idle no longer displays an error box. It now pops up an Open Module box (Alt+M). If a valid name is entered and a module is opened, a corresponding browser is also opened.
- Issue #4832: Save As to type Python files automatically adds .py to the name you enter (even if your system does not display it). Some systems automatically add .txt when type is Text files.
- Issue #21986: Code objects are not normally pickled by the pickle module. To match this, they are no longer pickled when running under Idle.
- Issue #17390: Adjust Editor window title; remove 'Python', move version to end.
- Issue #14105: Idle debugger breakpoints no longer disappear when inseting or deleting lines.
- Issue #17172: Turtledemo can now be run from Idle. Currently, the entry is on the Help menu, but it may move to Run. Patch by Ramchandra Apt and Lita Cho.
- Issue #21765: Add support for non-ascii identifiers to HyperParser.
- Issue #21940: Add unittest for WidgetRedirector. Initial patch by Saimadhav Heblikar.
- Issue #18592: Add unittest for SearchDialogBase. Patch by Phil Webster.
- Issue #21694: Add unittest for ParenMatch. Patch by Saimadhav Heblikar.
- Issue #21686: add unittest for HyperParser. Original patch by Saimadhav Heblikar.
- Issue #12387: Add missing upper(lower)case versions of default Windows key bindings for Idle so Caps Lock does not disable them. Patch by Roger Serwy.
- Issue #21695: Closing a Find-in-files output window while the search is still in progress no longer closes Idle.
- Issue #18910: Add unittest for textView. Patch by Phil Webster.
- Issue #18292: Add unittest for AutoExpand. Patch by Saihadhav Heblikar.
- Issue #18409: Add unittest for AutoComplete. Patch by Phil Webster.
- Issue #21477: htest.py - Improve framework, complete set of tests. Patches by Saimadhav Heblikar
- Issue #18104: Add idlelib/idle_test/htest.py with a few sample tests to begin consolidating and improving human-validated tests of Idle. Change other files as needed to work with htest. Running the module as **main** runs all tests.
- Issue #21139: Change default paragraph width to 72, the PEP 8 recommendation.
- Issue #21284: Paragraph reformat test passes after user changes reformat width.
- Issue #17654: Ensure IDLE menus are customized properly on OS X for non-framework builds and for all variants of Tk.

What's New in IDLE 3.4.0?

- Issue #17390: Display Python version on Idle title bar. Initial patch by Edmond Burnett.
- Issue #5066: Update IDLE docs. Patch by Todd Rovito.
- Issue #17625: Close the replace dialog after it is used.
- Issue #16226: Fix IDLE Path Browser crash. (Patch by Roger Serwy)
- Issue #15853: Prevent IDLE crash on OS X when opening Preferences menu with certain versions of Tk 8.5. Initial patch by Kevin Walzer.

What's New in IDLE 3.3.0?

- Issue #17625: Close the replace dialog after it is used.
- Issue #7163: Propagate return value of sys.stdout.write.
- Issue #15318: Prevent writing to sys.stdin.
- Issue #4832: Modify IDLE to save files with .py extension by default on Windows and OS X (Tk 8.5) as it already does with X11 Tk.
- Issue #13532, #15319: Check that arguments to sys.stdout.write are strings.
- Issue # 12510: Attempt to get certain tool tips no longer crashes IDLE. Erroneous tool tips have been corrected. Default added for callables.
- Issue10365: File open dialog now works instead of crashing even when parent window is closed while dialog is open.
- Issue 14876: use user-selected font for highlight configuration.
- Issue #14937: Perform auto-completion of filenames in strings even for non-ASCII filenames. Likewise for identifiers.
- Issue #8515: Set **file** when run file in IDLE. Initial patch by Bruce Frederiksen.
- IDLE can be launched as `python -m idlelib`
- Issue #14409: IDLE now properly executes commands in the Shell window when it cannot read the normal config files on startup and has to use the built-in default key bindings. There was previously a bug in one of the defaults.
- Issue #3573: IDLE hangs when passing invalid command line args (directory(ies) instead of file(s)).
- Issue #14018: Update checks for unstable system Tcl/Tk versions on OS X to include versions shipped with OS X 10.7 and 10.8 in addition to 10.6.

What's New in IDLE 3.2.1?

Release date: 15-May-11

- Issue #6378: Further adjust idle.bat to start associated Python
- Issue #11896: Save on Close failed despite selecting "Yes" in dialog.
- Issue #1028: Ctrl-space binding to show completions was causing IDLE to exit. Tk < 8.5 was sending invalid Unicode null; replaced with valid null.
- toggle failing on Tk 8.5, causing IDLE exits and strange selection behavior. Issue 4676. Improve selection extension behaviour.
- toggle non-functional when NumLock set on Windows. Issue 3851.

What's New in IDLE 3.1b1?

Release date: 06-May-09

- Use of 'filter' in keybindingDialog.py was causing custom key assignment to fail. Patch 5707 amaury.forgeotdarc.

What's New in IDLE 3.1a1?

Release date: 07-Mar-09

- Issue #4815: Offer conversion to UTF-8 if source files have no encoding declaration and are not encoded in UTF-8.
- Issue #4008: Fix problems with non-ASCII source files.
- Issue #4323: Always encode source as UTF-8 without asking the user (unless a different encoding is declared); remove user configuration of source encoding; all according to PEP 3120.
- Issue #2665: On Windows, an IDLE installation upgraded from an old version would not start if a custom theme was defined.

What's New in IDLE 2.7? (UNRELEASED, but merged into 3.1 releases above.)

Release date: XX-XXX-2010

- idle.py modified and simplified to better support developing experimental versions of IDLE which are not installed in the standard location.
- OutputWindow/PyShell right click menu "Go to file/line" wasn't working with file paths containing spaces. Bug 5559.
- Windows: Version string for the .chm help file changed, file not being accessed Patch 5783 Guilherme Polo
- Allow multiple IDLE GUI/subprocess pairs to exist simultaneously. Thanks to David Scherer for suggesting the use of an ephemeral port for the GUI. Patch 1529142 Weeble.
- Remove port spec from run.py and fix bug where subprocess fails to extract port from command line when warnings are present.
- Tk 8.5 Text widget requires 'wordprocessor' tabstyle attr to handle mixed space/tab properly. Issue 5129, patch by Guilherme Polo.
- Issue #3549: On MacOS the preferences menu was not present

What's New in IDLE 3.0 final?

Release date: 03-Dec-2008

- IDLE would print a "Unhandled server exception!" message when internal debugging is enabled.
- Issue #4455: IDLE failed to display the windows list when two windows have the same title.
- Issue #4383: When IDLE cannot make the connection to its subprocess, it would fail to properly display the error message.

What's New in IDLE 3.0a3?

Release date: 29-Feb-2008

- help() was not paging to the shell. Issue1650.
- CodeContext was not importing.
- Corrected two 3.0 compatibility errors reported by Mark Summerfield: http://mail.python.org/pipermail/python-3000/2007-December/011491.html
- Shell was not colorizing due to bug introduced at r57998, Bug 1586.
- Issue #1585: IDLE uses non-existent xrange() function.

What's New in IDLE 3.0a2?

Release date: 06-Dec-2007

- Windows EOL sequence not converted correctly, encoding error. Caused file save to fail. Bug 1130.

What's New in IDLE 3.0a1?

Release date: 31-Aug-2007

- IDLE converted to Python 3000 syntax.
- Strings became Unicode.
- CallTips module now uses the inspect module to produce the argspec.
- IDLE modules now use absolute import instead of implied relative import.
- atexit call replaces sys.exitfunc. The functionality of delete-exitfunc flag in config-main.cfg remains unchanged: if set, registered exit functions will be cleared before IDLE exits.

What's New in IDLE 2.6 final?

Release date: 01-Oct-2008, merged into 3.0 releases detailed above (3.0rc2)

- Issue #2665: On Windows, an IDLE installation upgraded from an old version would not start if a custom theme was defined.
- Home / Control-A toggles between left margin and end of leading white space. issue1196903, patch by Jeff Shute.
- Improved AutoCompleteWindow logic. issue2062, patch by Tal Einat.
- Autocompletion of filenames now support alternate separators, e.g. the '/' char on Windows. issue2061 Patch by Tal Einat.
- Configured selection highlighting colors were ignored; updating highlighting in the config dialog would cause non-Python files to be colored as if they were Python source; improve use of ColorDelagator. Patch 1334. Tal Einat.
- ScriptBinding event handlers weren't returning 'break'. Patch 2050, Tal Einat
- There was an error on exit if no sys.exitfunc was defined. Issue 1647.
- Could not open files in .idlerc directory if latter was hidden on Windows. Issue 1743, Issue 1862.
- Configure Dialog: improved layout for keybinding. Patch 1457 Tal Einat.
- tabpage.py updated: tabbedPages.py now supports multiple dynamic rows of tabs. Patch 1612746 Tal Einat.
- Add confirmation dialog before printing. Patch 1717170 Tal Einat.
- Show paste position if > 80 col. Patch 1659326 Tal Einat.
- Update cursor color without restarting. Patch 1725576 Tal Einat.
- Allow keyboard interrupt only when user code is executing in subprocess. Patch 1225 Tal Einat (reworked from IDLE-Spoon).
- configDialog cleanup. Patch 1730217 Tal Einat.
- textView cleanup. Patch 1718043 Tal Einat.
- Clean up EditorWindow close.
- Patch 1693258: Fix for duplicate "preferences" menu-OS X. Backport of r56204.
- OSX: Avoid crash for those versions of Tcl/Tk which don't have a console
- Bug in idlelib.MultiCall: Options dialog was crashing IDLE if there was an option in config-extensions w/o a value. Patch #1672481, Tal Einat
- Corrected some bugs in AutoComplete. Also, Page Up/Down in ACW implemented; mouse and cursor selection in ACWindow implemented; double Tab inserts current selection and closes ACW (similar to double-click and Return); scroll wheel now works in ACW. Added AutoComplete instructions to IDLE Help.
- AutoCompleteWindow moved below input line, will move above if there isn't enough space. Patch 1621265 Tal Einat

- Calltips now 'handle' tuples in the argument list (display '' :) Suggested solution by Christos Georgiou, Bug 791968.
- Add 'raw' support to configHandler. Patch 1650174 Tal Einat.
- Avoid hang when encountering a duplicate in a completion list. Bug 1571112.
- Patch #1362975: Rework CodeContext indentation algorithm to avoid hard-coding pixel widths.
- Bug #813342: Start the IDLE subprocess with -Qnew if the parent is started with that option.
- Honor the "Cancel" action in the save dialog (Debian bug #299092)
- Some syntax errors were being caught by tokenize during the tabnanny check, resulting in obscure error messages. Do the syntax check first. Bug 1562716, 1562719
- IDLE's version number takes a big jump to match the version number of the Python release of which it's a part.

What's New in IDLE 1.2?

Release date: 19-SEP-2006

What's New in IDLE 1.2c1?

Release date: 17-AUG-2006

- File menu hotkeys: there were three 'p' assignments. Reassign the 'Save Copy As' and 'Print' hotkeys to 'y' and 't'. Change the Shell hotkey from 's' to 'l'.
- IDLE honors new quit() and exit() commands from site.py Quitter() object. Patch 1540892, Jim Jewett
- The 'with' statement is now a Code Context block opener. Patch 1540851, Jim Jewett
- Retrieval of previous shell command was not always preserving indentation (since 1.2a1) Patch 1528468 Tal Einat.
- Changing tokenize (39046) to detect dedent broke tabnanny check (since 1.2a1)
- ToggleTab dialog was setting indent to 8 even if cancelled (since 1.2a1).
- When used w/o subprocess, all exceptions were preceded by an error message claiming they were IDLE internal errors (since 1.2a1).

What's New in IDLE 1.2b3?

Release date: 03-AUG-2006

- Bug #1525817: Don't truncate short lines in IDLE's tool tips.
- Bug #1517990: IDLE keybindings on MacOS X now work correctly
- Bug #1517996: IDLE now longer shows the default Tk menu when a path browser, class browser or debugger is the frontmost window on MacOS X
- EditorWindow.test() was failing. Bug 1417598
- EditorWindow failed when used stand-alone if sys.ps1 not set. Bug 1010370 Dave Florek
- Tooltips failed on new-syle class **init** args. Bug 1027566 Loren Guthrie
- Avoid occasional failure to detect closing paren properly. Patch 1407280 Tal Einat
- Rebinding Tab key was inserting 'tab' instead of 'Tab'. Bug 1179168.
- Colorizer now handles # correctly, also unicode strings and 'as' keyword in comment directly following import command. Closes 1325071. Patch 1479219 Tal Einat

What's New in IDLE 1.2b2?

Release date: 11-JUL-2006

What's New in IDLE 1.2b1?

Release date: 20-JUN-2006

What's New in IDLE 1.2a2?

Release date: 27-APR-2006

What's New in IDLE 1.2a1?

Release date: 05-APR-2006

- Patch #1162825: Support non-ASCII characters in IDLE window titles.
- Source file f.flush() after writing; trying to avoid lossage if user kills GUI.
- Options / Keys / Advanced dialog made functional. Also, allow binding of 'movement' keys.
- 'syntax' patch adds improved calltips and a new class attribute listbox. MultiCall module allows binding multiple actions to an event. Patch 906702 Noam Raphael
- Better indentation after first line of string continuation. IDLEfork Patch 681992, Noam Raphael
- Fixed CodeContext alignment problem, following suggestion from Tal Einat.
- Increased performance in CodeContext extension Patch 936169 Noam Raphael
- Mac line endings were incorrect when pasting code from some browsers when using X11 and the Fink distribution. Python Bug 1263656.
- when cursor is on a previous command retrieves that command. Instead of replacing the input line, the previous command is now appended to the input line. Indentation is preserved, and undo is enabled. Patch 1196917 Jeff Shute
- Clarify "tab/space" Error Dialog and "Tab Width" Dialog associated with the Untabify command.
- Corrected "tab/space" Error Dialog to show correct menu for Untabify. Patch 1196980 Jeff Shute
- New files are colorized by default, and colorizing is removed when saving as non-Python files. Patch 1196895 Jeff Shute Closes Python Bugs 775012 and 800432, partial fix IDLEfork 763524
- Improve subprocess link error notification.
- run.py: use Queue's blocking feature instead of sleeping in the main loop. Patch # 1190163 Michiel de Hoon
- Add config-main option to make the 'history' feature non-cyclic. Default remains cyclic. Python Patch 914546 Noam Raphael.
- Removed ability to configure tabs indent from Options dialog. This 'feature' has never worked and no one has complained. It is still possible to set a default tabs (v. spaces) indent 'manually' via config-main.def (or to turn on tabs for the current EditorWindow via the Format menu) but IDLE will encourage indentation via spaces.
- Enable setting the indentation width using the Options dialog. Bug # 783877
- Add keybindings for del-word-left and del-word-right.
- Discourage using an indent width other than 8 when using tabs to indent Python code.
- Restore use of EditorWindow.set_indentation_params(), was dead code since Autoindent was merged into EditorWindow. This allows IDLE to conform to the indentation width of a loaded file. (But it still will not switch to tabs even if the file uses tabs.) Any change in indent width is local to that window.
- Add Tabnanny check before Run/F5, not just when Checking module.
- If an extension can't be loaded, print warning and skip it instead of erroring out.

- Improve error handling when .idlerc can't be created (warn and exit).
- The GUI was hanging if the shell window was closed while a raw_input() was pending. Restored the quit() of the readline() mainloop(). http://mail.python.org/pipermail/idle-dev/2004-December/002307.html
- The remote procedure call module rpc.py can now access data attributes of remote registered objects. Changes to these attributes are local, however.

What's New in IDLE 1.1?

Release date: 30-NOV-2004

- On OpenBSD, terminating IDLE with ctrl-c from the command line caused a stuck subprocess MainThread because only the SocketThread was exiting.

What's New in IDLE 1.1b3/rc1?

Release date: 18-NOV-2004

- Saving a Keyset w/o making changes (by using the "Save as New Custom Key Set" button) caused IDLE to fail on restart (no new keyset was created in config-keys.cfg). Also true for Theme/highlights. Python Bug 1064535.

- A change to the linecache.py API caused IDLE to exit when an exception was raised while running without the subprocess (-n switch). Python Bug 1063840.

What's New in IDLE 1.1b2?

Release date: 03-NOV-2004

- When paragraph reformat width was made configurable, a bug was introduced that caused reformatting of comment blocks to ignore how far the block was indented, effectively adding the indentation width to the reformat width. This has been repaired, and the reformat width is again a bound on the total width of reformatted lines.

What's New in IDLE 1.1b1?

Release date: 15-OCT-2004

What's New in IDLE 1.1a3?

Release date: 02-SEP-2004

- Improve keyboard focus binding, especially in Windows menu. Improve window raising, especially in the Windows menu and in the debugger. IDLEfork 763524.

- If user passes a non-existent filename on the commandline, just open a new file, don't raise a dialog. IDLEfork 854928.

What's New in IDLE 1.1a2?

Release date: 05-AUG-2004

- EditorWindow.py was not finding the .chm help file on Windows. Typo at Rev 1.54. Python Bug 990954

- checking sys.platform for substring 'win' was breaking IDLE docs on Mac (darwin). Also, Mac Safari browser requires full file:// URIs. SF 900580.

What's New in IDLE 1.1a1?

Release date: 08-JUL-2004

- Redirect the warning stream to the shell during the ScriptBinding check of user code and format the warning similarly to an exception for both that check and for runtime warnings raised in the subprocess.

- CodeContext hint pane visibility state is now persistent across sessions. The pane no longer appears in the shell window. Added capability to limit extensions to shell window or editor windows. Noam Raphael addition to Patch 936169.

- Paragraph reformat width is now a configurable parameter in the Options GUI.

- New Extension: CodeContext. Provides block structuring hints for code which has scrolled above an edit window. Patch 936169 Noam Raphael.

- If nulls somehow got into the strings in recent-files.lst EditorWindow.update_recent_files_list() was failing. Python Bug 931336.

- If the normal background is changed via Configure/Highlighting, it will update immediately, thanks to the previously mentioned patch by Nigel Rowe.

- Add a highlight theme for builtin keywords. Python Patch 805830 Nigel Rowe This also fixed IDLEfork bug [693418] Normal text background color not refreshed and Python bug [897872] Unknown color name on HP-UX

- rpc.py:SocketIO - Large modules were generating large pickles when downloaded to the execution server. The return of the OK response from the subprocess initialization was interfering and causing the sending socket to be not ready. Add an IO ready test to fix this. Moved the polling IO ready test into pollpacket().

- Fix typo in rpc.py, s/b "pickle.PicklingError" not "pickle.UnpicklingError".

- Added a Tk error dialog to run.py inform the user if the subprocess can't connect to the user GUI process. Added a timeout to the GUI's listening socket. Added Tk error dialogs to PyShell.py to announce a failure to bind the port or connect to the subprocess. Clean up error handling during connection initiation phase. This is an update of Python Patch 778323.

- Print correct exception even if source file changed since shell was restarted. IDLEfork Patch 869012 Noam Raphael

- Keybindings with the Shift modifier now work correctly. So do bindings which use the Space key. Limit unmodified user keybindings to the function keys. Python Bug 775353, IDLEfork Bugs 755647, 761557

- After an exception, run.py was not setting the exception vector. Noam Raphael suggested correcting this so pdb's postmortem pm() would work. IDLEfork Patch 844675

- IDLE now does not fail to save the file anymore if the Tk buffer is not a Unicode string, yet eol_convention is. Python Bugs 774680, 788378

- IDLE didn't start correctly when Python was installed in "Program Files" on W2K and XP. Python Bugs 780451, 784183

- config-main.def documentation incorrectly referred to idle- instead of config- filenames. SF 782759 Also added note about .idlerc location.

What's New in IDLE 1.0?

Release date: 29-Jul-2003

- Added a banner to the shell discussing warnings possibly raised by personal firewall software. Added same comment to README.txt.

What's New in IDLE 1.0 release candidate 2?

Release date: 24-Jul-2003

- Calltip error when docstring was None Python Bug 775541

What's New in IDLE 1.0 release candidate 1?

Release date: 18-Jul-2003

- Updated extend.txt, help.txt, and config-extensions.def to correctly reflect the current status of the configuration system. Python Bug 768469
- Fixed: Call Tip Trimming May Loop Forever. Python Patch 769142 (Daniels)
- Replaced apply(f, args, kwds) with f(*args, **kwargs) to improve performance Python Patch 768187
- Break or continue statements outside a loop were causing IDLE crash Python Bug 767794
- Convert Unicode strings from readline to IOBinding.encoding. Also set sys.std{in|out|err}.encoding, for both the local and the subprocess case. SF IDLEfork patch 682347.

What's New in IDLE 1.0b2?

Release date: 29-Jun-2003

- Extend AboutDialog.ViewFile() to support file encodings. Make the CREDITS file Latin-1.
- Updated the About dialog to reflect re-integration into Python. Provide buttons to display Python's NEWS, License, and Credits, plus additional buttons for IDLE's README and NEWS.
- TextViewer() now has a third parameter which allows inserting text into the viewer instead of reading from a file.
- (Created the .../Lib/idlelib directory in the Python CVS, which is a clone of IDLEfork modified to install in the Python environment. The code in the interrupt module has been moved to thread.interrupt_main().)
- Printing the Shell window was failing if it was not saved first SF 748975
- When using the Search in Files dialog, if the user had a selection highlighted in his Editor window, insert it into the dialog search field.
- The Python Shell entry was disappearing from the Windows menu.
- Update the Windows file list when a file name change occurs
- Change to File / Open Module: always pop up the dialog, using the current selection as the default value. This is easier to use habitually.
- Avoided a problem with starting the subprocess when 'localhost' doesn't resolve to the user's loopback interface. SF 747772
- Fixed an issue with highlighted errors never de-colorizing. SF 747677. Also improved notification of Tabnanny Token Error.
- File / New will by default save in the directory of the Edit window from which it was initiated. SF 748973 Guido van Rossum patch.

What's New in IDLEfork 0.9b1?

Release date: 02-Jun-2003

- The current working directory of the execution environment (and shell following completion of execution) is now that of the module being run.
- Added the delete-exitfunc option to config-main.def. (This option is not included in the Options dialog.) Setting this to True (the default) will cause IDLE to not run sys.exitfunc/atexit when the subprocess exits.
- IDLE now preserves the line ending codes when editing a file produced on a different platform. SF 661759, SF 538584
- Reduced default editor font size to 10 point and increased window height to provide a better initial impression on Windows.
- Options / Fonts/Tabs / Set Base Editor Font: List box was not highlighting the default font when first installed on Windows. SF 661676
- Added Autosave feature: when user runs code from edit window, if the file has been modified IDLE will silently save it if Autosave is enabled. The option is set in the Options dialog, and the default is to prompt the user to save the file. SF 661318 Bruce Sherwood patch.
- Improved the RESTART annotation in the shell window when the user restarts the shell while it is generating output. Also improved annotation when user repeatedly hammers the Ctrl-F6 restart.
- Allow IDLE to run when not installed and cwd is not the IDLE directory SF Patch 686254 "Run IDLEfork from any directory without set-up" - Raphael
- When a module is run from an EditorWindow: if its directory is not in sys.path, prepend it. This allows the module to import other modules in the same directory. Do the same for a script run from the command line.
- Correctly restart the subprocess if it is running user code and the user attempts to run some other module or restarts the shell. Do the same if the link is broken and it is possible to restart the subprocess and re- connect to the GUI. SF RFE 661321.
- Improved exception reporting when running commands or scripts from the command line.
- Added a -n command line switch to start IDLE without the subprocess. Removed the Shell menu when running in that mode. Updated help messages.
- Added a comment to the shell startup header to indicate when IDLE is not using the subprocess.
- Restore the ability to run without the subprocess. This can be important for some platforms or configurations. (Running without the subprocess allows the debugger to trace through parts of IDLE itself, which may or may not be desirable, depending on your point of view. In addition, the traditional reload/import tricks must be use if user source code is changed.) This is helpful for developing IDLE using IDLE, because one instance can be used to edit the code and a separate instance run to test changes. (Multiple concurrent IDLE instances with subprocesses is a future feature)
- Improve the error message a user gets when saving a file with non-ASCII characters and no source encoding is specified. Done by adding a dialog 'EncodingMessage', which contains the line to add in a fixed-font entry widget, and which has a button to add that line to the file automatically. Also, add a configuration option 'EditorWindow/encoding', which has three possible values: none, utf-8, and locale. None is the default: IDLE will show this dialog when non-ASCII characters are encountered. utf-8 means that files with non-ASCII characters are saved as utf-8-with-bom. locale means that files are saved in the locale's encoding; the dialog is only displayed if the source contains characters

outside the locale's charset. SF 710733 - Loewis

- Improved I/O response by tweaking the wait parameter in various calls to signal.signal().
- Implemented a threaded subprocess which allows interrupting a pass loop in user code using the 'interrupt' extension. User code runs in MainThread, while the RPCServer is handled by SockThread. This is necessary because Windows doesn't support signals.
- Implemented the 'interrupt' extension module, which allows a subthread to raise a KeyboardInterrupt in the main thread.
- Attempting to save the shell raised an error related to saving breakpoints, which are not implemented in the shell
- Provide a correct message when 'exit' or 'quit' are entered at the IDLE command prompt SF 695861
- Eliminate extra blank line in shell output caused by not flushing stdout when user code ends with an unterminated print. SF 695861
- Moved responsibility for exception formatting (i.e. pruning IDLE internal calls) out of rpc.py into the client and server.
- Exit IDLE cleanly even when doing subprocess I/O
- Handle subprocess interrupt with an RPC message.
- Restart the subprocess if it terminates itself. (VPython programs do that)
- Support subclassing of exceptions, including in the shell, by moving the exception formatting to the subprocess.

What's New in IDLEfork 0.9 Alpha 2?

Release date: 27-Jan-2003

- Updated INSTALL.txt to claify use of the python2 rpm.
- Improved formatting in IDLE Help.
- Run menu: Replace "Run Script" with "Run Module".
- Code encountering an unhandled exception under the debugger now shows the correct traceback, with IDLE internal levels pruned out.
- If an exception occurs entirely in IDLE, don't prune the IDLE internal modules from the traceback displayed.
- Class Browser and Path Browser now use Alt-Key-2 for vertical zoom.
- IDLE icons will now install correctly even when setup.py is run from the build directory
- Class Browser now compatible with Python2.3 version of pyclbr.py
- Left cursor move in presence of selected text now moves from left end of the selection.
- Add Meta keybindings to "IDLE Classic Windows" to handle reversed Alt/Meta on some Linux distros.
- Change default: IDLE now starts with Python Shell.
- Removed the File Path from the Additional Help Sources scrolled list.
- Add capability to access Additional Help Sources on the web if the Help File Path begins with //http or www. (Otherwise local path is validated, as before.)
- Additional Help Sources were not being posted on the Help menu in the order entered. Implement sorting the list by [HelpFiles] 'option' number.
- Add Browse button to New Help Source dialog. Arrange to start in Python/Doc if platform is Windows, otherwise start in current directory.
- Put the Additional Help Sources directly on the Help menu instead of in an Extra Help cascade menu. Rearrange the Help menu so the Additional Help Sources come last. Update help.txt appropriately.
- Fix Tk root pop-ups in configSectionNameDialog.py and configDialog.py
- Uniform capitalization in General tab of ConfigDialog, update the doc string.
- Fix bug in ConfigDialog where SaveAllChangedConfig() was unexpectedly deleting Additional Help Sources from the user's config file.
- Make configHelpSourceEdit OK button the default and bind
- Fix Tk root pop-ups in configHelpSourceEdit: error dialogs not attached to parents.
- Use os.startfile() to open both Additional Help and Python Help on the Windows platform. The application associated with the file type will act as the viewer. Windows help files (.chm) are now supported via the Settings/General/Additional Help facility.
- If Python Help files are installed locally on Linux, use them instead of accessing python.org.
- Make the methods for finding the Python help docs more robust, and make them work in the installed

configuration, also.

- On the Save Before Run dialog, make the OK button the default. One less mouse action!
- Add a method: EditorWindow.get_geometry() for future use in implementing window location persistence.
- Removed the "Help/Advice" menu entry. Thanks, David! We'll remember!
- Change the "Classic Windows" theme's paste key to be .
- Rearrange the Shell menu to put Stack Viewer entries adjacent.
- Add the ability to restart the subprocess interpreter from the shell window; add an associated menu entry "Shell/Restart" with binding Control-F6. Update IDLE help.
- Upon a restart, annotate the shell window with a "restart boundary". Add a shell window menu "Shell/View Restart" with binding F6 to jump to the most recent restart boundary.
- Add Shell menu to Python Shell; change "Settings" to "Options".
- Remove incorrect comment in setup.py: IDLEfork is now installed as a package.
- Add INSTALL.txt, HISTORY.txt, NEWS.txt to installed configuration.
- In installer text, fix reference to Visual Python, should be VPython. Properly credit David Scherer.
- Modified idle, idle.py, idle.pyw to improve exception handling.

What's New in IDLEfork 0.9 Alpha 1?

Release date: 31-Dec-2002

- First release of major new functionality. For further details refer to Idle-dev and/or the Sourceforge CVS.
- Adapted to the Mac platform.
- Overhauled the IDLE startup options and revised the idle -h help message, which provides details of command line usage.
- Multiple bug fixes and usability enhancements.
- Introduced the new RPC implementation, which includes a debugger. The output of user code is to the shell, and the shell may be used to inspect the environment after the run has finished. (In version 0.8.1 the shell environment was separate from the environment of the user code.)
- Introduced the configuration GUI and a new About dialog.
- Removed David Scherer's Remote Procedure Call code and replaced with Guido van Rossum's. GvR code has support for the IDLE debugger and uses the shell to inspect the environment of code Run from an Edit window. Files removed: ExecBinding.py, loader.py, protocol.py, Remote.py, spawn.py

Refer to HISTORY.txt for additional information on earlier releases.

IDLE is Python's Tkinter-based Integrated DeveLopment Environment.

IDLE emphasizes a lightweight, clean design with a simple user interface. Although it is suitable for beginners, even advanced users will find that IDLE has everything they really need to develop pure Python code.

IDLE features a multi-window text editor with multiple undo, Python colorizing, and many other capabilities, e.g. smart indent, call tips, and autocompletion.

The editor has comprehensive search functions, including searching through multiple files. Class browsers and path browsers provide fast access to code objects from a top level viewpoint without dealing with code folding.

There is a Python Shell window which features colorizing and command recall.

IDLE executes Python code in a separate process, which is restarted for each Run (F5) initiated from an editor window. The environment can also be restarted from the Shell window without restarting IDLE.

This enhancement has often been requested, and is now finally available. The magic "reload/import *" incantations are no longer required when editing and testing a module two or three steps down the import chain.

(Personal firewall software may warn about the connection IDLE makes to its subprocess using this computer's internal loopback interface. This connection is not visible on any external interface and no data is sent to or received from the Internet.)

It is possible to interrupt tightly looping user code, even on Windows.

Applications which cannot support subprocesses and/or sockets can still run IDLE in a single process.

IDLE has an integrated debugger with stepping, persistent breakpoints, and call stack visibility.

There is a GUI configuration manager which makes it easy to select fonts, colors, keybindings, and startup options. This facility includes a feature which allows the user to specify additional help sources, either locally or on the web.

IDLE is coded in 100% pure Python, using the Tkinter GUI toolkit (Tk/Tcl) and is cross-platform, working on Unix, Mac, and Windows.

IDLE accepts command line arguments. Try idle -h to see the options.

If you find bugs or have suggestions or patches, let us know about them by using the Python issue tracker:

http://bugs.python.org

For further details and links, read the Help files and check the IDLE home page at

http://www.python.org/idle/

There is a mail list for IDLE: idle-dev@python.org. You can join at

http://mail.python.org/mailman/listinfo/idle-dev

Original IDLE todo, much of it now outdated:

TO DO:

- improve debugger:
  - manage breakpoints globally, allow bp deletion, tbreak, cbreak etc.
  - real object browser
  - help on how to use it (a simple help button will do wonders)
  - performance? (updates of large sets of locals are slow)
  - better integration of "debug module"
  - debugger should be global resource (attached to flist, not to shell)
  - fix the stupid bug where you need to step twice
  - display class name in stack viewer entries for methods
  - suppress tracing through IDLE internals (e.g. print) DONE
  - add a button to suppress through a specific module or class or method
  - more object inspection to stack viewer, e.g. to view all array items
- insert the initial current directory into sys.path DONE
- default directory attribute for each window instead of only for windows that have an associated filename
- command expansion from keywords, module contents, other buffers, etc.
- "Recent documents" menu item DONE
- Filter region command
- Optional horizontal scroll bar
- more Emacsisms:
  - ^K should cut to buffer
  - M-[, M-] to move by paragraphs
  - incremental search?
- search should indicate wrap-around in some way
- restructure state sensitive code to avoid testing flags all the time
- persistent user state (e.g. window and cursor positions, bindings)
- make backups when saving
- check file mtimes at various points
- Pluggable interface with RCS/CVS/Perforce/Clearcase
- better help?
- don't open second class browser on same module (nor second path browser)
- unify class and path browsers
- Need to define a standard way whereby one can determine one is running inside IDLE (needed for Tk mainloop, also handy for $PYTHONSTARTUP)
- Add more utility methods for use by extensions (a la get_selection)
- Way to run command in totally separate interpreter (fork+os.system?) DONE
- Way to find definition of fully-qualified name: In other words, select "UserDict.UserDict", hit some magic key and it loads up UserDict.py and finds the first def or class for UserDict.
- need a way to force colorization on/off
- need a way to force auto-indent on/off

Details:

- ^O (on Unix -- open-line) should honor autoindent
- after paste, show end of pasted text
- on Windows, should turn short filename to long filename (not only in argv!) (shouldn't this be done -- or undone -- by ntpath.normpath?)
- new autoindent after colon even indents when the colon is in a comment!
- sometimes forward slashes in pathname remain
- sometimes star in window name remains in Windows menu
- With unix bindings, ESC by itself is ignored
- Sometimes for no apparent reason a selection from the cursor to the end of the command buffer

appears, which is hard to get rid of because it stays when you are typing!
- The Line/Col in the status bar can be wrong initially in PyShell DONE

Structural problems:

- too much knowledge in FileList about EditorWindow (for example)
- should add some primitives for accessing the selection etc. to repeat cumbersome code over and over

==========

Jeff Bauer suggests:

- Open Module doesn't appear to handle hierarchical packages.
- Class browser should also allow hierarchical packages.
- Open and Open Module could benefit from a history, DONE either command line style, or Microsoft recent-file style.
- Add a Smalltalk-style inspector (i.e. Tkinspect)

The last suggestion is already a reality, but not yet integrated into IDLE. I use a module called inspector.py, that used to be available from python.org(?) It no longer appears to be in the contributed section, and the source has no author attribution.

In any case, the code is useful for visually navigating an object's attributes, including its container hierarchy.

```
>>> from inspector import Tkinspect
>>> Tkinspect(None, myObject)
```

Tkinspect could probably be extended and refined to integrate better into IDLE.

==========

Comparison to PTUI

- PTUI's help is better (HTML!)
- PTUI can attach a shell to any module
- PTUI has some more I/O commands: open multiple append examine (what's that?)

==========

Notes after trying to run Grail

- Grail does stuff to sys.path based on sys.argv[0]; you must set sys.argv[0] to something decent first (it is normally set to the path of the idle script).
- Grail must be exec'ed in **main** because that's imported by some other parts of Grail.
- Grail uses a module called History and so does idle :-(

==========

Robin Friedrich's items:

Things I'd like to see: - I'd like support for shift-click extending the selection. There's a bug now that it doesn't work the first time you try it. - Printing is needed. How hard can that be on Windows? FIRST CUT DONE - The python-mode trick of autoindenting a line with is neat and very handy. - (someday) a spellchecker for docstrings and comments. - a pagedown/up command key which moves to next class/def statement (top level) - split window capability - DnD text relocation/copying

Things I don't want to see. - line numbers... will probably slow things down way too much. - Please use another icon for the tree browser leaf. The small snake isn't cutting it.

- Customizable views (multi-window or multi-pane). (Markus Gritsch)
- Being able to double click (maybe double right click) on a callable object in the editor which shows the source of the object, if possible. (Gerrit Holl)
- Hooks into the guts, like in Emacs. (Mike Romberg)
- Sharing the editor with a remote tutor. (Martijn Faassen)
- Multiple views on the same file. (Tony J Ibbs)
- Store breakpoints in a global (per-project) database (GvR); Dirk Heise adds: save some space-trimmed context and search around when reopening a file that might have been edited by someone else.
- Capture menu events in extensions without changing the IDLE source. (Matthias Barmeier)
- Use overlapping panels (a "notebook" in MFC terms I think) for info that doesn't need to be accessible simultaneously (e.g. HTML source and output). Use multi-pane windows for info that does need to be shown together (e.g. class browser and source). (Albert Brandl)
- A project should invisibly track all symbols, for instant search, replace and cross-ref. Projects should be allowed to span multiple directories, hosts, etc. Project management files are placed in a directory you specify. A global mapping between project names and project directories should exist [not so sure --GvR]. (Tim Peters)
- Merge attr-tips and auto-expand. (Mark Hammond, Tim Peters)
- Python Shell should behave more like a "shell window" as users know it -- i.e. you can only edit the current command, and the cursor can't escape from the command area. (Albert Brandl)
- Set X11 class to "idle/Idle", set icon and title to something beginning with "idle" -- for window manangers. (Randall Hopper)
- Config files editable through a preferences dialog. (me) DONE

- Config files still editable outside the preferences dialog. (Randall Hopper) DONE
- When you're editing a command in PyShell, and there are only blank lines below the cursor, hitting Return should ignore or delete those blank lines rather than deciding you're not on the last line. (me)
- Run command (F5 c.s.) should be more like Pythonwin's Run -- a dialog with options to give command line arguments, run the debugger, etc. (me)
- Shouldn't be able to delete part of the prompt (or any text before it) in the PyShell. (Martijn Faassen) DONE
- Emacs style auto-fill (also smart about comments and strings). (Jeremy Hylton)
- Output of Run Script should go to a separate output window, not to the shell window. Output of separate runs should all go to the same window but clearly delimited. (David Scherer) REJECT FIRST, LATTER DONE
- GUI form designer to kick VB's butt. (Robert Geiger) THAT'S NOT IDLE
- Printing! Possibly via generation of PDF files which the user must then send to the printer separately. (Dinu Gherman) FIRST CUT

Grammar for 2to3. This grammar supports Python 2.x and 3.x.

Note: Changing the grammar specified in this file will most likely require corresponding changes in the parser module (../Modules/parsermodule.c). If you can't make the changes to that module yourself, please co-ordinate the required changes with someone who can; ask around on python-dev for help. Fred Drake fdrake at acm dot org will probably be listening there.

NOTE WELL: You should also follow all the steps listed in PEP 306, "How to Change Python's Grammar"

Commands for Kees Blom's railroad program

diagram:token NAME

diagram:token NUMBER

diagram:token STRING

diagram:token NEWLINE

diagram:token ENDMARKER

diagram:token INDENT

diagram:outputpython.bla

diagram:token DEDENT

diagram:output20.04cm0.0cm0.0cm

diagram:rules

Start symbols for the grammar:

file_input is a module or sequence of commands read from an input file;

single_input is a single interactive statement;

eval_input is the input for the eval() and input() functions.

NB: compound_stmt in single_input is followed by extra NEWLINE!

file_input: (NEWLINE | stmt)* ENDMARKER single_input: NEWLINE | simple_stmt | compound_stmt NEWLINE eval_input: testlist NEWLINE* ENDMARKER

decorator: '@' dotted_name ['(' [arglist] ')'] NEWLINE decorators: decorator+ decorated: decorators (classdef | funcdef) funcdef: 'def' NAME parameters ['->' test] ':' suite parameters: '(' [typedargslist] ')' typedargslist: ((tfpdef ['=' test] ',') *('* [tname] (',' tname ['=' test])* [',' '**' tname] | **'' tname) | tfpdef ['=' test] (',' tfpdef ['=' test])* [',']) tname: NAME [':' test] tfpdef: tname | '(' tfplist ')' tfplist: tfpdef (',' tfpdef)* [',']** *varargslist: ((vfpdef ['=' test] ',') ('* [vname] (',' vname ['=' test])* [',' '**' vname] | '' vname) | vfpdef ['=' test] (',' vfpdef ['=' test])* [',']) vname: NAME vfpdef: vname | '(' vfplist ')' vfplist: vfpdef (',' vfpdef)* [',']

stmt: simple_stmt | compound_stmt simple_stmt: small_stmt (';' small_stmt)* [';'] NEWLINE small_stmt: (expr_stmt | print_stmt | del_stmt | pass_stmt | flow_stmt | import_stmt | global_stmt | exec_stmt | assert_stmt) expr_stmt: testlist_star_expr (augassign (yield_expr|testlist) | ('=' (yield_expr|testlist_star_expr))*) *testlist_star_expr: (test|star_expr) (',' (test|star_expr))* [','] augassign: ('+=' | '-=' | '*=' | '@=' | '/=' | '%=' | '&=' | '|=' | '^=' | '<<=' | '>>=' | '**=' | '//=') # For normal assignments, additional restrictions enforced by the interpreter print_stmt: 'print' ([test (',' test)* [',']] | '>>' test [(',' test)+ [',']]) del_stmt: 'del' exprlist pass_stmt: 'pass' flow_stmt: break_stmt | continue_stmt | return_stmt | raise_stmt | yield_stmt break_stmt: 'break' continue_stmt: 'continue' return_stmt: 'return' [testlist] yield_stmt: yield_expr raise_stmt: 'raise' [test ['from' test | ',' test [',' test]]] import_stmt: import_name | import_from import_name: 'import' dotted_as_names import_from: ('from' ('.'* dotted_name | '.'+) 'import' ('' | '(' *import_as_names* ')' | *import_as_names*)) *import_as_name: NAME ['as' NAME] dotted_as_name: dotted_name ['as' NAME] import_as_names: import_as_name (',' import_as_name)* [','] dotted_as_names: dotted_as_name (',' dotted_as_name)* dotted_name: NAME ('.' NAME)* global_stmt: ('global' | 'nonlocal') NAME (',' NAME)* exec_stmt: 'exec' expr ['in' test [',' test]] assert_stmt: 'assert' test [',' test]*

compound_stmt: if_stmt | while_stmt | for_stmt | try_stmt | with_stmt | funcdef | classdef | decorated if_stmt: 'if' test ':' suite ('elif' test ':' suite)* ['else' ':' suite] while_stmt: 'while' test ':' suite ['else' ':' suite] for_stmt: 'for' exprlist 'in' testlist ':' suite ['else' ':' suite] try_stmt: ('try' ':' suite ((except_clause ':' suite)+ ['else' ':' suite] ['finally' ':' suite] | 'finally' ':' suite)) with_stmt: 'with' with_item (',' with_item)* ':' suite with_item: test ['as' expr] with_var: 'as' expr # NB compile.c makes sure that the default except clause is last except_clause: 'except' [test [(',' | 'as') test]] suite: simple_stmt | NEWLINE INDENT stmt+ DEDENT

Backward compatibility cruft to support:

[x for x in lambda: True, lambda: False if x()]

even while also allowing:

lambda x: 5 if x else 2

(But not a mix of the two)

testlist_safe: old_test [(',' old_test)+ [',']] old_test: or_test | old_lambdef old_lambdef: 'lambda' [varargslist] ':' old_test

test: or_test ['if' or_test 'else' test] | lambdef or_test: and_test ('or' and_test) *and_test: not_test ('and' not_test)* not_test: 'not' not_test | comparison comparison: expr (comp_op expr) *comp_op:* '<'|'>'|'=='|'>='|'<='|'<>'|'!='|'in'|'not' 'in'|'is'|'is' 'not' *star_expr:* '' expr expr: xor_expr ('|' xor_expr) *xor_expr: and_expr ('^' and_expr)* and_expr: shift_expr ('&' shift_expr) *shift_expr: arith_expr (('<<'|'>>') arith_expr)* arith_expr: term (('+'|'-') term) *term: factor (('*'|'@'|'/'|'%'|'//') factor) factor: ('+'|'-'|'~') factor* | *power power: atom trailer* ['**' factor] atom: ('(' [yield_expr|testlist_gexp] ')' | '[' [listmaker] ']' | '{' [dictsetmaker] '}' | `'' testlist1 ''` | NAME | NUMBER | STRING+ | '.' '.' '.') listmaker: (test|star_expr) (comp_for | (',' (test|star_expr))* [',']) testlist_gexp: (test|star_expr) (comp_for | (',' (test|star_expr))* [',']) lambdef: 'lambda' [varargslist] ':' test trailer: '(' [arglist] ')' | '[' subscriptlist ']' | '.' NAME subscriptlist: subscript (',' subscript)* [','] subscript: test | [test] ':' [test] [sliceop] sliceop: ':' [test] exprlist: (expr|star_expr) (',' (expr|star_expr))* [','] testlist: test (',' test)* [','] dictsetmaker: ((test ':' test (comp_for | (',' test ':' test)* [','])) | (test (comp_for | (',' test)* [','])))

classdef: 'class' NAME ['(' [arglist] ')'] ':' suite

arglist: (argument ',')* (argument [','] |'' *test (',' argument)* [',' '**' test] |'**' test) argument: test [comp_for] | test '=' test # Really [keyword '='] test

comp_iter: comp_for | comp_if comp_for: 'for' exprlist 'in' testlist_safe [comp_iter] comp_if: 'if' old_test [comp_iter]

testlist1: test (',' test)*

not used in grammar, but may appear in "node" passed from Parser to Compiler

encoding_decl: NAME

yield_expr: 'yield' [yield_arg] yield_arg: 'from' test | testlist

A grammar to describe tree matching patterns.

Not shown here:

- 'TOKEN' stands for any token (leaf node)

- 'any' stands for any node (leaf or interior)

With 'any' we can still specify the sub-structure.

The start symbol is 'Matcher'.

Matcher: Alternatives ENDMARKER

Alternatives: Alternative ('|' Alternative)*

Alternative: (Unit | NegatedUnit)+

Unit: [NAME '='] (STRING [Repeater] | NAME [Details] [Repeater] | '(' Alternatives ')' [Repeater] | '[' Alternatives ']')

NegatedUnit: 'not' (STRING | NAME [Details] | '(' Alternatives ')')

Repeater: '*' | '+' | '{' NUMBER [',' NUMBER] '}'

Details: '<' Alternatives '>'

如何在 Python 中使用既有的 C library?　在資訊科技快速發展的今天, 開發及測試軟體的速度是不容忽視的 課題. 為加快開發及測試的速度, 我們便常希望能利用一些已開發好的 library, 並有一個 fast prototyping 的 programming language 可 供使用. 目前有許許多多的 library 是以 C 寫成, 而 Python 是一個 fast prototyping 的 programming language. 故我們希望能將既有的 C library 拿到 Python 的環境中測試及整合. 其中最主要也是我們所 要討論的問題就是:

ļ Ě鴿⋈泜 ÊÊê êê

똠방각하 펠시콜라

㉷㉷납!! 因九月패밀륀궈 ⓡⓖ횤¿¿¿ 긍뒙 ⓔ뎨 ㉷. . 쬬영ⓔ능횹 . . . . 서울뤄 뎐학乙 家횤 ! ! !ㅠ.ㅠ 흐흐흐 ㄱㄱㄱ☆ㅠ_ㅠ 어릨 탸콰긐 뎌응 칑九들乙 ㉷드긐 설릌 家횤 . . . . 굴애쉌 ⓔ궈 ⓡ릘㉭긐 因仁川女中까짘 와쒀횤 ! ! 쬬영ⓔ 家능궈 ☆上관 없능궈능 쬬능뒈횤 글애듴 ⓡ려듀九 싀풔숴횤 어릨 因仁川女中싁⑨들앜!! ㉷㉷납♡ ∩∩*

Python の開発は、1990 年ごろから開始されています。 開発者の Guido van Rossum は教育用のプログラミング言語「ABC」の開発に参加していましたが、ABC は実用上の目的にはあまり適していませんでした。 このため、Guido はより実用的なプログラミング言語の開発を開始し、英国 BBS 放送のコメディ番組「モンティ パイソン」のファンである Guido はこの言語を「Python」と名づけました。 このような背景から生まれた Python の言語設計は、「シンプル」で「習得が容易」という目標に重点が置かれています。 多くのスクリプト系言語ではユーザの目先の利便性を優先して色々な機能を言語要素として取り入れる場合が多いのですが、Python ではそういった小細工が追加されることはあまりありません。 言語自体の機能は最小限に押さえ、必要な機能は拡張モジュールとして追加する、というのが Python のポリシーです。

ノか ゜ト ゜トキ喝塀 妛麹 麈鮹餐

Python の開発は、1990 年ごろから開始されています。 開発者の Guido van Rossum は教育用のプログラミング言語「ABC」の開発に参加していましたが、ABC は実用上の目的にはあまり適していませんでした。 このため、Guido はより実用的なプログラミング言語の開発を開始し、英国 BBS 放送のコメディ番組「モンティ パイソン」のファンである Guido はこの言語を「Python」と名づけました。 このような背景から生まれた Python の言語設計は、「シンプル」で「習得が容易」という目標に重点が置かれています。 多くのスクリプト系言語ではユーザの目先の利便性を優先して色々な機能を言語要素として取り入れる場合が多いのですが、Python ではそういった小細工が追加されることはあまりありません。 言語自体の機能は最小限に押さえ、必要な機能は拡張モジュールとして追加する、というのが Python のポリシーです。

Python（派森）语言是一种功能强大而完善的通用型计算机程序设计语言， 已经具有十多年的发展历史，成熟且稳定。这种语言具有非常简捷而清晰 的语法特点，适合完成各种高层任务，几乎可以在所有的操作系统中 运行。这种语言简单而强大，适合各种人士学习使用。目前，基于这 种语言的相关技术正在飞速的发展，用户数量急剧扩大，相关的资源非常多。 如何在 Python 中使用既有的 C library? 在資訊科技快速發展的今天, 開發及測試軟體的速度是不容忽視的 課題. 為加快開發及測試的速度, 我們便常希望能利用一些已開發好的 library, 並有一個 fast prototyping 的 programming language 可 供使用. 目前有許許多多的 library 是以 C 寫成, 而 Python 是一個 fast prototyping 的 programming language. 故我們希望能將既有的 C library 拿到 Python 的環境中測試及整合. 其中最主要也是我們所 要討論的問題就是:

This sentence is in ASCII. The next sentence is in GB.己所不欲，勿施於人。Bye.

This sentence is in ASCII. The next sentence is in GB.{<:Ky2;S{#,NpJ)l6HK!#}Bye.

Python の開発は、1990 年ごろから開始されています。 開発者の Guido van Rossum は教育用のプログラミング言語「ABC」の開発に参加していましたが、ABC は実用上の目的にはあまり適していませんでした。 このため、Guido はより実用的なプログラミング言語の開発を開始し、英国 BBS 放送のコメディ番組「モンティ パイソン」のファンである Guido はこの言語を「Python」と名づけました。 このような背景から生まれた Python の言語設計は、「シンプル」で「習得が容易」という目標に重点が置かれています。 多くのスクリプト系言語ではユーザの目先の利便性を優先して色々な機能を言語要素として取り入れる場合が多いのですが、Python ではそういった小細工が追加されることはあまりありません。 言語自体の機能は最小限に押さえ、必要な機能は拡張モジュールとして追加する、というのが Python のポリシーです。

$)C!] FD@L=c(Python)@: 9h?l1b =10m, 0-7BGQ GA7N1W7!9V >p>n@T4O4Y. FD@L=c@: H?@2@{@N 0mF(iPd:)GQ 9.9}0z 5?@{ E8@LGN, 1W8.0m @NEMGA8.FC H/0f@: FD@L=c@; =:E)83FC0z ?)7/ :P>_?!<-?M 4k:N:P@G GC7'F{?!<@G :|8% >VGC8.DI@Lp>n7N 885i>nA]4O4Y.

!YC90!3!: 3/>F6s >1~ E-! 1]>x@L @|4O4Y. 1W710E 4Y.

똠방각하 펲시콜라

㉷㉷납!! 因九月패밀륀궈 ⓡⓖ횤¿¿¿ 긍뒙 ⓔ뎨 ㉷. . 쬬영ⓔ능횹 . . . . 서울뤄 뎐학乙 家횤 ! ! !ㅠ.ㅠ 흐흐흐 ㄱㄱㄱ☆ㅠ_ㅠ 어릨 탸콰긐 뎌응 칑九들乙 ㉷드긐 설릌 家횤 . . . . 굴애쉌 ⓔ궈 ⓡ릘㉭긐 因仁川女中까짘 와쒀횤 ! ! 쬬영ⓔ 家능궈 ☆上관 없능궈능 쬬능뒈횤 글애듴 ⓡ려듀九 싀풔숴횤 어릨 因仁川女中싁⑨들앜!! ㉷㉷납♡ ∩∩*

-- 'f' code formatting, with explicit precision (>= 0). Output always -- has the given number of places after the point; zeros are added if -- necessary to make this true.

-- zeros %.0f 0 -> 0 %.1f 0 -> 0.0 %.2f 0 -> 0.00 %.3f 0 -> 0.000 %.50f 0 -> 0.00000000000000000000000000000000000000000000000000

-- precision 0; result should never include a . %.0f 1.5 -> 2 %.0f 2.5 -> 2 %.0f 3.5 -> 4 %.0f 0.0 -> 0 %.0f 0.1 -> 0 %.0f 0.001 -> 0 %.0f 10.0 -> 10 %.0f 10.1 -> 10 %.0f 10.01 -> 10 %.0f 123.456 -> 123 %.0f 1234.56 -> 1235 %.0f 1e49 -> 9999999999999999464902769475481793196872414789632 %.0f 9.9999999999999987e+49 -> 99999999999999986860582406952576489172979654066176 %.0f 1e50 -> 100000000000000007629769841091887003294964970946560

-- precision 1 %.1f 0.0001 -> 0.0 %.1f 0.001 -> 0.0 %.1f 0.01 -> 0.0 %.1f 0.04 -> 0.0 %.1f 0.06 -> 0.1 %.1f 0.25 -> 0.2 %.1f 0.75 -> 0.8 %.1f 1.4 -> 1.4 %.1f 1.5 -> 1.5 %.1f 10.0 -> 10.0 %.1f 1000.03 -> 1000.0 %.1f 1234.5678 -> 1234.6 %.1f 1234.7499 -> 1234.7 %.1f 1234.75 -> 1234.8

-- precision 2 %.2f 0.0001 -> 0.00 %.2f 0.001 -> 0.00 %.2f 0.004999 -> 0.00 %.2f 0.005001 -> 0.01 %.2f 0.01 -> 0.01 %.2f 0.125 -> 0.12 %.2f 0.375 -> 0.38 %.2f 1234500 -> 1234500.00 %.2f 1234560 -> 1234560.00 %.2f 1234567 -> 1234567.00 %.2f 1234567.8 -> 1234567.80 %.2f 1234567.89 -> 1234567.89 %.2f 1234567.891 -> 1234567.89 %.2f 1234567.8912 -> 1234567.89

-- alternate form always includes a decimal point. This only -- makes a difference when the precision is 0. %#.0f 0 -> 0. %#.1f 0 -> 0.0 %#.0f 1.5 -> 2. %#.0f 2.5 -> 2. %#.0f 10.1 -> 10. %#.0f 1234.56 -> 1235. %#.1f 1.4 -> 1.4 %#.2f 0.375 -> 0.38

-- if precision is omitted it defaults to 6 %f 0 -> 0.000000 %f 1230000 -> 1230000.000000 %f 1234567 -> 1234567.000000 %f 123.4567 -> 123.456700 %f 1.23456789 -> 1.234568 %f 0.00012 -> 0.000120 %f 0.000123 -> 0.000123 %f 0.00012345 -> 0.000123 %f 0.000001 -> 0.000001 %f 0.0000005001 -> 0.000001 %f 0.0000004999 -> 0.000000

-- 'e' code formatting with explicit precision (>= 0). Output should -- always have exactly the number of places after the point that were -- requested.

-- zeros %.0e 0 -> 0e+00 %.1e 0 -> 0.0e+00 %.2e 0 -> 0.00e+00 %.10e 0 -> 0.0000000000e+00 %.50e 0 -> 0.00000000000000000000000000000000000000000000000000e+00

-- precision 0. no decimal point in the output %.0e 0.01 -> 1e-02 %.0e 0.1 -> 1e-01 %.0e 1 -> 1e+00 %.0e 10 -> 1e+01 %.0e 100 -> 1e+02 %.0e 0.012 -> 1e-02 %.0e 0.12 -> 1e-01 %.0e 1.2 -> 1e+00 %.0e 12 -> 1e+01 %.0e 120 -> 1e+02 %.0e 123.456 -> 1e+02 %.0e 0.000123456 -> 1e-04 %.0e 123456000 -> 1e+08 %.0e 0.5 -> 5e-01 %.0e 1.4 -> 1e+00 %.0e 1.5 -> 2e+00 %.0e 1.6 -> 2e+00 %.0e 2.4999999 -> 2e+00 %.0e 2.5 -> 2e+00 %.0e 2.5000001 -> 3e+00 %.0e 3.499999999999 -> 3e+00 %.0e 3.5 -> 4e+00 %.0e 4.5 -> 4e+00 %.0e 5.5 -> 6e+00 %.0e 6.5 -> 6e+00 %.0e 7.5 -> 8e+00 %.0e 8.5 -> 8e+00 %.0e 9.4999 -> 9e+00 %.0e 9.5 -> 1e+01 %.0e 10.5 -> 1e+01 %.0e 14.999 -> 1e+01 %.0e 15 -> 2e+01

-- precision 1 %.1e 0.0001 -> 1.0e-04 %.1e 0.001 -> 1.0e-03 %.1e 0.01 -> 1.0e-02 %.1e 0.1 -> 1.0e-01 %.1e 1 -> 1.0e+00 %.1e 10 -> 1.0e+01 %.1e 100 -> 1.0e+02 %.1e 120 -> 1.2e+02 %.1e 123 -> 1.2e+02 %.1e 123.4 -> 1.2e+02

-- precision 2 %.2e 0.00013 -> 1.30e-04 %.2e 0.000135 -> 1.35e-04 %.2e 0.0001357 -> 1.36e-04 %.2e 0.0001 -> 1.00e-04 %.2e 0.001 -> 1.00e-03 %.2e 0.01 -> 1.00e-02 %.2e 0.1 -> 1.00e-01 %.2e 1 -> 1.00e+00 %.2e 10 -> 1.00e+01 %.2e 100 -> 1.00e+02 %.2e 1000 -> 1.00e+03 %.2e 1500 -> 1.50e+03 %.2e 1590 -> 1.59e+03 %.2e 1598 -> 1.60e+03 %.2e 1598.7 -> 1.60e+03 %.2e 1598.76 -> 1.60e+03 %.2e 9999 -> 1.00e+04

-- omitted precision defaults to 6 %e 0 -> 0.000000e+00 %e 165 -> 1.650000e+02 %e 1234567 -> 1.234567e+06 %e 12345678 -> 1.234568e+07 %e 1.1 -> 1.100000e+00

-- alternate form always contains a decimal point. This only makes -- a difference when precision is 0.

%#.0e 0.01 -> 1.e-02 %#.0e 0.1 -> 1.e-01 %#.0e 1 -> 1.e+00 %#.0e 10 -> 1.e+01 %#.0e 100 -> 1.e+02

%#.0e 0.012 -> 1.e-02 %#.0e 0.12 -> 1.e-01 %#.0e 1.2 -> 1.e+00 %#.0e 12 -> 1.e+01 %#.0e 120 -> 1.e+02 %#.0e 123.456 -> 1.e+02 %#.0e 0.000123456 -> 1.e-04 %#.0e 123456000 -> 1.e+08 %#.0e 0.5 -> 5.e-01 %#.0e 1.4 -> 1.e+00 %#.0e 1.5 -> 2.e+00 %#.0e 1.6 -> 2.e+00 %#.0e 2.4999999 -> 2.e+00 %#.0e 2.5 -> 2.e+00 %#.0e 2.5000001 -> 3.e+00 %#.0e 3.499999999999 -> 3.e+00 %#.0e 3.5 -> 4.e+00 %#.0e 4.5 -> 4.e+00 %#.0e 5.5 -> 6.e+00 %#.0e 6.5 -> 6.e+00 %#.0e 7.5 -> 8.e+00 %#.0e 8.5 -> 8.e+00 %#.0e 9.4999 -> 9.e+00 %#.0e 9.5 -> 1.e+01 %#.0e 10.5 -> 1.e+01 %#.0e 14.999 -> 1.e+01 %#.0e 15 -> 2.e+01 %#.1e 123.4 -> 1.2e+02 %#.2e 0.0001357 -> 1.36e-04

-- 'g' code formatting.

-- zeros %.0g 0 -> 0 %.1g 0 -> 0 %.2g 0 -> 0 %.3g 0 -> 0 %.4g 0 -> 0 %.10g 0 -> 0 %.50g 0 -> 0 %.100g 0 -> 0

-- precision 0 doesn't make a lot of sense for the 'g' code (what does -- it mean to have no significant digits?); in practice, it's interpreted -- as identical to precision 1 %.0g 1000 -> 1e+03 %.0g 100 -> 1e+02 %.0g 10 -> 1e+01 %.0g 1 -> 1 %.0g 0.1 -> 0.1 %.0g 0.01 -> 0.01 %.0g 1e-3 -> 0.001 %.0g 1e-4 -> 0.0001 %.0g 1e-5 -> 1e-05 %.0g 1e-6 -> 1e-06 %.0g 12 -> 1e+01 %.0g 120 -> 1e+02 %.0g 1.2 -> 1 %.0g 0.12 -> 0.1 %.0g 0.012 -> 0.01 %.0g 0.0012 -> 0.001 %.0g 0.00012 -> 0.0001 %.0g 0.000012 -> 1e-05 %.0g 0.0000012 -> 1e-06

-- precision 1 identical to precision 0 %.1g 1000 -> 1e+03 %.1g 100 -> 1e+02 %.1g 10 -> 1e+01 %.1g 1 -> 1 %.1g 0.1 -> 0.1 %.1g 0.01 -> 0.01 %.1g 1e-3 -> 0.001 %.1g 1e-4 -> 0.0001 %.1g 1e-5 -> 1e-05 %.1g 1e-6 -> 1e-06 %.1g 12 -> 1e+01 %.1g 120 -> 1e+02 %.1g 1.2 -> 1 %.1g 0.12 -> 0.1 %.1g 0.012 -> 0.01 %.1g 0.0012 -> 0.001 %.1g 0.00012 -> 0.0001 %.1g 0.000012 -> 1e-05 %.1g 0.0000012 -> 1e-06

-- precision 2 %.2g 1000 -> 1e+03 %.2g 100 -> 1e+02 %.2g 10 -> 10 %.2g 1 -> 1 %.2g 0.1 -> 0.1 %.2g 0.01 -> 0.01 %.2g 0.001 -> 0.001 %.2g 1e-4 -> 0.0001 %.2g 1e-5 -> 1e-05 %.2g 1e-6 -> 1e-06 %.2g 1234 -> 1.2e+03 %.2g 123 -> 1.2e+02 %.2g 12.3 -> 12 %.2g 1.23 -> 1.2 %.2g 0.123 -> 0.12 %.2g 0.0123 -> 0.012 %.2g 0.00123 -> 0.0012 %.2g 0.000123 -> 0.00012 %.2g 0.0000123 -> 1.2e-05

-- bad cases from http://bugs.python.org/issue9980 %.12g 38210.0 -> 38210 %.12g 37210.0 -> 37210 %.12g 36210.0 -> 36210

-- alternate g formatting: always include decimal point and -- exactly significant digits. %#.0g 0 -> 0. %#.1g 0 -> 0. %#.2g 0 -> 0.0 %#.3g 0 -> 0.00 %#.4g 0 -> 0.000

%#.0g 0.2 -> 0.2 %#.1g 0.2 -> 0.2 %#.2g 0.2 -> 0.20 %#.3g 0.2 -> 0.200 %#.4g 0.2 -> 0.2000 %#.10g 0.2 -> 0.2000000000

%#.0g 2 -> 2. %#.1g 2 -> 2. %#.2g 2 -> 2.0 %#.3g 2 -> 2.00 %#.4g 2 -> 2.000

%#.0g 20 -> 2.e+01 %#.1g 20 -> 2.e+01 %#.2g 20 -> 20. %#.3g 20 -> 20.0 %#.4g 20 -> 20.00

%#.0g 234.56 -> 2.e+02 %#.1g 234.56 -> 2.e+02 %#.2g 234.56 -> 2.3e+02 %#.3g 234.56 -> 235. %#.4g 234.56 -> 234.6 %#.5g 234.56 -> 234.56 %#.6g 234.56 -> 234.560

-- repr formatting. Result always includes decimal point and at -- least one digit after the point, or an exponent. %r 0 -> 0.0 %r 1 -> 1.0

%r 0.01 -> 0.01 %r 0.02 -> 0.02 %r 0.03 -> 0.03 %r 0.04 -> 0.04 %r 0.05 -> 0.05

-- values >= 1e16 get an exponent %r 10 -> 10.0 %r 100 -> 100.0 %r 1e15 -> 1000000000000000.0 %r 9.999e15 -> 9999000000000000.0 %r 9999999999999998 -> 9999999999999998.0 %r 9999999999999999 -> 1e+16 %r 1e16 -> 1e+16 %r 1e17 -> 1e+17

-- as do values < 1e-4 %r 1e-3 -> 0.001 %r 1.001e-4 -> 0.0001001 %r 1.0000000000000001e-4 -> 0.0001 %r 1.000000000000001e-4 -> 0.0001000000000000001 %r 1.00000000001e-4 -> 0.000100000000001 %r 1.0000000001e-4 -> 0.00010000000001 %r 1e-4 -> 0.0001 %r 0.99999999999999999e-4 -> 0.0001 %r 0.9999999999999999e-4 -> 9.999999999999999e-05 %r 0.999999999999e-4 -> 9.99999999999e-05 %r 0.999e-4 -> 9.99e-05 %r 1e-5 -> 1e-05

Python IEEE 754 floating point support

```
from sys import float_info as FI from math import * PI = pi E = e
```

You must never compare two floats with == because you are not going to get what you expect. We treat two floats as equal if the difference between them is small than epsilon. >>> EPS = 1E-15 >>> def equal(x, y): ... """Almost equal helper for floats""" ... return abs(x - y) < EPS

NaNs and INFs

In Python 2.6 and newer NaNs (not a number) and infinity can be constructed from the strings 'inf' and 'nan'.

    INF = float('inf') NINF = float('-inf') NAN = float('nan')

    INF inf NINF -inf NAN nan

The math module's `isnan` and `isinf` functions can be used to detect INF and NAN: >>> isinf(INF), isinf(NINF), isnan(NAN) (True, True, True) >>> INF == -NINF True

Infinity

Ambiguous operations like `0 * inf` or `inf - inf` result in NaN. >>> INF * 0 nan >>> INF - INF nan >>> INF / INF nan

However unambigous operations with inf return inf: >>> INF * INF inf >>> 1.5 * INF inf >>> 0.5 * INF inf >>> INF / 1000 inf

Not a Number

NaNs are never equal to another number, even itself >>> NAN == NAN False >>> NAN < 0 False >>> NAN >= 0 False

All operations involving a NaN return a NaN except for nan**0 and 1**nan. >>> 1 + NAN nan >>> 1 * NAN nan >>> 0 * NAN nan >>> 1 ** NAN 1.0 >>> NAN ** 0 1.0 >>> 0 ** NAN nan >>> (1.0 + FI.epsilon) * NAN nan

Misc Functions

The power of 1 raised to x is always 1.0, even for special values like 0, infinity and NaN.

> pow(1, 0) 1.0 pow(1, INF) 1.0 pow(1, -INF) 1.0 pow(1, NAN) 1.0

The power of 0 raised to x is defined as 0, if x is positive. Negative values are a domain error or zero division error and NaN result in a silent NaN.

> pow(0, 0) 1.0 pow(0, INF) 0.0 pow(0, -INF) Traceback (most recent call last): ... ValueError: math domain error 0 ** -1 Traceback (most recent call last): ... ZeroDivisionError: 0.0 cannot be raised to a negative power pow(0, NAN) nan

Trigonometric Functions

sin(INF) Traceback (most recent call last): ... ValueError: math domain error sin(NINF) Traceback (most recent call last): ... ValueError: math domain error sin(NAN) nan cos(INF) Traceback (most recent call last): ... ValueError: math domain error cos(NINF) Traceback (most recent call last): ... ValueError: math domain error cos(NAN) nan tan(INF) Traceback (most recent call last): ... ValueError: math domain error tan(NINF) Traceback (most recent call last): ... ValueError: math domain error tan(NAN) nan

Neither pi nor tan are exact, but you can assume that tan(pi/2) is a large value and tan(pi) is a very small value: >>> tan(PI/2) > 1E10 True >>> -tan(-PI/2) > 1E10 True >>> tan(PI) < 1E-15 True

asin(NAN), acos(NAN), atan(NAN) (nan, nan, nan) asin(INF), asin(NINF) Traceback (most recent call last): ... ValueError: math domain error acos(INF), acos(NINF) Traceback (most recent call last): ... ValueError: math domain error equal(atan(INF), PI/2), equal(atan(NINF), -PI/2) (True, True)

Hyberbolic Functions

Mailcap file for test_mailcap; based on RFC 1524

Referred to by test_mailcap.py

This is a comment.

application/frame; showframe %s; print="cat %s | lp" application/postscript; ps-to-terminal %s;
needsterminal application/postscript; ps-to-terminal %s;
compose=idraw %s application/x-dvi; xdvi %s application/x-movie; movieplayer %s; compose=moviemaker
%s;
description="Movie";
x11-bitmap="/usr/lib/Zmail/bitmaps/movie.xbm" application/*; echo "This is "%t" but
is 50 % Greek to me" ; cat %s; copiousoutput

audio/basic; showaudio %s; compose=audiocompose %s; edit=audiocompose %s;
description="An audio fragment" audio/* ; /usr/local/bin/showaudio %t

image/rgb; display %s #image/gif; display %s image/x-xwindowdump; display %s

The continuation char shouldn't
make a difference in a comment.

message/external-body; showexternal %s %{access-type} %{name} %{site}
%{directory} %{mode} %{server}; needsterminal; composetyped = extcompose %s;
description="A reference to data stored in an external location"

text/richtext; shownonascii iso-8859-8 -e richtext -p %s; test=test "echo \ %{charset} | tr '[A-Z]' '[a-z]'" = iso-8859-8; copiousoutput

video/mpeg; mpeg_play %s video/*; animate %s

This is a sample doctest in a text file.

In this example, we'll rely on a global variable being set for us already:

    favorite_color 'blue'

We can make this fail by disabling the blank-line feature.

    if 1: ... print('a') ... print() ... print('b') a b

Here we check that `__file__` is provided:

type(**file**)

This is a sample doctest in a text file that contains non-ASCII characters. This file is encoded using UTF-8.

In order to get this test to pass, we have to manually specify the encoding.

    'föö' 'f66'

    'bąr' 'b105r'

Return-Path: bbb at zzz dot org Delivered-To: bbb@zzz.org Received: by mail.zzz.org (Postfix, from userid 889) id 27CEAD38CC; Fri, 4 May 2001 14:05:44 -0400 (EDT) MIME-Version: 1.0 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit Message-ID: 15090.61304.110929.45684 at aaa dot zzz dot org From: bbb@ddd.com (John X. Doe) To: bbb@zzz.org Subject: This is a test message Date: Fri, 4 May 2001 14:05:44 -0400

Hi,

Do you like this message?

-Me

MIME-version: 1.0 From: ppp-request@zzz.org Sender: ppp-admin@zzz.org To: ppp@zzz.org Subject: Ppp digest, Vol 1 #2 - 5 msgs Date: Fri, 20 Apr 2001 20:18:00 -0400 (EDT) X-Mailer: Mailman v2.0.4 X-Mailman-Version: 2.0.4 Content-Type: multipart/mixed; boundary="192.168.1.2.889.32614.987812255.500.21814"

--192.168.1.2.889.32614.987812255.500.21814 Content-type: text/plain; charset=us-ascii Content-description: Masthead (Ppp digest, Vol 1 #2)

Send Ppp mailing list submissions to ppp@zzz.org

To subscribe or unsubscribe via the World Wide Web, visit http://www.zzz.org/mailman/listinfo/ppp or, via email, send a message with subject or body 'help' to ppp-request@zzz.org

You can reach the person managing the list at ppp-admin@zzz.org

When replying, please edit your Subject line so it is more specific than "Re: Contents of Ppp digest..."

--192.168.1.2.889.32614.987812255.500.21814 Content-type: text/plain; charset=us-ascii Content-description: Today's Topics (5 msgs)

Today's Topics:

1. testing #1 (Barry A. Warsaw)
2. testing #2 (Barry A. Warsaw)
3. testing #3 (Barry A. Warsaw)
4. testing #4 (Barry A. Warsaw)
5. testing #5 (Barry A. Warsaw)

--192.168.1.2.889.32614.987812255.500.21814 Content-Type: multipart/digest; boundary="----"

Message: 1 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit Date: Fri, 20 Apr 2001 20:16:13 -0400 To: ppp@zzz.org From: barry@digicool.com (Barry A. Warsaw) Subject: [Ppp] testing #1 Precedence: bulk

hello

Message: 2 Date: Fri, 20 Apr 2001 20:16:21 -0400 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit To: ppp@zzz.org From: barry@digicool.com (Barry A. Warsaw) Precedence: bulk

hello

Message: 3 Date: Fri, 20 Apr 2001 20:16:25 -0400 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit To: ppp@zzz.org From: barry@digicool.com (Barry A. Warsaw) Subject: [Ppp] testing #3 Precedence: bulk

hello

Message: 4 Date: Fri, 20 Apr 2001 20:16:28 -0400 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit To: ppp@zzz.org From: barry@digicool.com (Barry A. Warsaw) Subject: [Ppp] testing #4 Precedence: bulk

hello

Message: 5 Date: Fri, 20 Apr 2001 20:16:32 -0400 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit To: ppp@zzz.org From: barry@digicool.com (Barry A. Warsaw) Subject: [Ppp] testing #5 Precedence: bulk

hello

--192.168.1.2.889.32614.987812255.500.21814 Content-type: text/plain; charset=us-ascii Content-description: Digest Footer

---

Ppp mailing list Ppp@zzz.org http://www.zzz.org/mailman/listinfo/ppp

--192.168.1.2.889.32614.987812255.500.21814--

End of Ppp Digest

Return-Path: barry at python dot org Delivered-To: barry@python.org Received: by mail.python.org (Postfix, from userid 889) id C2BF0D37C6; Tue, 11 Sep 2001 00:05:05 -0400 (EDT) MIME-Version: 1.0 Content-Type: multipart/mixed; boundary="h90VIIIKmx" Content-Transfer-Encoding: 7bit Message-ID: 15261.36209.358846.118674 at anthem dot python dot org From: barry@python.org (Barry A. Warsaw) To: barry@python.org Subject: a simple multipart Date: Tue, 11 Sep 2001 00:05:05 -0400 X-Mailer: VM 6.95 under 21.4 (patch 4) "Artificial Intelligence" XEmacs Lucid X-Attribution: BAW X-Oblique-Strategy: Make a door into a window

--h90VIIIKmx Content-Type: text/plain Content-Disposition: inline; filename="msg.txt" Content-Transfer-Encoding: 7bit

a simple kind of mirror to reflect upon our own

--h90VIIIKmx Content-Type: text/plain Content-Disposition: inline; filename="msg.txt" Content-Transfer-Encoding: 7bit

a simple kind of mirror to reflect upon our own

--h90VIIIKmx--

MIME-Version: 1.0 From: Barry Warsaw barry at python dot org To: Dingus Lovers cravindogs at cravindogs dot com Subject: Lyrics Date: Fri, 20 Apr 2001 19:35:02 -0400 Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY Content-Type: text/plain; charset="us-ascii"

--BOUNDARY Content-Type: text/html; charset="iso-8859-1"

--BOUNDARY Content-Type: text/plain

--BOUNDARY Content-Type: text/plain; charset="koi8-r"

--BOUNDARY--

Return-Path: xx at xx dot dk Received: from fepD.post.tele.dk (195.41.46.149) by mail.groupcare.dk (LSMTP for Windows NT v1.1b) with SMTP id 0.0014F8A2 at mail dot groupcare dot dk; Mon, 30 Apr 2001 12:17:50 +0200 User-Agent: Microsoft-Outlook-Express-Macintosh-Edition/5.02.2106 Subject: XX From: xx@xx.dk To: XX Message-ID: Mime-version: 1.0 Content-type: multipart/mixed; boundary="MS_Mac_OE_3071477847_720252_MIME_Part"

Denne meddelelse er i MIME-format. Da dit postl

--MS_Mac_OE_3071477847_720252_MIME_Part Content-type: multipart/alternative; boundary="MS_Mac_OE_3071477847_720252_MIME_Part"

--MS_Mac_OE_3071477847_720252_MIME_Part Content-type: text/plain; charset="ISO-8859-1" Content-transfer-encoding: quoted-printable

Some removed test.

--MS_Mac_OE_3071477847_720252_MIME_Part Content-type: text/html; charset="ISO-8859-1" Content-transfer-encoding: quoted-printable

Some removed text.

--MS_Mac_OE_3071477847_720252_MIME_Part--

--MS_Mac_OE_3071477847_720252_MIME_Part Content-type: image/gif; name="xx.gif"; x-mac-creator="6F676C65"; x-mac-type="47494666" Content-disposition: attachment Content-transfer-encoding: base64

Some removed base64 encoded chars.

--MS_Mac_OE_3071477847_720252_MIME_Part--

Return-Path: <> Delivered-To: scr-admin@socal-raves.org Received: from cougar.noc.ucla.edu (cougar.noc.ucla.edu [169.232.10.18]) by babylon.socal-raves.org (Postfix) with ESMTP id CCC2C51B84 for scr-admin at socal-raves dot org; Sun, 23 Sep 2001 20:13:54 -0700 (PDT) Received: from sims-ms-daemon by cougar.noc.ucla.edu (Sun Internet Mail Server sims.3.5.2000.03.23.18.03.p10) id 0GK500B01D0B8Y at cougar dot noc dot ucla dot edu for scr-admin@socal-raves.org; Sun, 23 Sep 2001 20:14:35 -0700 (PDT) Received: from cougar.noc.ucla.edu (Sun Internet Mail Server sims.3.5.2000.03.23.18.03.p10) id 0GK500B01D0B8X at cougar dot noc dot ucla dot edu; Sun, 23 Sep 2001 20:14:35 -0700 (PDT) Date: Sun, 23 Sep 2001 20:14:35 -0700 (PDT) From: Internet Mail Delivery postmaster at ucla dot edu Subject: Delivery Notification: Delivery has failed To: scr-admin@socal-raves.org Message-id: 0GK500B04D0B8X at cougar dot noc dot ucla dot edu MIME-version: 1.0 Sender: scr-owner@socal-raves.org Errors-To: scr-owner@socal-raves.org X-BeenThere: scr@socal-raves.org X-Mailman-Version: 2.1a3 Precedence: bulk List-Help: mailto:scr-request@socal-raves.org?subject=help (scr-request at socal-raves dot org?subject=help) List-Post: mailto:scr@socal-raves.org (scr at socal-raves dot org) List-Subscribe: http://socal-raves.org/mailman/listinfo/scr, mailto:scr-request@socal-raves.org?subject=subscribe (scr-request at socal-raves dot org?subject=subscribe) List-Id: SoCal-Raves List-Unsubscribe: http://socal-raves.org/mailman/listinfo/scr, mailto:scr-request@socal-raves.org?subject=unsubscribe (scr-request at socal-raves dot org?subject=unsubscribe) List-Archive: http://socal-raves.org/mailman/private/scr/ Content-Type: multipart/report; boundary="Boundary_(ID_PGS2F2a+z+/jL7hupKgRhA)"

--Boundary_(ID_PGS2F2a+z+/jL7hupKgRhA) Content-type: text/plain; charset=ISO-8859-1

This report relates to a message you sent with the following header fields:

Message-id: 002001c144a6$8752e060$56104586 at oxy dot edu Date: Sun, 23 Sep 2001 20:10:55 -0700 From: "Ian T. Henry" henryi at oxy dot edu To: SoCal Raves scr at socal-raves dot org Subject: [scr] yeah for Ians!!

Your message cannot be delivered to the following recipients:

Recipient address: jangel1@cougar.noc.ucla.edu Reason: recipient reached disk quota

--Boundary_(ID_PGS2F2a+z+/jL7hupKgRhA) Content-type: message/DELIVERY-STATUS

Original-envelope-id: 0GK500B4HD0888@cougar.noc.ucla.edu Reporting-MTA: dns; cougar.noc.ucla.edu

Action: failed Status: 5.0.0 (recipient reached disk quota) Original-recipient: rfc822;jangel1@cougar.noc.ucla.edu Final-recipient: rfc822;jangel1@cougar.noc.ucla.edu

--Boundary_(ID_PGS2F2a+z+/jL7hupKgRhA) Content-type: MESSAGE/RFC822

Return-path: scr-admin@socal-raves.org Received: from sims-ms-daemon by cougar.noc.ucla.edu (Sun Internet Mail Server sims.3.5.2000.03.23.18.03.p10) id 0GK500B01D0B8X at cougar dot noc dot ucla dot edu; Sun, 23 Sep 2001 20:14:35 -0700 (PDT) Received: from panther.noc.ucla.edu by cougar.noc.ucla.edu (Sun Internet Mail Server sims.3.5.2000.03.23.18.03.p10) with ESMTP id 0GK500B4GD0888 at cougar dot noc dot ucla dot edu for jangel1@sims-ms-daemon; Sun, 23 Sep 2001 20:14:33 -0700 (PDT) Received: from babylon.socal-raves.org (ip-209-85-222-117.dreamhost.com [209.85.222.117]) by panther.noc.ucla.edu (8.9.1a/8.9.1) with ESMTP id UAA09793 for jangel1 at ucla dot edu; Sun, 23 Sep 2001 20:14:32 -0700 (PDT) Received: from babylon (localhost [127.0.0.1]) by babylon.socal-raves.org (Postfix) with ESMTP id D3B2951B70; Sun, 23 Sep 2001 20:13:47 -0700 (PDT) Received: by babylon.socal-raves.org (Postfix, from userid 60001) id A611F51B82; Sun, 23 Sep 2001 20:13:46 -0700 (PDT) Received: from tiger.cc.oxy.edu (tiger.cc.oxy.edu [134.69.3.112]) by babylon.socal-raves.org (Postfix) with ESMTP id ADA7351B70 for scr at socal-raves dot org; Sun, 23 Sep 2001 20:13:44 -0700 (PDT) Received: from ent (n16h86.dhcp.oxy.edu [134.69.16.86]) by tiger.cc.oxy.edu (8.8.8/8.8.8) with SMTP id UAA08100 for scr at socal-raves dot org; Sun, 23 Sep 2001 20:14:24 -0700 (PDT) Date: Sun, 23 Sep 2001 20:10:55 -0700 From: "Ian T. Henry" henryi at oxy dot edu Subject: [scr] yeah for Ians!! Sender: scr-admin@socal-raves.org To: SoCal Raves scr at socal-raves dot org Errors-to: scr-admin@socal-raves.org Message-id: 002001c144a6$8752e060$56104586 at oxy dot edu MIME-version: 1.0 X-Mailer: Microsoft Outlook Express 5.50.4522.1200 Content-type: text/plain; charset=us-ascii Precedence: bulk Delivered-to: scr-post@babylon.socal-raves.org Delivered-to: scr@socal-raves.org X-Converted-To-Plain-Text: from multipart/alternative by demime 0.98e X-Converted-To-Plain-Text: Alternative section used was text/plain X-BeenThere: scr@socal-raves.org X-Mailman-Version: 2.1a3

List-Help: mailto:scr-request@socal-raves.org?subject=help (scr-request at socal-raves dot org?subject=help) List-Post: mailto:scr@socal-raves.org (scr at socal-raves dot org) List-Subscribe: http://socal-raves.org/mailman/listinfo/scr, mailto:scr-request@socal-raves.org?subject=subscribe (scr-request at socal-raves dot org?subject=subscribe) List-Id: SoCal-Raves List-Unsubscribe: http://socal-raves.org/mailman/listinfo/scr, mailto:scr-request@socal-raves.org?subject=unsubscribe (scr-request at socal-raves dot org?subject=unsubscribe) List-Archive: http://socal-raves.org/mailman/private/scr/

I always love to find more Ian's that are over 3 years old!!

Ian _____ For event info, list questions, or to unsubscribe, see http://www.socal-raves.org/

--Boundary_(ID_PGS2F2a+z+/jL7hupKgRhA)--

MIME-Version: 1.0 From: Barry barry at digicool dot com To: Dingus Lovers cravindogs at cravindogs dot com Subject: Here is your dingus fish Date: Fri, 20 Apr 2001 19:35:02 -0400 Content-Type: multipart/mixed; boundary="BOUNDARY"

Hi there,

This is the dingus fish.

[Non-text (image/gif) part of message omitted, filename dingusfish.gif]

From: aperson@dom.ain To: bperson@dom.ain Subject: Test Content-Type: multipart/mixed; boundary="BOUNDARY"

MIME message --BOUNDARY Content-Type: text/plain; charset="us-ascii" MIME-Version: 1.0 Content-Transfer-Encoding: 7bit

One --BOUNDARY Content-Type: text/plain; charset="us-ascii" MIME-Version: 1.0 Content-Transfer-Encoding: 7bit

Two --BOUNDARY-- End of MIME message

Mime-Version: 1.0 Message-Id: Date: Tue, 16 Oct 2001 13:59:25 +0300 To: a@example.com From: b@example.com Content-Type: multipart/mixed; boundary="============*-1208892523*=============="

--============*-1208892523*============== Content-Type: text/plain; charset="us-ascii" ; format="flowed"

Text text text. --============*-1208892523*============== Content-Id: Content-Type: image/jpeg; name="wibble.JPG" ; x-mac-type="4A504547" ; x-mac-creator="474B4F4E" Content-Disposition: attachment; filename="wibble.JPG" Content-Transfer-Encoding: base64

/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAALCAXABIEBAREA
g6bCjjw/pIZSjO6FWFpIdjySOmCNrO7DBZibUXhTwtCixw+GtAijVdqxxaPp0aKvmGXa
qrbBQvms0mAMeYS/3iTV1dG0hHaRNK01XblnWxtVdjkHLMIgTyqnk9VB7CrP2KzIINpa
4O7I+zxYO9WV8jZg71Zlb+8rMDkEirAVQFAUAKAFAAAUAYAUDgADgY6DjpRtXj5RxjHA
4wQRj0wQCMdCAewpaKKK/9k= --============*-1208892523*============== Content-Id: Content-Type: image/jpeg; name="wibble2.JPG" ; x-mac-type="4A504547" ; x-mac-creator="474B4F4E" Content-Disposition: attachment; filename="wibble2.JPG" Content-Transfer-Encoding: base64

/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAALCAXABJ0BAREA
/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQA
W6NFJJBEkU10kKGTcWMDwxuU+0JHvk8qAtOpNwqSR0n8c3BlDyXHlqsUltHEiTvdXLxR
7vMiGDNJAJWkAMk8ZkCFp5G2oo5W++lNrbQtNfTQxJAuXlupz9oS4d5Y1W+E2XlWZJJE
Y7LWYQxTLE1zuMbfBPxw8X2fibVdIbSbI6nLZxX635t9TjtYreWR7WGKJTLJFFKSlozO 0ShxIXM43uC3/9k= --============*-1208892523*============== Content-Type: text/plain; charset="us-ascii" ; format="flowed"

Text text text. --============*-1208892523*==============--

Return-Path: aperson at dom dot ain Received: by mail.dom.ain (Postfix, from userid 889) id B9D0AD35DB; Tue, 4 Jun 2002 21:46:59 -0400 (EDT) Message-ID: 15613.28051.707126.569693 at dom dot ain Date: Tue, 4 Jun 2002 21:46:59 -0400 MIME-Version: 1.0 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit Subject: bug demonstration 12345678911234567892123456789312345678941234567895123456789612345678971234567898112345678 more text From: aperson@dom.ain (Anne P. Erson) To: bperson@dom.ain (Barney P. Erson)

test

From: aperson@dom.ain MIME-Version: 1.0 Content-Type: multipart/digest; boundary=BOUNDARY

--BOUNDARY

Content-Type: text/plain; charset=us-ascii To: aa@bb.org From: cc@dd.org Subject: ee

message 1

--BOUNDARY

Content-Type: text/plain; charset=us-ascii To: aa@bb.org From: cc@dd.org Subject: ee

message 2

--BOUNDARY--

Delivered-To: freebsd-isp@freebsd.org Date: Wed, 27 Sep 2000 11:11:09 -0500 From: Anne Person aperson at example dot com To: Barney Dude bdude at example dot com Subject: Re: Limiting Perl CPU Utilization... Mime-Version: 1.0 Content-Type: multipart/signed; micalg=*ansi-x3.4-1968''pgp-md5; protocol*=ansi-x3.4-1968''application%2Fpgp-signature; boundary*="ansi-x3.4-1968''EeQfGwPcQSOJBaQU" Content-Disposition: inline Sender: owner-freebsd-isp@FreeBSD.ORG Precedence: bulk X-Loop: FreeBSD.org

--EeQfGwPcQSOJBaQU Content-Type: text/plain; charset*=ansi-x3.4-1968''us-ascii Content-Disposition: inline Content-Transfer-Encoding: quoted-printable

part 1

--EeQfGwPcQSOJBaQU Content-Type: text/plain Content-Disposition: inline

part 2

--EeQfGwPcQSOJBaQU--

Content-Type: multipart/mixed; boundary=ABCDE

--ABCDE Content-Type: text/x-one

Blah

--ABCDE --ABCDE Content-Type: text/x-two

Blah

--ABCDE --ABCDE --ABCDE --ABCDE Content-Type: text/x-two

Blah

--ABCDE--

MIME-Version: 1.0 Content-Type: text/html; boundary="--961284236552522269"

----961284236552522269 Content-Type: text/html; Content-Transfer-Encoding: 7Bit

----961284236552522269--

From: "Allison Dunlap" xxx at example dot com To: yyy@example.com Subject: 64423 Date: Sun, 11 Jul 2004 16:09:27 -0300 MIME-Version: 1.0 Content-Type: multipart/alternative;

Blah blah blah

From: foo at bar dot baz To: baz at bar dot foo Subject: test X-Long-Line: Some really long line contains a lot of text and thus has to be rewrapped because it is some really long line MIME-Version: 1.0 Content-Type: multipart/signed; boundary="borderline"; protocol="application/pgp-signature"; micalg=pgp-sha1

This is an OpenPGP/MIME signed message (RFC 2440 and 3156) --borderline Content-Type: text/plain X-Long-Line: Another really long line contains a lot of text and thus has to be rewrapped because it is another really long line

This is the signed contents.

--borderline Content-Type: application/pgp-signature; name="signature.asc" Content-Description: OpenPGP digital signature Content-Disposition: attachment; filename="signature.asc"

-----BEGIN PGP SIGNATURE----- Version: GnuPG v2.0.6 (GNU/Linux)

iD8DBQFG03voRhp6o4m9dFsRApSZAKCCAN3IkJlVRg6NvAiMHIvvIuMGPQCeLZtj
FGwfnRHFBFO/S4/DKysm0II= =t7+s -----END PGP SIGNATURE-----

--borderline--

Return-Path: sender at example dot net Delivery-Date: Mon, 08 Feb 2010 14:05:16 +0100 Received: from example.org (example.org [64.5.53.58]) by example.net (node=mxbap2) with ESMTP (Nemesis) id UNIQUE for someone@example.com; Mon, 08 Feb 2010 14:05:16 +0100 Date: Mon, 01 Feb 2010 12:21:16 +0100 From: "Sender" sender at example dot net To: someone at example dot com Subject: GroupwiseForwardingTest Mime-Version: 1.0 Content-Type: message/rfc822

Return-path: sender at example dot net Message-ID: 4B66B890.4070408 at teconcept dot de Date: Mon, 01 Feb 2010 12:18:40 +0100 From: "Dr. Sender" sender at example dot net MIME-Version: 1.0 To: "Recipient" recipient at example dot com Subject: GroupwiseForwardingTest Content-Type: text/plain; charset=ISO-8859-15 Content-Transfer-Encoding: 7bit

Testing email forwarding with Groupwise 1.2.2010

-- coding: utf-8 --

IMPORTANT: unlike the other test_tokenize-*.txt files, this file # does NOT have the utf-8 BOM signature '' at the start # of it. Make sure this is not added inadvertently by your editor # if any changes are made to this file!

Arbitrary encoded utf-8 text (stolen from test_doctest2.py).

x = 'ЉЊЈЁЂ' def y(): """ And again in a comment. ЉЊЈЁЂ """ pass

-- coding: utf-8 --

IMPORTANT: this file has the utf-8 BOM signature ''

at the start of it. Make sure this is preserved if any changes

are made!

Arbitrary encoded utf-8 text (stolen from test_doctest2.py).

x = 'ЉЊЈЁЂ' def y(): """ And again in a comment. ЉЊЈЁЂ """ pass

Tests for the 'tokenize' module.

Large bits stolen from test_grammar.py.

Comments

```
"#" #' #" #
# # abc '''# #'''

x = 1 #
```

Balancing continuation

a = (3, 4, 5, 6) y = [3, 4, 5] z = {'a':5, 'b':6} x = (len(repr(y)) + 5*x - a[3] - x + len({ }))

Backslash means line continuation:

```
x = 1
+ 1
```

Backslash does not means continuation in comments :
x = 0

Ordinary integers

0xff != 255 0o377 != 255 2147483647 != 0o17777777777 -2147483647-1 != 0o20000000000 0o37777777777 != -1 0xffffffff != -1; 0o37777777777 != -1; -0o1234567 == 0O001234567; 0b10101 == 0B00010101

Long integers

x = 0 x = 0 x = 0xffffffffffffffff x = 0xffffffffffffffff x = 0o77777777777777777 x = 0B11101010111111111 x = 1234567890123456789012345678901234567890 x = 1234567890123456789012345678901234567890

Floating-point numbers

x = 3.14 x = 314. x = 0.314 # XXX x = 000.314 x = .314 x = 3e14 x = 3E14 x = 3e-14 x = 3e+14 x = 3.e14 x = .3e14 x = 3.1e4

String literals

x = ''; y = ""; x = '''; y = """; x = '"'; y = """; x = "doesn't "shrink" does it" y = 'doesn't "shrink" does it' x = "does "shrink" doesn't it" y = 'does "shrink" doesn't it' x = """ The "quick" brown fox jumps over the 'lazy' dog. """ y = '"quick"foxover'lazy' dog.' y = ''' The "quick" brown fox jumps over the 'lazy' dog. '''; y = "
The "quick"
brown fox
jumps over
the 'lazy' dog.
"; y = '
The "quick"
brown fox
jumps over
the 'lazy' dog.
'; x = r'\' + R'\' x = r''' + '' y = r''' foo bar \ baz''' + R''' foo''' y = r"""foo bar \ baz """ + R'''spam ''' x = b'abc' + B'ABC' y = b"abc" + B"ABC" x = br'abc' + Br'ABC' + bR'ABC' + BR'ABC' y = br"abc" + Br"ABC" + bR"ABC" + BR"ABC" x = rb'abc' + rB'ABC' + Rb'ABC' + RB'ABC' y = rb"abc" + rB"ABC" + Rb"ABC" + RB"ABC" x = br'\' + BR'\' x = rb'\' + RB'\' x = br''' + '' x = rb''' + '' y = br''' foo bar \ baz''' + BR''' foo''' y = Br"""foo bar \ baz """ + bR'''spam ''' y = rB"""foo bar \ baz """ + Rb'''spam '''

Indentation

if 1: x = 2 if 1: x = 2 if 1: while 0: if 0: x = 2 x = 2 if 0: if 2: while 0: if 1: x = 2

Operators

def d22(a, b, c=1, d=2): pass def d01v(a=1, *restt, **restd): pass

(x, y) != ({'a':1}, {'b':2})

comparison

if 1 < 1 > 1 == 1 >= 1 <= 1 != 1 != 1 in 1 not in 1 is 1 is not 1: pass

binary

x = 1 & 1 x = 1 ^ 1 x = 1 | 1

shift

x = 1 << 1 >> 1

additive

$x = 1 - 1 + 1 - 1 + 1$

multiplicative

```
x = 1 / 1 * 1 % 1
```

unary

x = ~1 ^ 1 & 1 | 1 & 1 ^ -1 x = -1*1/1* + *1*1 - ---1*1

selector

import sys, time x = sys.modules['time'].time()

@staticmethod def foo(): pass

@staticmethod def foo(x:1)->1: pass

This file describes some special Python build types enabled via compile-time preprocessor defines.

IMPORTANT: if you want to build a debug-enabled Python, it is recommended that you use `./configure --with-pydebug`, rather than the options listed here.

However, if you wish to define some of these options individually, it is best to define them in the EXTRA_CFLAGS make variable; `make EXTRA_CFLAGS="-DPy_REF_DEBUG"`.

Py_REF_DEBUG

Turn on aggregate reference counting. This arranges that extern _Py_RefTotal hold a count of all references, the sum of ob_refcnt across all objects. Passing `-X showrefcount` on the command line causes the interactive interpreter to print the reference count total as well the number of memory blocks allocated after each statement:

```
>>> 23
23
[8288 refs, 14332 blocks]
>>>
```

Note that if this count increases when you're not storing away new objects, there's probably a leak. Remember, though, that in interactive mode the special name "_" holds a reference to the last result displayed!

Py_REF_DEBUG also checks after every decref to verify that the refcount hasn't gone negative, and causes an immediate fatal error if it has.

Special gimmicks:

sys.gettotalrefcount() Return current total of all refcounts.

Py_TRACE_REFS

Turn on heavy reference debugging. This is major surgery. Every PyObject grows two more pointers, to maintain a doubly-linked list of all live heap-allocated objects. Most built-in type objects are not in this list, as they're statically allocated. Starting in Python 2.3, if COUNT_ALLOCS (see below) is also defined, a static type object T does appear in this list if at least one object of type T has been created.

Note that because the fundamental PyObject layout changes, Python modules compiled with Py_TRACE_REFS are incompatible with modules compiled without it.

Py_TRACE_REFS implies Py_REF_DEBUG.

Special gimmicks:

sys.getobjects(max[, type]) Return list of the (no more than) max most-recently allocated objects, most recently allocated first in the list, least-recently allocated last in the list. max=0 means no limit on list length. If an optional type object is passed, the list is also restricted to objects of that type. The return list itself, and some temp objects created just to call sys.getobjects(), are excluded from the return list. Note that the list returned is just another object, though, so may appear in the return list the next time you call getobjects(); note that every object in the list is kept alive too, simply by virtue of being in the list.

envvar PYTHONDUMPREFS If this envvar exists, Py_Finalize() arranges to print a list of all still-live heap objects. This is printed twice, in different formats, before and after Py_Finalize has cleaned up everything it can clean up. The first output block produces the repr() of each object so is more informative; however, a lot of stuff destined to die is still alive then. The second output block is much harder to work with (repr() can't be invoked anymore -- the interpreter has been torn down too far), but doesn't list any objects that will die. The tool script combinerefs.py can be run over this to combine the info from both output blocks. The second output block, and combinerefs.py, were new in Python 2.3b1.

PYMALLOC_DEBUG

When pymalloc is enabled (WITH_PYMALLOC is defined), calls to the PyObject_ memory routines are handled by Python's own small-object allocator, while calls to the PyMem_ memory routines are directed to the system malloc/ realloc/free. If PYMALLOC_DEBUG is also defined, calls to both PyObject_ and PyMem_ memory routines are directed to a special debugging mode of Python's small-object allocator.

This mode fills dynamically allocated memory blocks with special, recognizable bit patterns, and adds debugging info on each end of dynamically allocated memory blocks. The special bit patterns are:

define CLEANBYTE 0xCB /* clean (newly allocated) memory / *#define DEADBYTE 0xDB* / dead (newly freed) memory / *#define FORBIDDENBYTE 0xFB* / forbidden -- untouchable bytes */

Strings of these bytes are unlikely to be valid addresses, floats, or 7-bit ASCII strings.

Let S = sizeof(size_t). 2*S bytes are added at each end of each block of N bytes requested. The memory layout is like so, where p represents the address returned by a malloc-like or realloc-like function (p[i:j] means the slice of bytes from* (p+i) inclusive up to *(p+j) exclusive; note that the treatment of negative indices differs from a Python slice):

p[-2*S:-S] Number of bytes originally asked for. This is a size_t, big-endian (easier to read in a memory dump). p[-S] API ID. See PEP 445. This is a character, but seems undocumented. p[-S+1:0] Copies of FORBIDDENBYTE. Used to catch under- writes and reads. p[0:N] The requested memory, filled with copies of CLEANBYTE, used to catch reference to uninitialized memory. When a realloc-like function is called requesting a larger memory block, the new excess bytes are also filled with CLEANBYTE. When a free-like function is called, these are overwritten with DEADBYTE, to catch reference to freed memory. When a realloc- like function is called requesting a smaller memory block, the excess old bytes are also filled with DEADBYTE. p[N:N+S] Copies of FORBIDDENBYTE. Used to catch over- writes and reads. p[N+S:N+2*S] A serial number, incremented by 1 on each call to a malloc-like or realloc-like function. Big-endian size_t. If "bad memory" is detected later, the serial number gives an excellent way to set a breakpoint on the next run, to capture the instant at which this block was passed out. The static function bumpserialno() in obmalloc.c is the only place the serial number is incremented, and exists so you can set such a breakpoint easily.

A realloc-like or free-like function first checks that the FORBIDDENBYTEs at each end are intact. If they've been altered, diagnostic output is written to stderr, and the program is aborted via Py_FatalError(). The other main failure mode is provoking a memory error when a program reads up one of the special bit patterns and tries to use it as an address. If you get in a debugger then and look at the object, you're likely to see that it's entirely filled with 0xDB (meaning freed memory is getting used) or 0xCB (meaning uninitialized memory is getting used).

Note that PYMALLOC_DEBUG requires WITH_PYMALLOC.

Special gimmicks:

envvar PYTHONMALLOCSTATS If this envvar exists, a report of pymalloc summary statistics is printed to stderr whenever a new arena is allocated, and also by Py_Finalize().

Changed in 2.5: The number of extra bytes allocated is 4*sizeof(size_t). Before it was 16 on all boxes, reflecting that Python couldn't make use of allocations >= 2**32 bytes even on 64-bit boxes before 2.5.

Py_DEBUG

This is what is generally meant by "a debug build" of Python.

Py_DEBUG implies LLTRACE, Py_REF_DEBUG, Py_TRACE_REFS, and PYMALLOC_DEBUG (if WITH_PYMALLOC is enabled). In addition, C assert()s are enabled (via the C way: by not defining NDEBUG), and some routines do additional sanity checks inside "#ifdef Py_DEBUG" blocks.

COUNT_ALLOCS

Each type object grows three new members:

```
/* Number of times an object of this type was allocated. */
int tp_allocs;

/* Number of times an object of this type was deallocated. */
int tp_frees;

/* Highwater mark:  the maximum value of tp_allocs - tp_frees so
 * far; or, IOW, the largest number of objects of this type alive at
 * the same time.
 */
int tp_maxalloc;
```

Allocation and deallocation code keeps these counts up to date. Py_Finalize() displays a summary of the info returned by sys.getcounts() (see below), along with assorted other special allocation counts (like the number of tuple allocations satisfied by a tuple free-list, the number of 1-character strings allocated, etc).

Before Python 2.2, type objects were immortal, and the COUNT_ALLOCS implementation relies on that. As of Python 2.2, heap-allocated type/ class objects can go away. COUNT_ALLOCS can blow up in 2.2 and 2.2.1 because of this; this was fixed in 2.2.2. Use of COUNT_ALLOCS makes all heap-allocated type objects immortal, except for those for which no object of that type is ever allocated.

Starting with Python 2.3, If Py_TRACE_REFS is also defined, COUNT_ALLOCS arranges to ensure that the type object for each allocated object appears in the doubly-linked list of all objects maintained by Py_TRACE_REFS.

Special gimmicks:

sys.getcounts() Return a list of 4-tuples, one entry for each type object for which at least one object of that type was allocated. Each tuple is of the form:

```
    (tp_name, tp_allocs, tp_frees, tp_maxalloc)

Each distinct type object gets a distinct entry in this list, even if two or
more type objects have the same tp_name (in which case there's no way to
distinguish them by looking at this list).  The list is ordered by time of
first object allocation: the type object for which the first allocation of
an object of that type occurred most recently is at the front of the list.
```

LLTRACE

Compile in support for Low Level TRACE-ing of the main interpreter loop.

When this preprocessor symbol is defined, before PyEval_EvalFrame (eval_frame in 2.3 and 2.2, eval_code2 before that) executes a frame's code it checks the frame's global namespace for a variable "**lltrace**". If such a variable is found, mounds of information about what the interpreter is doing are sprayed to stdout, such as every opcode and opcode argument and values pushed onto and popped off the value stack.

Not useful very often, but very useful when needed.

CALL_PROFILE

Count the number of function calls executed.

When this symbol is defined, the ceval mainloop and helper functions count the number of function calls made. It keeps detailed statistics about what kind of object was called and whether the call hit any of the special fast paths in the code.

WITH_TSC

Super-lowlevel profiling of the interpreter. When enabled, the sys module grows a new function:

settscdump(bool) If true, tell the Python interpreter to dump VM measurements to stderr. If false, turn off dump. The measurements are based on the processor's time-stamp counter.

This build option requires a small amount of platform specific code. Currently this code is present for linux/x86 and any PowerPC platform that uses GCC (i.e. OS X and linux/ppc).

On the PowerPC the rate at which the time base register is incremented is not defined by the architecture specification, so you'll need to find the manual for your specific processor. For the 750CX, 750CXe and 750FX (all sold as the G3) we find:

```
The time base counter is clocked at a frequency that is one-fourth that of
the bus clock.
```

This build is enabled by the --with-tsc flag to configure.

Bignum support (Fast Number Theoretic Transform or FNT):

Bignum arithmetic in libmpdec uses the scheme for fast convolution of integer sequences from:

J. M. Pollard: The fast Fourier transform in a finite field http://www.ams.org/journals/mcom/1971-25-114/S0025-5718-1971-0301966-0/home.html

The transform in a finite field can be used for convolution in the same way as the Fourier Transform. The main advantages of the Number Theoretic Transform are that it is both exact and very memory efficient.

Convolution in pseudo-code: ~˜

~

fnt_convolute(a, b): x = fnt(a) # forward transform of a y = fnt(b) # forward transform of b z = pairwise multiply x[i] and y[i] result = inv_fnt(z) # backward transform of z.

Extending the maximum transform length (Chinese Remainder Theorem):

The maximum transform length is quite limited when using a single prime field. However, it is possible to use multiple primes and recover the result using the Chinese Remainder Theorem.

Multiplication in pseudo-code: ~

~

_mpd_fntmul(u, v): c1 = fnt_convolute(u, v, P1) # convolute modulo prime1 c2 = fnt_convolute(u, v, P2) # convolute modulo prime2 c3 = fnt_convolute(u, v, P3) # convolute modulo prime3 result = crt3(c1, c2, c3) # Chinese Remainder Theorem

Optimized transform functions:

There are three different fnt() functions:

std_fnt: "standard" decimation in frequency transform for array lengths of 2**n. Performs well up to 1024 words.

sixstep: Cache-friendly algorithm for array lengths of 2**n. Outperforms std_fnt for large arrays.

fourstep: Algorithm for array lengths of 3 * 2**n. Also cache friendly in large parts.

List of bignum-only files:

Functions from these files are only used in _mpd_fntmul().

umodarith.h -> fast low level routines for unsigned modular arithmetic numbertheory.c -> routines for setting up the FNT difradix2.c -> decimation in frequency transform, used as the "base case" by the following three files:

```
fnt.c          -> standard transform for smaller arrays
sixstep.c      -> transform large arrays of length 2**n
fourstep.c     -> transform arrays of length 3 * 2**n
```

convolute.c -> do the actual fast convolution, using one of the three transform functions. transpose.c -> transpositions needed for the sixstep algorithm. crt.c -> Chinese Remainder Theorem: use information from three transforms modulo three different primes to get the final result.

```
================================================================================
Calculate (a * b) % p using special primes
================================================================================
```

A description of the algorithm can be found in the apfloat manual by Tommila [1].

Definitions:

In the whole document, "==" stands for "is congruent with".

Result of a * b in terms of high/low words:

1. hi * 2**64 + lo = a * b

Special primes:

2. p = 2**64 - z + 1, where z = 2**n

Single step modular reduction:

3. R(hi, lo) = hi * z - hi + lo

Strategy:

1. Set (hi, lo) to the result of a * b.
2. Set (hi', lo') to the result of R(hi, lo).
3. Repeat step b) until 0 <= hi' * 2**64 + lo' < 2*p.
4. If the result is less than p, return lo'. Otherwise return lo' - p.

The reduction step b) preserves congruence:

```
hi * 2**64 + lo == hi * z - hi + lo   (mod p)

Proof:
~~~~~~

   hi * 2**64 + lo = (2**64 - z + 1) * hi + z * hi - hi + lo

                   = p * hi               + z * hi - hi + lo

                   == z * hi - hi + lo   (mod p)
```

Maximum numbers of step b):

To avoid unnecessary formalism, define:

def R(hi, lo, z): return divmod(hi * z - hi + lo, 2**64)

For simplicity, assume hi=264-1, lo=264-1 after the initial multiplication a * b. This is of course impossible but certainly covers all cases.

Then, for p1:

hi=264-1; lo=264-1; z=232 **p1 = 2**64 - z + 1

hi, lo = R(hi, lo, z) # First reduction hi, lo = R(hi, lo, z) # Second reduction hi * 2**64 + lo < 2 * p1 # True

For p2:

hi=264-1; lo=264-1; z=234 p2 = 264 - z + 1

hi, lo = R(hi, lo, z) # First reduction hi, lo = R(hi, lo, z) # Second reduction hi, lo = R(hi, lo, z) # Third reduction hi * 2**64 + lo < 2 * p2 # True

For p3:

hi=264-1; lo=264-1; z=240 p3 = 264 - z + 1

hi, lo = R(hi, lo, z) # First reduction hi, lo = R(hi, lo, z) # Second reduction hi, lo = R(hi, lo, z) # Third reduction hi * 2**64 + lo < 2 * p3 # True

Step d) preserves congruence and yields a result < p:

Case hi = 0:

```
   Case lo < p: trivial.

   Case lo >= p:

      lo == lo - p   (mod p)             # result is congruent

      p <= lo < 2*p  ->  0 <= lo - p < p # result is in the correct range
```

Case hi = 1:

```
   p < 2**64 /\ 2**64 + lo < 2*p  ->  lo < p  # lo is always less than p

   2**64 + lo == 2**64 + (lo - p)   (mod p)   # result is congruent

              = lo - p   # exactly the same value as the previous RHS
                         # in uint64_t arithmetic.

   p < 2**64 + lo < 2*p  ->  0 < 2**64 + (lo - p) < p  # correct range
```

[1] http://www.apfloat.org/apfloat/2.40/apfloat.pdf

========== Calculate (a * b) % p using the 80-bit x87 FPU ==========

A description of the algorithm can be found in the apfloat manual by Tommila [1].

The proof follows an argument made by Granlund/Montgomery in [2].

Definitions and assumptions:

The 80-bit extended precision format uses 64 bits for the significand:

1. F = 64

The modulus is prime and less than 2**31:

2. 2 <= p < 2**31

The factors are less than p:

3. 0 <= a < p
4. 0 <= b < p

The product a * b is less than 2**62 and is thus exact in 64 bits:

5. n = a * b

The product can be represented in terms of quotient and remainder:

6. n = q * p + r

Using (3), (4) and the fact that p is prime, the remainder is always greater than zero:

7. 0 <= q < p / 1 <= r < p

Strategy:

Precalculate the 80-bit long double inverse of p, with a maximum relative error of 2**(1-F):

8. pinv = (long double)1.0 / p

Calculate an estimate for q = floor(n/p). The multiplication has another maximum relative error of 2**(1-F):

9. qest = n * pinv

If we can show that q < qest < q+1, then trunc(qest) = q. It is then easy to recover the remainder r. The complete algorithm is:

1. Set the control word to 64-bit precision and truncation mode.
2. n = a * b # Calculate exact product.
3. qest = n * pinv # Calculate estimate for the quotient.
4. q = (qest+263)-263 # Truncate qest to the exact quotient.
5. r = n - q * p # Calculate remainder.

Proof for q < qest < q+1:

Using the cumulative error, the error bounds for qest are:

$$9.\quad \frac{n}{p} \le qest \le \frac{n * (1 + 2^{(1-F)})^2}{p}$$

Lemma 1:

$$(10)\quad q < \frac{n}{p * (1 + 2^{(1-F)})^2} = \frac{q * p + r}{p * (1 + 2^{(1-F)})^2}$$

Proof:
~~~~~~

$$(I)\quad q * p * (1 + 2^{(1-F)})^2 < q * p + r$$

$$(II)\quad q * p * 2^{(2-F)} + q * p * 2^{(2-2*F)} < r$$

Using (1) and (7), it is sufficient to show that:

$$(III)\quad q * p * 2^{(-62)} + q * p * 2^{(-126)} < 1 <= r$$

(III) can easily be verified by substituting the largest possible values p = 2**31-1 and q = 2**31-2.

The critical cases occur when r = 1, n = m * p + 1. These cases can be exhaustively verified with a test program.

Lemma 2: --------

$$(11)\quad \frac{n * (1 + 2^{(1-F)})^2}{p} = \frac{(q * p + r) * (1 + 2^{(1-F)})^2}{p} < q + 1$$

Proof:
~~~~~~

$$(I)\quad (q * p + r) + (q * p + r) * 2^{(2-F)} + (q * p + r) * 2^{(2-2*F)} < q * p + p$$

$$(II)\quad (q * p + r) * 2^{(2-F)} + (q * p + r) * 2^{(2-2*F)} < p - r$$

Using (1) and (7), it is sufficient to show that:

$$(III)\quad (q * p + r) * 2^{(-62)} + (q * p + r) * 2^{(-126)} < 1 <= p - r$$

(III) can easily be verified by substituting the largest possible values p = 2**31-1, q = 2**31-2 and r = 2**31-2.

The critical cases occur when r = (p - 1), n = m * p - 1. These cases can be exhaustively verified with a test program.

[1] http://www.apfloat.org/apfloat/2.40/apfloat.pdf [2] http://gmplib.org/~tege/divcnst-pldi94.pdf [Section 7: "Use of floating point"]

```
(* Coq proof for (10) and (11) *)

Require Import ZArith. Require Import QArith. Require Import Qpower. Require Import Qabs. Require Import Psatz.

Open Scope Q_scope.

Ltac qreduce T := rewrite <- (Qred_correct (T)); simpl (Qred (T)).

Theorem Qlt_move_right : forall x y z:Q, x + z < y <-> x < y - z. Proof. intros. split. intros. psatzl Q. intros. psatzl Q. Qed.

Theorem Qlt_mult_by_z : forall x y z:Q, 0 < z -> (x < y <-> x * z < y * z). Proof. intros. split. intros. apply Qmult_lt_compat_r. trivial. trivial. intros. rewrite <- (Qdiv_mult_l x z). rewrite <- (Qdiv_mult_l y z). apply Qmult_lt_compat_r. apply Qlt_shift_inv_l. trivial. psatzl Q. trivial. psatzl Q. psatzl Q. Qed.

Theorem Qle_mult_quad : forall (a b c d:Q), 0 <= a -> a <= c -> 0 <= b -> b <= d -> a * b <= c * d. intros. psatz Q. Qed.

Theorem q_lt_qest: forall (p q r:Q), (0 < p) -> (p <= (2#1)^31 - 1) -> (0 <= q) -> (q <= p - 1) -> (1 <= r) -> (r <= p - 1) -> q < (q * p + r) / (p * (1 + (2#1)^(-63))2). Proof. intros. rewrite Qlt_mult_by_z with (z := (p * (1 + (2#1)^(-63))2)).

unfold Qdiv. rewrite <- Qmult_assoc. rewrite (Qmult_comm (/ (p * (1 + (2 # 1) ^ (-63)) ^ 2)) (p * (1 + (2 # 1) ^ (-63)) ^ 2)). rewrite Qmult_inv_r. rewrite Qmult_1_r.

assert (q * (p * (1 + (2 # 1) ^ (-63)) ^ 2) == q * p + (q * p) * ((2 # 1) ^ (-62) + (2 # 1) ^ (-126))). qreduce ((1 + (2 # 1) ^ (-63)) ^ 2). qreduce ((2 # 1) ^ (-62) + (2 # 1) ^ (-126)). ring_simplify. reflexivity. rewrite H5.

rewrite Qplus_comm. rewrite Qlt_move_right. ring_simplify (q * p + r - q * p). qreduce ((2 # 1) ^ (-62) + (2 # 1) ^ (-126)).

apply Qlt_le_trans with (y := 1). rewrite Qlt_mult_by_z with (z := 85070591730234615865843651857942052864 # 18446744073709551617). ring_simplify.

apply Qle_lt_trans with (y := ((2 # 1) ^ 31 - (2#1)) * ((2 # 1) ^ 31 - 1)). apply Qle_mult_quad. assumption. psatzl Q. psatzl Q. psatzl Q. psatzl Q. psatzl Q. assumption. psatzl Q. psatzl Q. Qed.

Theorem qest_lt_qplus1: forall (p q r:Q), (0 < p) -> (p <= (2#1)^31 - 1) -> (0 <= q) -> (q <= p - 1) -> (1 <= r) -> (r <= p - 1) -> ((q * p + r) * (1 + (2#1)^(-63))2) / p < q + 1. Proof. intros. rewrite Qlt_mult_by_z with (z := p).

unfold Qdiv. rewrite <- Qmult_assoc. rewrite (Qmult_comm (/ p) p). rewrite Qmult_inv_r. rewrite Qmult_1_r.

assert ((q * p + r) * (1 + (2 # 1) ^ (-63)) ^ 2 == q * p + r + (q * p + r) * ((2 # 1) ^ (-62) + (2 # 1) ^ (-126))). qreduce ((1 + (2 # 1) ^ (-63)) ^ 2). qreduce ((2 # 1) ^ (-62) + (2 # 1) ^ (-126)). ring_simplify. reflexivity. rewrite H5.

rewrite <- Qplus_assoc. rewrite <- Qplus_comm. rewrite Qlt_move_right. ring_simplify ((q + 1) * p - q * p).

rewrite <- Qplus_comm. rewrite Qlt_move_right.

apply Qlt_le_trans with (y := 1). qreduce ((2 # 1) ^ (-62) + (2 # 1) ^ (-126)).

rewrite Qlt_mult_by_z with (z := 85070591730234615865843651857942052864 # 18446744073709551617). ring_simplify.

ring_simplify in H0. apply Qle_lt_trans with (y := (2147483646 # 1) * (2147483647 # 1) + (2147483646 # 1)).

apply Qplus_le_compat. apply Qle_mult_quad. assumption. psatzl Q. auto with qarith. assumption. psatzl Q. auto with qarith. auto with qarith. psatzl Q. psatzl Q. assumption. Qed.
```

This document contains links to the literature used in the process of creating the library. The list is probably not complete.

Mike Cowlishaw: General Decimal Arithmetic Specification http://speleotrove.com/decimal/decarith.html

Jean-Michel Muller: On the definition of ulp (x) lara.inist.fr/bitstream/2332/518/1/LIP-RR2005-09.pdf

T. E. Hull, A. Abrham: Properly rounded variable precision square root http://portal.acm.org/citation.cfm?id=214413

T. E. Hull, A. Abrham: Variable precision exponential function http://portal.acm.org/citation.cfm?id=6498

Roman E. Maeder: Storage allocation for the Karatsuba integer multiplication algorithm. http://www.springerlink.com/content/w15058mj6v59t565/

J. M. Pollard: The fast Fourier transform in a finite field http://www.ams.org/journals/mcom/1971-25-114/S0025-5718-1971-0301966-0/home.html

David H. Bailey: FFTs in External or Hierarchical Memory http://crd.lbl.gov/~dhbailey/dhbpapers/

W. Morven Gentleman: Matrix Multiplication and Fast Fourier Transforms http://www.alcatel-lucent.com/bstj/vol47-1968/articles/bstj47-6-1099.pdf

Mikko Tommila: Apfloat documentation http://www.apfloat.org/apfloat/2.41/apfloat.pdf

Joerg Arndt: "Matters Computational" http://www.jjj.de/fxt/

Karl Hasselstrom: Fast Division of Large Integers www.treskal.com/kalle/exjobb/original-report.pdf

The Six Step Transform:

In libmpdec, the six-step transform is the Matrix Fourier Transform (See matrix-transform.txt) in disguise. It is called six-step transform after a variant that appears in [1]. The algorithm requires that the input array can be viewed as an R*C matrix.

Algorithm six-step (forward transform):

1a) Transpose the matrix.

1b) Apply a length R FNT to each row.

1c) Transpose the matrix.

2. Multiply each matrix element (addressed by j*C+m) by r**(j*m).
3. Apply a length C FNT to each row.
4. Transpose the matrix.

Note that steps 1a) - 1c) are exactly equivalent to step 1) of the Matrix Fourier Transform. For large R, it is faster to transpose twice and do a transform on the rows than to perform a column transpose directly.

Algorithm six-step (inverse transform):

0. View the matrix as a C*R matrix.
1. Transpose the matrix, producing an R*C matrix.
2. Apply a length C FNT to each row.
3. Multiply each matrix element (addressed by i*C+n) by r**(i*n).

4a) Transpose the matrix.

4b) Apply a length R FNT to each row.

4c) Transpose the matrix.

Again, steps 4a) - 4c) are equivalent to step 4) of the Matrix Fourier Transform.

--

[1] David H. Bailey: FFTs in External or Hierarchical Memory http://crd.lbl.gov/~dhbailey/dhbpapers/

libmpdec

libmpdec is a fast C/C++ library for correctly-rounded arbitrary precision decimal floating point arithmetic. It is a complete implementation of Mike Cowlishaw/IBM's General Decimal Arithmetic Specification.

Files required for the Python _decimal module ==========

Core files for small and medium precision arithmetic ----------

```
basearith.{c,h}  ->  Core arithmetic in base 10**9 or 10**19.
bits.h           ->  Portable detection of least/most significant one-bit.
constants.{c,h}  ->  Constants that are used in multiple files.
context.c        ->  Context functions.
io.{c,h}         ->  Conversions between mpd_t and ASCII strings,
                     mpd_t formatting (allows UTF-8 fill character).
memory.{c,h}     ->  Allocation handlers with overflow detection
                     and functions for switching between static
                     and dynamic mpd_t.
mpdecimal.{c,h}  ->  All (quiet) functions of the specification.
typearith.h      ->  Fast primitives for double word multiplication,
                     division etc.

Visual Studio only:
~~~~~~~~~~~~~~~~~~~
  vccompat.h    ->  snprintf <==> sprintf_s and similar things.
  vcstdint.h    ->  stdint.h (included in VS 2010 but not in VS 2008).
  vcdiv64.asm   ->  Double word division used in typearith.h. VS 2008 does
                    not allow inline asm for x64. Also, it does not provide
                    an intrinsic for double word division.
```

Files for bignum arithmetic: ----------

```
The following files implement the Fast Number Theoretic Transform
used for multiplying coefficients with more than 1024 words (see
mpdecimal.c: _mpd_fntmul()).

  umodarith.h        ->  Fast low level routines for unsigned modular arithmetic.
  numbertheory.{c,h} ->  Routines for setting up the Number Theoretic Transform.
  difradix2.{c,h}    ->  Decimation in frequency transform, used as the
                         "base case" by the following three files:

    fnt.{c,h}        ->  Transform arrays up to 4096 words.
    sixstep.{c,h}    ->  Transform larger arrays of length 2**n.
    fourstep.{c,h}   ->  Transform larger arrays of length 3 * 2**n.

  convolute.{c,h}    ->  Fast convolution using one of the three transform
                         functions.
  transpose.{c,h}    ->  Transpositions needed for the sixstep algorithm.
  crt.{c,h}          ->  Chinese Remainder Theorem: use information from three
                         transforms modulo three different primes to get the
                         final result.
```

Pointers to literature, proofs and more

literature/ ----------

```
REFERENCES.txt  ->  List of relevant papers.
bignum.txt      ->  Explanation of the Fast Number Theoretic Transform (FNT).
fnt.py          ->  Verify constants used in the FNT; Python demo for the
                    O(N**2) discrete transform.

matrix-transform.txt -> Proof for the Matrix Fourier Transform used in
                        fourstep.c.
six-step.txt         -> Show that the algorithm used in sixstep.c is
                        a variant of the Matrix Fourier Transform.
mulmod-64.txt        -> Proof for the mulmod64 algorithm from
                        umodarith.h.
mulmod-ppro.txt      -> Proof for the x87 FPU modular multiplication
                        from umodarith.h.
umodarith.lisp       -> ACL2 proofs for many functions from umodarith.h.
```

Library Author

Stefan Krah skrah at bytereef dot org

About

_decimal.c is a wrapper for the libmpdec library. libmpdec is a fast C library for correctly-rounded arbitrary precision decimal floating point arithmetic. It is a complete implementation of Mike Cowlishaw/IBM's General Decimal Arithmetic Specification.

Build process for the module

As usual, the build process for _decimal.so is driven by setup.py in the top level directory. setup.py autodetects the following build configurations:

1. x64 - 64-bit Python, x86_64 processor (AMD, Intel)
2. uint128 - 64-bit Python, compiler provides __uint128_t (gcc)
3. ansi64 - 64-bit Python, ANSI C
4. ppro - 32-bit Python, x86 CPU, PentiumPro or later
5. ansi32 - 32-bit Python, ANSI C
6. ansi-legacy - 32-bit Python, compiler without uint64_t
7. universal - Mac OS only (multi-arch)

It is possible to override autodetection by exporting:

PYTHON_DECIMAL_WITH_MACHINE=value, where value is one of the above options.

NOTE

decimal.so is not built from a static libmpdec.a since doing so led to failures on AIX (user report) and Windows (mixing static and dynamic CRTs causes locale problems and more).

This directory contains extended tests and a benchmark against decimal.py:

bench.py -> Benchmark for small and large precisions. Usage: ../../../python bench.py

formathelper.py -> randdec.py -> Generate test cases for deccheck.py. randfloat.py ->

deccheck.py -> Run extended tests. Usage: ../../../python deccheck.py [--short|--medium|--long|--all]

All about co_lnotab, the line number table.

Code objects store a field named co_lnotab. This is an array of unsigned bytes disguised as a Python string. It is used to map bytecode offsets to source code line #s for tracebacks and to identify line number boundaries for line tracing.

The array is conceptually a compressed list of (bytecode offset increment, line number increment) pairs. The details are important and delicate, best illustrated by example:

```
byte code offset    source code line number
    0           1
    6           2
   50           7
  350                 307
  361                 308
```

Instead of storing these numbers literally, we compress the list by storing only the increments from one row to the next. Conceptually, the stored list might look like:

```
0, 1,  6, 1,  44, 5,  300, 300,  11, 1
```

The above doesn't really work, but it's a start. Note that an unsigned byte can't hold negative values, or values larger than 255, and the above example contains two such values. So we make two tweaks:

1. there's a deep assumption that byte code offsets and their corresponding line #s both increase monotonically, and

2. if at least one column jumps by more than 255 from one row to the next, more than one pair is written to the table. In case #b, there's no way to know from looking at the table later how many were written. That's the delicate part. A user of co_lnotab desiring to find the source line number corresponding to a bytecode address A should do something like this

   lineno = addr = 0 for addr_incr, line_incr in co_lnotab: addr += addr_incr if addr > A: return lineno lineno += line_incr

(In C, this is implemented by PyCode_Addr2Line().) In order for this to work, when the addr field increments by more than 255, the line # increment in each pair generated must be 0 until the remaining addr increment is < 256. So, in the example above, assemble_lnotab in compile.c should not (as was actually done until 2.2) expand 300, 300 to 255, 255, 45, 45, but to 255, 0, 45, 255, 0, 45.

The above is sufficient to reconstruct line numbers for tracebacks, but not for line tracing. Tracing is handled by PyCode_CheckLineNumber() in codeobject.c and maybe_call_line_trace() in ceval.c.

*** Tracing ***

To a first approximation, we want to call the tracing function when the line number of the current instruction changes. Re-computing the current line for every instruction is a little slow, though, so each time we compute the line number we save the bytecode indices where it's valid:

```
 *instr_lb <= frame->f_lasti < *instr_ub
```

is true so long as execution does not change lines. That is, *instr_lb holds the first bytecode index of the current line, and* instr_ub holds the first bytecode index of the next line. As long as the above expression is true, maybe_call_line_trace() does not need to call PyCode_CheckLineNumber(). Note that the same line may appear multiple times in the lnotab, either because the bytecode jumped more than 255 indices between line number changes or because the compiler inserted the same line twice. Even in that case, *instr_ub holds the first index of the next line.

However, we don't *always* want to call the line trace function when the above test fails.

Consider this code:

1: def f(a): 2: while a: 3: print 1, 4: break 5: else: 6: print 2,

which compiles to this:

2 0 SETUP_LOOP 19 (to 22) >> 3 LOAD_FAST 0 (a) 6 POP_JUMP_IF_FALSE 17

3 9 LOAD_CONST 1 (1) 12 PRINT_ITEM

4 13 BREAK_LOOP
14 JUMP_ABSOLUTE 3 >> 17 POP_BLOCK

6 18 LOAD_CONST 2 (2) 21 PRINT_ITEM
>> 22 LOAD_CONST 0 (None) 25 RETURN_VALUE

If 'a' is false, execution will jump to the POP_BLOCK instruction at offset 17 and the co_lnotab will claim that execution has moved to line 4, which is wrong. In this case, we could instead associate the POP_BLOCK with line 5, but that would break jumps around loops without else clauses.

We fix this by only calling the line trace function for a forward jump if the co_lnotab indicates we have jumped to the *start* of a line, i.e. if the current instruction offset matches the offset given for the start of a line by the co_lnotab. For backward jumps, however, we always call the line trace function, which lets a debugger stop on every evaluation of a loop guard (which usually won't be the first opcode in a line).

Why do we set f_lineno when tracing, and only just before calling the trace function? Well, consider the code above when 'a' is true. If stepping through this with 'n' in pdb, you would stop at line 1 with a "call" type event, then line events on lines 2, 3, and 4, then a "return" type event -- but because the code for the return actually falls in the range of the "line 6" opcodes, you would be shown line 6 during this event. This is a change from the behaviour in 2.2 and before, and I've found it confusing in practice. By setting and using f_lineno when tracing, one can report a line number different from that suggested by f_lasti on this one occasion where it's desirable.

XXX Write description XXX Dont't forget to mention upx

XXX Add pointer to this file into PC/README.txt

In Win32, DLL's are "pre-linked" using a specified base address. When the DLL is loaded, an attempt is made to place it at that address. If that address is already in use, a new base address is selected, and the DLL subject to fixups. Apparently, these fixups are very slow, and significant performance gains can be made by selecting a good base address.

This document is to allocate base addresses to core Python and Python .PYD files, to give a better change of optimal performance. This base address is passed to the linker using the /BASE command line switch.

Python.exe/Pythonw.exe - 1d000000 - 1e000000 (-1) Python.dll - 1e000000 - 1e100000 (-1)

Standard Extension Modules 1e100000 - 1e200000 "" - _symtable 1e100000 - 1e110000 pyd removed in 2.4 - bsddb 1e180000 - 1e188000 pyd removed in 3.0 - _tkinter 1e190000 - 1e1A0000 - parser 1e1A0000 - 1e1B0000 pyd removed in 2.4 - zlib 1e1B0000 - 1e1C0000 - winreg 1e1C0000 - 1e1D0000 pyd removed in 2.4 - _socket 1e1D0000 - 1e1E0000 - _sre 1e1E0000 - 1e1F0000 pyd removed in 2.4 - mmap 1e1F0000 - 1e1FFFFF pyd removed in 2.4

More standard extensions 1D100000 - 1e000000 - pyexpat 1D100000 - 1D110000 - select 1D110000 - 1D120000 - unicodedata 1D120000 - 1D160000 - winsound 1D160000 - 1D170000 - bZ2 1D170000 - 1D180000 - datetime 1D180000 - 1D190000 pyd removed in 2.4 - _csv 1D190000 - 1D1A0000 pyd removed in 2.4 - _ctypes 1D1A0000 - 1D1B0000

Other extension modules - win32api 1e200000 - 1e220000 - win32ras 1e220000 - 1e230000 - win32lz 1e230000 - 1e240000 - timer 1e240000 - 1e250000 - mmapfile 1e250000 - 1e260000 - win32pipe 1e260000 - 1e270000 - avl 1e270000 - 1e270000 - dbhash 1e280000 - 1e290000 - win32net 1e290000 - 1e2A0000 - win32security 1e2A0000 - 1e2B0000 - win32print 1e2B0000 - 1e2c0000 - 1e2d0000 - 1e2e0000 - win32gui 1e2e0000 - 1e2f0000 - _imaging 1e2f0000 - 1e300000 - multiarray 1e300000 - 1e310000 - win32help 1e310000 - 1e320000 - win32clipboard 1e320000 - 1e330000 - win2kras 1e330000 - 1e340000 - pythoncom 1e340000 - 1e400000 - win32ui 1e400000 - 1e500000 - win32uiole 1e500000 - 1e600000 - pywintypes 1e600000 - 1e700000 - win32process 1e700000 - 1e800000 - odbc 1e710000 - 1e720000 - dbi 1e720000 - 1e730000 - win32file 1e730000 - 1e740000 - win32wnet 1e740000 - 1e750000 - win32com.shell 1e750000 - 1e760000 - win32com.internet 1e760000 - 1e770000 - win32com.exchange 1e770000 - 1e780000 - win32com.exchdapi 1e780000 - 1e790000 - win32com.axscript 1e790000 - 1e7a0000 - win32com.axdebug 1e7b0000 - 1e7c0000 - win32com.adsi 1e7f0000 - 1e800000 - win32event 1e810000 - 1e820000 - win32evtlog 1e820000 - 1e830000 - win32com.axcontrol 1e830000 - 1e840000

Quick Start Guide

1. Install Microsoft Visual Studio 2015, any edition.
2. Install Subversion, and make sure 'svn.exe' is on your PATH.
3. Run "build.bat -e" to build Python in 32-bit Release configuration.
4. (Optional, but recommended) Run the test suite with "rt.bat -q".

Building Python using Microsoft Visual C++

This directory is used to build CPython for Microsoft Windows NT version 6.0 or higher (Windows Vista, Windows Server 2008, or later) on 32 and 64 bit platforms. Using this directory requires an installation of Microsoft Visual C++ 2015 (MSVC 14.0) of any edition. The specific requirements are as follows:

Visual Studio Express 2015 for Desktop Visual Studio Professional 2015 Either edition is sufficient for building all configurations except for Profile Guided Optimization. The Python build solution pcbuild.sln makes use of Solution Folders, which this edition does not support. Any time pcbuild.sln is opened or reloaded by Visual Studio, a warning about Solution Folders will be displayed, which can be safely dismissed with no impact on your ability to build Python. Required for building 64-bit Debug and Release configuration builds Visual Studio Premium 2015 Required for building Release configuration builds that make use of Profile Guided Optimization (PGO), on either platform.

All you need to do to build is open the solution "pcbuild.sln" in Visual Studio, select the desired combination of configuration and platform, then build with "Build Solution". You can also build from the command line using the "build.bat" script in this directory; see below for details. The solution is configured to build the projects in the correct order.

The solution currently supports two platforms. The Win32 platform is used to build standard x86-compatible 32-bit binaries, output into the win32 sub-directory. The x64 platform is used for building 64-bit AMD64 (aka x86_64 or EM64T) binaries, output into the amd64 sub-directory. The Itanium (IA-64) platform is no longer supported. See the "Building for AMD64" section below for more information about 64-bit builds.

Four configuration options are supported by the solution: Debug Used to build Python with extra debugging capabilities, equivalent to using ./configure --with-pydebug on UNIX. All binaries built using this configuration have "_d" added to their name: python35_d.dll, python_d.exe, parser_d.pyd, and so on. Both the build and rt (run test) batch files in this directory accept a -d option for debug builds. If you are building Python to help with development of CPython, you will most likely use this configuration. PGInstrument, PGUpdate Used to build Python in Release configuration using PGO, which requires Premium Edition of Visual Studio. See the "Profile Guided Optimization" section below for more information. Build output from each of these configurations lands in its own sub-directory of this directory. The official Python releases may be built using these configurations. Release Used to build Python as it is meant to be used in production settings, though without PGO.

Building Python using the build.bat script

In this directory you can find build.bat, a script designed to make building Python on Windows simpler. This script will use the env.bat script to detect one of Visual Studio 2015, 2013, 2012, or 2010, any of which may be used to build Python, though only Visual Studio 2015 is officially supported.

By default, build.bat will build Python in Release configuration for the 32-bit Win32 platform. It accepts several arguments to change this behavior:

-c Set the configuration (see above) -d Shortcut for "-c Debug" -p Set the platform to build for ("Win32" or "x64") -r Rebuild instead of just building -t Set the target (Build, Rebuild, Clean or CleanAll) -e Use get_externals.bat to fetch external sources -M Don't build in parallel -v Increased output messages

Up to 9 MSBuild switches can also be passed, though they must be passed after specifying any of the above switches. For example, use:

build.bat -e -d /fl

to do a debug build with externals fetched as needed and write detailed build logs to a file. If the MSBuild switch requires an equal sign ("="), the entire switch must be quoted:

build.bat -e -d "/p:ExternalsDir=P:-externals"

There may also be other situations where quotes are necessary.

C Runtime

Visual Studio 2015 uses version 14 of the C runtime (MSVCRT14). The executables no longer use the "Side by Side" assemblies used in previous versions of the compiler. This simplifies distribution of applications.

The run time libraries are available under the VC/Redist folder of your Visual Studio distribution. For more info, see the Readme in the VC/Redist folder.

Sub-Projects

The CPython project is split up into several smaller sub-projects which are managed by the pcbuild.sln solution file. Each sub-project is represented by a .vcxproj and a .vcxproj.filters file starting with the name of the sub-project. These sub-projects fall into a few general categories:

The following sub-projects represent the bare minimum required to build a functioning CPython interpreter. If nothing else builds but these, you'll have a very limited but usable python.exe: pythoncore .dll and .lib python .exe make_buildinfo, make_versioninfo helpers to provide necessary information to the build process

These sub-projects provide extra executables that are useful for running CPython in different ways: pythonw pythonw.exe, a variant of python.exe that doesn't open a Command Prompt window pylauncher py.exe, the Python Launcher for Windows, see http://docs.python.org/3/using/windows.html#launcher pywlauncher pyw.exe, a variant of py.exe that doesn't open a Command Prompt window _testembed _testembed.exe, a small program that embeds Python for testing purposes, used by test_capi.py

These are miscellaneous sub-projects that don't really fit the other categories: _freeze_importlib _freeze_importlib.exe, used to regenerate Python.h after changes have been made to Lib_bootstrap.py bdist_wininst ..-14.0[-amd64].exe, the base executable used by the distutils bdist_wininst command python3dll python3.dll, the PEP 384 Stable ABI dll xxlimited builds an example module that makes use of the PEP 384 Stable ABI, see Modules.c

The following sub-projects are for individual modules of the standard library which are implemented in C; each one builds a DLL (renamed to .pyd) of the same name as the project: _ctypes _ctypes_test _decimal _elementtree _hashlib _msi _multiprocessing _overlapped _socket _testcapi _testbuffer _testimportmultiple pyexpat select unicodedata winsound

The following Python-controlled sub-projects wrap external projects. Note that these external libraries are not necessary for a working interpreter, but they do implement several major features. See the "Getting External Sources" section below for additional information about getting the source for building these libraries. The sub-projects are: _bz2 Python wrapper for version 1.0.6 of the libbzip2 compression library Homepage: http://www.bzip.org/ _lzma Python wrapper for the liblzma compression library, using pre-built binaries of XZ Utils version 5.0.5 Homepage: http://tukaani.org/xz/ _ssl Python wrapper for version 1.0.1j of the OpenSSL secure sockets library, which is built by ssl.vcxproj Homepage: http://www.openssl.org/

```
Building OpenSSL requires nasm.exe (the Netwide Assembler), version
2.10 or newer from
    http://www.nasm.us/
to be somewhere on your PATH.  More recent versions of OpenSSL may
need a later version of NASM. If OpenSSL's self tests don't pass,
you should first try to update NASM and do a full rebuild of
OpenSSL.  get_externals.py also downloads a snapshot of NASM, and the
libeay and ssleay sub-projects use that version of nasm.exe.

The libeay/ssleay sub-projects expect your OpenSSL sources to have
already been configured and be ready to build.  If you get your sources
from svn.python.org as suggested in the "Getting External Sources"
section below, the OpenSSL source will already be ready to go.  If
you want to build a different version, you will need to run

   PCbuild\prepare_ssl.py path\to\openssl-source-dir

That script will prepare your OpenSSL sources in the same way that
those available on svn.python.org have been prepared.  Note that
Perl must be installed and available on your PATH to configure
OpenSSL.  ActivePerl is recommended and is available from
    http://www.activestate.com/activeperl/

The libeay and ssleay sub-projects will build the modules of OpenSSL
```

required by _ssl and _hashlib and may need to be manually updated when
upgrading to a newer version of OpenSSL or when adding new
functionality to _ssl or _hashlib. They will not clean up their output
with the normal Clean target; CleanAll should be used instead.

_sqlite3 Wraps SQLite 3.8.3.1, which is itself built by sqlite3.vcxproj Homepage: http://www.sqlite.org/
_tkinter Wraps version 8.6.1 of the Tk windowing system. Homepage: http://www.tcl.tk/

Tkinter's dependencies are built by the tcl.vcxproj and tk.vcxproj
projects. The tix.vcxproj project also builds the Tix extended
widget set for use with Tkinter.

Those three projects install their respective components in a
directory alongside the source directories called "tcltk" on
Win32 and "tcltk64" on x64. They also copy the Tcl and Tk DLLs
into the current output directory, which should ensure that Tkinter
is able to load Tcl/Tk without having to change your PATH.

The tcl, tk, and tix sub-projects do not clean their builds with
the normal Clean target; if you need to rebuild, you should use the
CleanAll target or manually delete their builds.

Getting External Sources

The last category of sub-projects listed above wrap external projects Python doesn't control, and as such a little more work is required in order to download the relevant source files for each project before they can be built. However, a simple script is provided to make this as painless as possible, called "get_externals.bat" and located in this directory. This script extracts all the external sub-projects from http://svn.python.org/projects/external via Subversion (so you'll need svn.exe on your PATH) and places them in ..(relative to this directory).

It is also possible to download sources from each project's homepage, though you may have to change folder names or pass the names to MSBuild as the values of certain properties in order for the build solution to find them. This is an advanced topic and not necessarily fully supported.

Building for AMD64

The build process for AMD64 / x64 is very similar to standard builds, you just have to set x64 as platform. In addition, the HOST_PYTHON environment variable must point to a Python interpreter (at least 2.4), to support cross-compilation from Win32.

Profile Guided Optimization

The solution has two configurations for PGO. The PGInstrument configuration must be built first. The PGInstrument binaries are linked against a profiling library and contain extra debug information. The PGUpdate configuration takes the profiling data and generates optimized binaries.

The build_pgo.bat script automates the creation of optimized binaries. It creates the PGI files, runs the unit test suite or PyBench with the PGI python, and finally creates the optimized files.

See http://msdn.microsoft.com/en-us/library/e7k32f4k(VS.100).aspx for more on this topic.

Static library

The solution has no configuration for static libraries. However it is easy to build a static library instead of a DLL. You simply have to set the "Configuration Type" to "Static Library (.lib)" and alter the preprocessor macro "Py_ENABLE_SHARED" to "Py_NO_ENABLE_SHARED". You may also have to change the "Runtime Library" from "Multi-threaded DLL (/MD)" to "Multi-threaded (/MT)".

Visual Studio properties

The PCbuild solution makes use of Visual Studio property files (*.props) to simplify each project. The properties can be viewed in the Property Manager (View -> Other Windows -> Property Manager) but should be carefully modified by hand.

The property files used are: * python (versions, directories and build names) * pyproject (base settings for all projects) * openssl (used by libeay and ssleay projects) * tcltk (used by _tkinter, tcl, tk and tix projects)

The pyproject property file defines all of the build settings for each project, with some projects overriding certain specific values. The GUI doesn't always reflect the correct settings and may confuse the user with false information, especially for settings that automatically adapt for diffirent configurations.

Your Own Extension DLLs

If you want to create your own extension module DLL (.pyd), there's an example with easy-to-follow instructions in ..; read the file readme.txt there first.

This license applies to the bootstrapper application that is embedded within the installer. It has no impact on the licensing for the rest of the installer or Python itself, as no code covered by this license exists in any other part of the product.

Microsoft Reciprocal License (MS-RL)

This license governs use of the accompanying software. If you use the software, you accept this license. If you do not accept the license, do not use the software.

1. Definitions The terms "reproduce," "reproduction," "derivative works," and "distribution" have the same meaning here as under U.S. copyright law. A "contribution" is the original software, or any additions or changes to the software. A "contributor" is any person that distributes its contribution under this license. "Licensed patents" are a contributor's patent claims that read directly on its contribution.

2. Grant of Rights

1. Copyright Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free copyright license to reproduce its contribution, prepare derivative works of its contribution, and distribute its contribution or any derivative works that you create.
2. Patent Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free license under its licensed patents to make, have made, use, sell, offer for sale, import, and/or otherwise dispose of its contribution in the software or derivative works of the contribution in the software.

3. Conditions and Limitations

1. Reciprocal Grants- For any file you distribute that contains code from the software (in source code or binary format), you must provide recipients the source code to that file along with a copy of this license, which license will govern that file. You may license other files that are entirely your own work and do not contain code from the software under any terms you choose.
2. No Trademark License- This license does not grant you rights to use any contributors' name, logo, or trademarks.
3. If you bring a patent claim against any contributor over patents that you claim are infringed by the software, your patent license from such contributor to the software ends automatically.
4. If you distribute any portion of the software, you must retain all copyright, patent, trademark, and attribution notices that are present in the software.
5. If you distribute any portion of the software in source code form, you may do so only under this license by including a complete copy of this license with your distribution. If you distribute any portion of the software in compiled or object code form, you may only do so under a license that complies with this license.
6. The software is licensed "as-is." You bear the risk of using it. The contributors give no express warranties, guarantees or conditions. You may have additional consumer rights under your local laws which this license cannot change. To the extent permitted under your local laws, the contributors exclude the implied warranties of merchantability, fitness for a particular purpose and non-infringement.

Additional Conditions for this Windows binary build

This program is linked with and uses Microsoft Distributable Code, copyrighted by Microsoft Corporation. The Microsoft Distributable Code includes the following files:

appcrt140.dll desktopcrt140.dll vcruntime140.dll msvcp140.dll concrt140.dll vccorlib140.dll

If you further distribute programs that include the Microsoft Distributable Code, you must comply with the restrictions on distribution specified by Microsoft. In particular, you must require distributors and external end users to agree to terms that protect the Microsoft Distributable Code at least as much as Microsoft's own requirements for the Distributable Code. See Microsoft's documentation (included in its developer tools and on its website at microsoft.com) for specific details.

Redistribution of the Windows binary build of the Python interpreter complies with this agreement, provided that you do not:

- alter any copyright, trademark or patent notice in Microsoft's Distributable Code;
- use Microsoft's trademarks in your programs' names or in a way that suggests your programs come from or are endorsed by Microsoft;
- distribute Microsoft's Distributable Code to run on a platform other than Microsoft operating systems, run-time technologies or application platforms; or
- include Microsoft Distributable Code in malicious, deceptive or unlawful programs.

These restrictions apply only to the Microsoft Distributable Code as defined above, not to Python itself or any programs running on the Python interpreter. The redistribution of the Python interpreter and libraries is governed by the Python Software License included with this file, or by other licenses as marked.

Additional Conditions for this Windows binary build

This program is linked with and uses Microsoft Distributable Code, copyrighted by Microsoft Corporation. The Microsoft Distributable Code includes the following files:

msvcr90.dll msvcp90.dll msvcm90.dll

If you further distribute programs that include the Microsoft Distributable Code, you must comply with the restrictions on distribution specified by Microsoft. In particular, you must require distributors and external end users to agree to terms that protect the Microsoft Distributable Code at least as much as Microsoft's own requirements for the Distributable Code. See Microsoft's documentation (included in its developer tools and on its website at microsoft.com) for specific details.

Redistribution of the Windows binary build of the Python interpreter complies with this agreement, provided that you do not:

- alter any copyright, trademark or patent notice in Microsoft's Distributable Code;
- use Microsoft's trademarks in your programs' names or in a way that suggests your programs come from or are endorsed by Microsoft;
- distribute Microsoft's Distributable Code to run on a platform other than Microsoft operating systems, run-time technologies or application platforms; or
- include Microsoft Distributable Code in malicious, deceptive or unlawful programs.

These restrictions apply only to the Microsoft Distributable Code as defined above, not to Python itself or any programs running on the Python interpreter. The redistribution of the Python interpreter and libraries is governed by the Python Software License included with this file, or by other licenses as marked.

Quick Build Info

For testing, the installer should be built with the Tools/msi/build.bat script:

```
build.bat [-x86] [-x64] [--doc]
```

For an official release, the installer should be built with the Tools/msi/buildrelease.bat script and environment variables:

```
set PYTHON=<path to Python 2.7>
set SPHINXBUILD=<path to sphinx-build.exe>
set PATH=<path to Mercurial (hg.exe)>;
         <path to HTML Help Compiler (hhc.exe)>;%PATH%

buildrelease.bat [-x86] [-x64] [-D] [-B]
    [-o <output directory>] [-c <certificate name>]
```

See the Building the Installer section for more information.

Overview

Python is distributed on Windows as an installer that will configure the user's system. This allows users to have a functioning copy of Python without having to build it themselves.

The main tasks of the installer are:

- copy required files into the expected layout
- configure system settings so the installation can be located by other programs
- add entry points for modifying, repairing and uninstalling Python
- make it easy to launch Python, its documentation, and IDLE

Each of these is discussed in a later section of this document.

Structure of the Installer

The installer is structured as a 'layout', which consists of a number of CAB and MSI files and a single EXE.

The EXE is the main entry point into the installer. It contains the UI and command-line logic, as well as the ability to locate and optionally download other parts of the layout.

Each MSI contains the logic required to install a component or feature of Python. These MSIs should not be launched directly by users. MSIs can be embedded into the EXE or automatically downloaded as needed.

Each CAB contains the files making up a Python installation. CABs are embedded into their associated MSI and are never seen by users.

MSIs are only required when the related feature or component is being installed. When components are not selected for installation, the associated MSI is not downloaded. This allows the installer to offer options to install debugging symbols and binaries without increasing the initial download size by separating them into their own MSIs.

Building the Installer

For testing, the installer should be built with the Tools/msi/build.bat script:

```
build.bat [-x86] [-x64] [--doc]
```

This script will build the required configurations of Python and generate an installer layout in PCBuild/(win32|amd64)/en-us.

Specify -x86 and/or -x64 to build for each platform. If neither is specified, both platforms will be built. Currently, both the debug and release versions of Python are required for the installer.

Specify --doc to build the documentation (.chm) file. If the file is not available, it will simply be excluded from the installer. Ensure %PYTHON% and %SPHINXBUILD% are set when passing this option. You may also set %HTMLHELP% to the Html Help Compiler (hhc.exe), or put HHC on your PATH or in externals/.

If WiX is not found on your system, it will be automatically downloaded and extracted to the externals/ directory.

For an official release, the installer should be built with the Tools/msi/buildrelease.bat script:

```
set PYTHON=<path to Python 2.7>
set SPHINXBUILD=<path to sphinx-build.exe>
set PATH=<path to Mercurial (hg.exe)>;
         <path to HTML Help Compiler (hhc.exe)>;%PATH%

buildrelease.bat [-x86] [-x64] [-D] [-B]
    [-o <output directory>] [-c <certificate name>]
```

Specify -x86 and/or -x64 to build for each platform. If neither is specified, both platforms will be built. Currently, both the debug and release versions of Python are required for the installer.

Specify -D to skip rebuilding the documentation. The documentation is required for a release and the build will fail if it is not available.

Specify -B to skip rebuilding Python. This is useful to only rebuild the installer layout after a previous call to buildrelease.bat.

Specify -o to set an output directory. The installer layouts will be copied to platform-specific subdirectories of this path.

Specify -c to choose a code-signing certificate to be used for all the signable binaries in Python as well as each file making up the installer. Official releases of Python must be signed.

Ensure %PYTHON% and %SPHINXBUILD% are set when passing this option. You may also set %HTMLHELP% to the Html Help Compiler (hhc.exe), or put HHC on your PATH or in externals/. You will also need Mercurial (hg.exe) on your PATH.

If WiX is not found on your system, it will be automatically downloaded and extracted to the externals/ directory.

To manually build layouts of the installer, build one of the projects in the bundle folder.

```
msbuild bundle\snapshot.wixproj
msbuild bundle\releaseweb.wixproj
msbuild bundle\releaselocal.wixproj
msbuild bundle\full.wixproj
```

snapshot.wixproj produces a test installer versioned based on the date.

releaseweb.wixproj produces a release installer that does not embed any of the layout.

releaselocal.wixproj produces a release installer that embeds the files required for a default installation.

full.wixproj produces a test installer that embeds the entire layout.

The following properties may be passed when building these projects.

/p:BuildForRelease=(true|false) When true, adds extra verification to ensure a complete installer is produced. For example, binutils is required when building for a release to generate MinGW-compatible libraries, and the build will be aborted if this fails. Defaults to false.

/p:ReleaseUri=(any URI) Used to generate unique IDs for the installers to allow side-by-side installation. Forks of Python can use the same installer infrastructure by providing a unique URI for this property. It does not need to be an active internet address. Defaults to $(ComputerName).

```
Official releases use http://www.python.org/(architecture name)
```

/p:DownloadUrlBase=(any URI) Specifies the base of a URL where missing parts of the installer layout can be downloaded from. The build version and architecture will be appended to create the full address. If omitted, missing components will not be automatically downloaded.

/p:DownloadUrl=(any URI) Specifies the full URL where missing parts of the installer layout can be downloaded from. Should normally include '{2}', which will be substituted for the filename. If omitted, missing components will not be automatically downloaded. If specified, this value overrides DownloadUrlBase.

/p:SigningCertificate=(certificate name) Specifies the certificate to sign the installer layout with. If omitted, the layout will not be signed.

/p:RebuildAll=(true|false) When true, rebuilds all of the MSIs making up the layout. Defaults to true.

Modifying the Installer

The code for the installer is divided into three main groups: packages, the bundle and the bootstrap application.

Packages

Packages appear as subdirectories of Tools/msi (other than the bundle/ directory). The project file is a .wixproj and the build output is a single MSI. Packages are built with the WiX Toolset. Some project files share source files and use preprocessor directives to enable particular features. These are typically used to keep the sources close when the files are related, but produce multiple independent packages.

A package is the smallest element that may be independently installed or uninstalled (as used in this installer). For example, the test suite has its own package, as users can choose to add or remove it after the initial installation.

All the files installed by a single package should be related, though some packages may not install any files. For example, the pip package executes the ensurepip package, but does not add or remove any of its own files. (It is represented as a package because of its installed/uninstalled nature, as opposed to the "precompile standard library" option, for example.) Dependencies between packages are handled by the bundle, but packages should detect when dependencies are missing and raise an error.

Packages that include a lot of files may use an InstallFiles element in the .wixproj file to generate sources. See lib/lib.wixproj for an example, and msi.targets and csv_to_wxs.py for the implementation. This element is also responsible for generating the code for cleaning up and removing **pycache** folders in any directory containing .py files.

All packages are built with the Tools/msi/common.wxs file, and so any directory or property in this file may be referenced. Of particular interest:

REGISTRYKEY (property) The registry key for the current installation.

InstallDirectory (directory) The root install directory for the current installation. Subdirectories are also specified in this file (DLLs, Lib, etc.)

MenuDir (directory) The Start Menu folder for the current installation.

UpgradeTable (property) Every package should reference this property to include upgrade information.

The .wxl_template file is specially handled by the build system for this project to perform {{substitutions}} as defined in msi.targets. They should be included in projects as items, where .wxl files are normally included as items.

Bundle

The bundle is compiled to the main EXE entry point that for most users will represent the Python installer. It is built from Tools/msi/bundle with packages references in Tools/msi/bundle/packagegroups.

Build logic for the bundle is in bundle.targets, but should be invoked through one of the .wixproj files as described in Building the Installer.

The UI is separated between Default.thm (UI layout), Default.wxl (strings), bundle.wxs (properties) and the bootstrap application. Bundle.wxs also contains the chain, which is the list of packages to install and the order they should be installed in. These refer to named package groups in bundle/packagegroups.

Each package group specifies one or more packages to install. Most packages require two separate entries to support both per-user and all-users installations. Because these reuse the same package, it does not increase the overall size of the package.

Package groups refer to payload groups, which allow better control over embedding and downloading files than the default settings. Whether files are embedded and where they are downloaded from depends on settings created by the project files.

Package references can include install conditions that determine when to install the package. When a package is a dependency for others, the condition should be crafted to ensure it is installed.

MSI packages are installed or uninstalled based on their current state and the install condition. This makes them most suitable for features that are clearly present or absent from the user's machine.

EXE packages are executed based on a customisable condition that can be omitted. This makes them suitable for pre- or post-install tasks that need to run regardless of whether features have been added or removed.

Bootstrap Application

The bootstrap application is a C++ application that controls the UI and installation. While it does not directly compile into the main EXE of the installer, it forms the main active component. Most of the installation functionality is provided by WiX, and so the bootstrap application is predominantly responsible for the code behind the UI that is defined in the Default.thm file. The bootstrap application code is in bundle/bootstrap and is built automatically when building the bundle.

Installation Layout

There are two installation layouts for Python on Windows, with the only differences being supporting files. A layout is selected implicitly based on whether the install is for all users of the machine or just for the user performing the installation.

The default installation location when installing for all users is "%ProgramFiles%3.X" for the 64-bit interpreter and "%ProgramFiles(x86)%3.X" for the 32-bit interpreter. (Note that the latter path is equivalent to "%ProgramFiles%3.X" when running a 32-bit version of Windows.) This location requires administrative privileges to install or later modify the installation.

The default installation location when installing for the current user is "%LocalAppData%3X" for the 64-bit interpreter and "%LocalAppData%3X-32" for the 32-bit interpreter. Only the current user can access this location. This provides a suitable level of protection against malicious modification of Python's files.

Within this install directory is the following approximate layout:

..exe The core executable files .Stdlib extensions (*.pyd) and dependencies .Documentation (*.chm) .Development headers (*.h) .Standard library .Test suite .Development libraries (*.lib) .Launcher scripts (*.exe, .py) .Tcl dependencies (*.dll, .tcl and others) .Tool scripts (*.py)

When installed for all users, the following files are installed to either "%SystemRoot%32" or "%SystemRoot%64" as appropriate. For the current user, they are installed in the Python install directory.

.3x.dll The core interpreter .3.dll The stable ABI reference .140.dll Microsoft Visual C Runtime .140.dll Microsoft Visual C Runtime .140.dll Microsoft Visual C Runtime

When installed for all users, the following files are installed to "%SystemRoot%" (typically "C:") to ensure they are always available on PATH. (See Launching Python below.) For the current user, they are installed in the Python install directory.

..exe PEP 397 launcher

System Settings

On installation, registry keys are created so that other applications can locate and identify installations of Python. The locations of these keys vary based on the install type.

For 64-bit interpreters installed for all users, the root key is: HKEY_LOCAL_MACHINE.X

For 32-bit interpreters installed for all users on a 64-bit operating system, the root key is: HKEY_LOCAL_MACHINE6432Node.X-32

For 32-bit interpreters installed for all users on a 32-bit operating system, the root key is: HKEY_LOCAL_MACHINE.X-32

For 64-bit interpreters installed for the current user: HKEY_CURRENT_USER.X

For 32-bit interpreters installed for the current user: HKEY_CURRENT_USER.X-32

When the core Python executables are installed, a key "InstallPath" is created within the root key with its default value set to the executable's install directory. Within this key, a key "InstallGroup" is created with its default value set to the product name "Python 3.X".

When the Python standard library is installed, a key "PythonPath" is created within the root key with its default value set to the full path to the Lib folder followed by the path to the DLLs folder, separated by a semicolon.

When the documentation is installed, a key "Help" is created within the root key, with a subkey "Main Python Documentation" with its default value set to the full path to the installed CHM file.

The py.exe launcher is installed as part of a regular Python install, but using a separate mechanism that allows it to more easily span versions of Python. As a result, it has different root keys for its registry entries:

When installed for all users on a 64-bit operating system, the launcher's root key is: HKEY_LOCAL_MACHINE6432Node

When installed for all users on a 32-bit operating system, the launcher's root key is: HKEY_LOCAL_MACHINE

When installed for the current user: HKEY_CURRENT_USER

When the launcher is installed, a key "InstallPath" is created within its root key with its default value set to the launcher's install directory. File associations are also created for .py, .pyw, .pyc and .pyo files.

Launching Python

When a feature offering user entry points in the Start Menu is installed, a folder "Python 3.X" is created. Every shortcut should be created within this folder, and each shortcut should include the version and platform to allow users to identify the shortcut in a search results page.

The core Python executables creates a shortcut "Python 3.X (32-bit)" or "Python 3.X (64-bit)" depending on the interpreter.

The documentation creates a shortcut "Python 3.X 32-bit Manuals" or "Python 3.X 64-bit Manuals". The documentation is identical for all platforms, but the shortcuts need to be separate to avoid uninstallation conflicts.

Installing IDLE creates a shortcut "IDLE (Python 3.X 32-bit)" or "IDLE (Python 3.X 64-bit)" depending on the interpreter.

For users who often launch Python from a Command Prompt, an option is provided to add the directory containing python.exe to the user or system PATH variable. If the option is selected, the install directory and the Scripts directory will be added at the start of the system PATH for an all users install and the user PATH for a per-user install.

When the user only has one version of Python installed, this will behave as expected. However, because Windows searches the system PATH before the user PATH, users cannot override a system-wide installation of Python on their PATH. Further, because the installer can only prepend to the path, later installations of Python will take precedence over earlier installations, regardless of interpreter version.

Because it is not possible to automatically create a sensible PATH configuration, users are recommended to use the py.exe launcher and manually modify their PATH variable to add Scripts directories in their preferred order. System-wide installations of Python should consider not modifying PATH, or using an alternative technology to modify their users' PATH variables.

The py.exe launcher is recommended because it uses a consistent and sensible search order for Python installations. User installations are preferred over system-wide installs, and later versions are preferred regardless of installation order (with the exception that py.exe currently prefers 2.x versions over 3.x versions without the -3 command line argument).

For both 32-bit and 64-bit interpreters, the 32-bit version of the launcher is installed. This ensures that the search order is always consistent (as the 64-bit launcher is subtly different from the 32-bit launcher) and also avoids the need to install it multiple times. Future versions of Python will upgrade the launcher in-place, using Windows Installer's upgrade functionality to avoid conflicts with earlier installed versions.

When installed, file associations are created for .py, .pyc and .pyo files to launch with py.exe and .pyw files to launch with pyw.exe. This makes Python files respect shebang lines by default and also avoids conflicts between multiple Python installations.

Repair, Modify and Uninstall

After installation, Python may be modified, repaired or uninstalled by running the original EXE again or via the Programs and Features applet (formerly known as Add or Remove Programs).

Modifications allow features to be added or removed. The install directory and kind (all users/single user) cannot be modified. Because Windows Installer caches installation packages, removing features will not require internet access unless the package cache has been corrupted or deleted. Adding features that were not previously installed and are not embedded or otherwise available will require internet access.

Repairing will rerun the installation for all currently installed features, restoring files and registry keys that have been modified or removed. This operation generally will not redownload any files unless the cached packages have been corrupted or deleted.

Removing Python will clean up all the files and registry keys that were created by the installer, as well as **pycache** folders that are explicitly handled by the installer. Python packages installed later using a tool like pip will not be removed. Some components may be installed by other installers (such as the MSVCRT) and these will not be removed if another product has a dependency on them.

HTML 4.0 color names

Black #000000 Silver #c0c0c0 Gray #808080 White #ffffff Maroon #800000 Red #ff0000 Purple #800080 Fuchsia #ff00ff Green #008000 Lime #00ff00 Olive #808000 Yellow #ffff00 Navy #000080 Blue #0000ff Teal #008080 Aqua #00ffff

named colors from http://www.lightlink.com/xine/bells/namedcolors.html

White #FFFFFF Red #FF0000 Green #00FF00 Blue #0000FF Magenta #FF00FF Cyan #00FFFF Yellow #FFFF00 Black #000000 Aquamarine #70DB93 Baker's Chocolate #5C3317 Blue Violet #9F5F9F Brass #B5A642 Bright Gold #D9D919 Brown #A62A2A Bronze #8C7853 Bronze II #A67D3D Cadet Blue #5F9F9F Cool Copper #D98719 Copper #B87333 Coral #FF7F00 Corn Flower Blue #42426F Dark Brown #5C4033 Dark Green #2F4F2F Dark Green Copper #4A766E Dark Olive Green #4F4F2F Dark Orchid #9932CD Dark Purple #871F78 Dark Slate Blue #6B238E Dark Slate Grey #2F4F4F Dark Tan #97694F Dark Turquoise #7093DB Dark Wood #855E42 Dim Grey #545454 Dusty Rose #856363 Feldspar #D19275 Firebrick #8E2323 Forest Green #238E23 Gold #CD7F32 Goldenrod #DBDB70 Grey #C0C0C0 Green Copper #527F76 Green Yellow #93DB70 Hunter Green #215E21 Indian Red #4E2F2F Khaki #9F9F5F Light Blue #C0D9D9 Light Grey #A8A8A8 Light Steel Blue #8F8FBD Light Wood #E9C2A6 Lime Green #32CD32 Mandarian Orange #E47833 Maroon #8E236B Medium Aquamarine #32CD99 Medium Blue #3232CD Medium Forest Green #6B8E23 Medium Goldenrod #EAEAAE Medium Orchid #9370DB Medium Sea Green #426F42 Medium Slate Blue #7F00FF Medium Spring Green #7FFF00 Medium Turquoise #70DBDB Medium Violet Red #DB7093 Medium Wood #A68064 Midnight Blue #2F2F4F Navy Blue #23238E Neon Blue #4D4DFF Neon Pink #FF6EC7 New Midnight Blue #00009C New Tan #EBC79E Old Gold #CFB53B Orange #FF7F00 Orange Red #FF2400 Orchid #DB70DB Pale Green #8FBC8F Pink #BC8F8F Plum #EAADEA Quartz #D9D9F3 Rich Blue #5959AB Salmon #6F4242 Scarlet #8C1717 Sea Green #238E68 Semi-Sweet Chocolate #6B4226 Sienna #8E6B23 Silver #E6E8FA Sky Blue #3299CC Slate Blue #007FFF Spicy Pink #FF1CAE Spring Green #00FF7F Steel Blue #236B8E Summer Sky #38B0DE Tan #DB9370 Thistle #D8BFD8 Turquoise #ADEAEA Very Dark Brown #5C4033 Very Light Grey #CDCDCD Violet #4F2F4F Violet Red #CC3299 Wheat #D8D8BF Yellow Green #99CC32

De-facto NS & MSIE recognized HTML color names

AliceBlue #f0f8ff AntiqueWhite #faebd7 Aqua #00ffff Aquamarine #7fffd4 Azure #f0ffff Beige #f5f5dc Bisque #ffe4c4 Black #000000 BlanchedAlmond #ffebcd Blue #0000ff BlueViolet #8a2be2 Brown #a52a2a BurlyWood #deb887 CadetBlue #5f9ea0 Chartreuse #7fff00 Chocolate #d2691e Coral #ff7f50 CornflowerBlue #6495ed Cornsilk #fff8dc Crimson #dc143c Cyan #00ffff DarkBlue #00008b DarkCyan #008b8b DarkGoldenrod #b8860b DarkGray #a9a9a9 DarkGreen #006400 DarkKhaki #bdb76b DarkMagenta #8b008b DarkOliveGreen #556b2f DarkOrange #ff8c00 DarkOrchid #9932cc DarkRed #8b0000 DarkSalmon #e9967a DarkSeaGreen #8fbc8f DarkSlateBlue #483d8b DarkSlateGray #2f4f4f DarkTurquoise #00ced1 DarkViolet #9400d3 DeepPink #ff1493 DeepSkyBlue #00bfff DimGray #696969 DodgerBlue #1e90ff FireBrick #b22222 FloralWhite #fffaf0 ForestGreen #228b22 Fuchsia #ff00ff Gainsboro #dcdcdc GhostWhite #f8f8ff Gold #ffd700 Goldenrod #daa520 Gray #808080 Green #008000 GreenYellow #adff2f Honeydew #f0fff0 HotPink #ff69b4 IndianRed #cd5c5c Indigo #4b0082 Ivory #fffff0 Khaki #f0e68c Lavender #e6e6fa LavenderBlush #fff0f5 LawnGreen #7cfc00 LemonChiffon #fffacd LightBlue #add8e6 LightCoral #f08080 LightCyan #e0ffff LightGoldenrodYellow #fafad2 LightGreen #90ee90 LightGrey #d3d3d3 LightPink #ffb6c1 LightSalmon #ffa07a LightSeaGreen #20b2aa LightSkyBlue #87cefa LightSlateGray #778899 LightSteelBlue #b0c4de LightYellow #ffffe0 Lime #00ff00 LimeGreen #32cd32 Linen #faf0e6 Magenta #ff00ff Maroon #800000 MediumAquamarine #66cdaa MediumBlue #0000cd MediumOrchid #ba55d3 MediumPurple #9370db MediumSeaGreen #3cb371 MediumSlateBlue #7b68ee MediumSpringGreen #00fa9a MediumTurquoise #48d1cc MediumVioletRed #c71585 MidnightBlue #191970 MintCream #f5fffa MistyRose #ffe4e1 Moccasin #ffe4b5 NavajoWhite #ffdead Navy #000080 OldLace #fdf5e6 Olive #808000 OliveDrab #6b8e23 Orange #ffa500 OrangeRed #ff4500 Orchid #da70d6 PaleGoldenrod #eee8aa PaleGreen #98fb98 PaleTurquoise #afeeee PaleVioletRed #db7093 PapayaWhip #ffefd5 PeachPuff #ffdab9 Peru #cd853f Pink #ffc0cb Plum #dda0dd PowderBlue #b0e0e6 Purple #800080 Red #ff0000 RosyBrown #bc8f8f RoyalBlue #4169e1 SaddleBrown #8b4513 Salmon #fa8072 SandyBrown #f4a460 SeaGreen #2e8b57 Seashell #fff5ee Sienna #a0522d Silver #c0c0c0 SkyBlue #87ceeb SlateBlue #6a5acd SlateGray #708090 Snow #fffafa SpringGreen #00ff7f SteelBlue #4682b4 Tan #d2b48c Teal #008080 Thistle #d8bfd8 Tomato #ff6347 Turquoise #40e0d0 Violet #ee82ee Wheat #f5deb3 White #ffffff WhiteSmoke #f5f5f5 Yellow #ffff00 YellowGreen #9acd32

Websafe RGB values

000000

000033

000066

000099

0000cc

0000ff

003300

003333

003366

003399

0033cc

0033ff

006600

006633

006666

006699

0066cc

0066ff

009900

009933

009966

009999

0099cc

0099ff

00cc00

00cc33

00cc66

00cc99

00cccc

00ccff

00ff00

00ff33

00ff66

00ff99

00ffcc

00ffff

330000

330033

330066

330099

3300cc

3300ff

333300

333333

333366

333399

3333cc

3333ff

336600

336633

336666

336699

3366cc

3366ff

339900

339933

339966

339999

3399cc

3399ff

33cc00

33cc33

33cc66

33cc99

33cccc

33ccff

33ff00

33ff33

33ff66

33ff99

33ffcc

33ffff

660000

660033

660066

660099

6600cc

6600ff

663300

663333

663366

663399

6633cc

6633ff

666600

666633

666666

666699

6666cc

6666ff

669900

669933

669966

669999

6699cc

6699ff

66cc00

66cc33

66cc66

66cc99

66cccc

66ccff

66ff00

66ff33

66ff66

66ff99

66ffcc

66ffff

990000

990033

990066

990099

9900cc

9900ff

993300

993333

993366

993399

9933cc

9933ff

996600

996633

996666

996699

9966cc

9966ff

999900

999933

999966

999999

9999cc

9999ff

99cc00

99cc33

99cc66

99cc99

99cccc

99ccff

99ff00

99ff33

99ff66

99ff99

99ffcc

99ffff

cc0000

cc0033

cc0066

cc0099

cc00cc

cc00ff

cc3300

cc3333

cc3366

cc3399

cc33cc

cc33ff

cc6600

cc6633

cc6666

cc6699

cc66cc

cc66ff

cc9900

cc9933

cc9966

cc9999

cc99cc

cc99ff

cccc00

cccc33

cccc66

cccc99

cccccc

ccccff

ccff00

ccff33

ccff66

ccff99

ccffcc

ccffff

ff0000

ff0033

ff0066

ff0099

ff00cc

ff00ff

ff3300

ff3333

ff3366

ff3399

ff33cc

ff33ff

ff6600

ff6633

ff6666

ff6699

ff66cc

ff66ff

ff9900

ff9933

ff9966

ff9999

ff99cc

ff99ff

ffcc00

ffcc33

ffcc66

ffcc99

ffcccc

ffccff

ffff00

ffff33

ffff66

ffff99

ffffcc

ffffff

! $XConsortium: rgb.txt,v 10.41 94/02/20 18:39:36 rws Exp $ 255 250 250 snow 248 248 255 ghost white 248 248 255 GhostWhite 245 245 245 white smoke 245 245 245 WhiteSmoke 220 220 220 gainsboro 255 250 240 floral white 255 250 240 FloralWhite 253 245 230 old lace 253 245 230 OldLace 250 240 230 linen 250 235 215 antique white 250 235 215 AntiqueWhite 255 239 213 papaya whip 255 239 213 PapayaWhip 255 235 205 blanched almond 255 235 205 BlanchedAlmond 255 228 196 bisque 255 218 185 peach puff 255 218 185 PeachPuff 255 222 173 navajo white 255 222 173 NavajoWhite 255 228 181 moccasin 255 248 220 cornsilk 255 255 240 ivory 255 250 205 lemon chiffon 255 250 205 LemonChiffon 255 245 238 seashell 240 255 240 honeydew 245 255 250 mint cream 245 255 250 MintCream 240 255 255 azure 240 248 255 alice blue 240 248 255 AliceBlue 230 230 250 lavender 255 240 245 lavender blush 255 240 245 LavenderBlush 255 228 225 misty rose 255 228 225 MistyRose 255 255 255 white 0 0 0 black 47 79 79 dark slate gray 47 79 79 DarkSlateGray 47 79 79 dark slate grey 47 79 79 DarkSlateGrey 105 105 105 dim gray 105 105 105 DimGray 105 105 105 dim grey 105 105 105 DimGrey 112 128 144 slate gray 112 128 144 SlateGray 112 128 144 slate grey 112 128 144 SlateGrey 119 136 153 light slate gray 119 136 153 LightSlateGray 119 136 153 light slate grey 119 136 153 LightSlateGrey 190 190 190 gray 190 190 190 grey 211 211 211 light grey 211 211 211 LightGrey 211 211 211 light gray 211 211 211 LightGray 25 25 112 midnight blue 25 25 112 MidnightBlue 0 0 128 navy 0 0 128 navy blue 0 0 128 NavyBlue 100 149 237 cornflower blue 100 149 237 CornflowerBlue 72 61 139 dark slate blue 72 61 139 DarkSlateBlue 106 90 205 slate blue 106 90 205 SlateBlue 123 104 238 medium slate blue 123 104 238 MediumSlateBlue 132 112 255 light slate blue 132 112 255 LightSlateBlue 0 0 205 medium blue 0 0 205 MediumBlue 65 105 225 royal blue 65 105 225 RoyalBlue 0 0 255 blue 30 144 255 dodger blue 30 144 255 DodgerBlue 0 191 255 deep sky blue 0 191 255 DeepSkyBlue 135 206 235 sky blue 135 206 235 SkyBlue 135 206 250 light sky blue 135 206 250 LightSkyBlue 70 130 180 steel blue 70 130 180 SteelBlue 176 196 222 light steel blue 176 196 222 LightSteelBlue 173 216 230 light blue 173 216 230 LightBlue 176 224 230 powder blue 176 224 230 PowderBlue 175 238 238 pale turquoise 175 238 238 PaleTurquoise 0 206 209 dark turquoise 0 206 209 DarkTurquoise 72 209 204 medium turquoise 72 209 204 MediumTurquoise 64 224 208 turquoise 0 255 255 cyan 224 255 255 light cyan 224 255 255 LightCyan 95 158 160 cadet blue 95 158 160 CadetBlue 102 205 170 medium aquamarine 102 205 170 MediumAquamarine 127 255 212 aquamarine 0 100 0 dark green 0 100 0 DarkGreen 85 107 47 dark olive green 85 107 47 DarkOliveGreen 143 188 143 dark sea green 143 188 143 DarkSeaGreen 46 139 87 sea green 46 139 87 SeaGreen 60 179 113 medium sea green 60 179 113 MediumSeaGreen 32 178 170 light sea green 32 178 170 LightSeaGreen 152 251 152 pale green 152 251 152 PaleGreen 0 255 127 spring green 0 255 127 SpringGreen 124 252 0 lawn green 124 252 0 LawnGreen 0 255 0 green 127 255 0 chartreuse 0 250 154 medium spring green 0 250 154 MediumSpringGreen 173 255 47 green yellow 173 255 47 GreenYellow 50 205 50 lime green 50 205 50 LimeGreen 154 205 50 yellow green 154 205 50 YellowGreen 34 139 34 forest green 34 139 34 ForestGreen 107 142 35 olive drab 107 142 35 OliveDrab 189 183 107 dark khaki 189 183 107 DarkKhaki 240 230 140 khaki 238 232 170 pale goldenrod 238 232 170 PaleGoldenrod 250 250 210 light goldenrod yellow 250 250 210 LightGoldenrodYellow 255 255 224 light yellow 255 255 224 LightYellow 255 255 0 yellow 255 215 0 gold 238 221 130 light goldenrod 238 221 130 LightGoldenrod 218 165 32 goldenrod 184 134 11 dark goldenrod 184 134 11 DarkGoldenrod 188 143 143 rosy brown 188 143 143 RosyBrown 205 92 92 indian red 205 92 92 IndianRed 139 69 19 saddle brown 139 69 19 SaddleBrown 160 82 45 sienna 205 133 63 peru 222 184 135 burlywood 245 245 220 beige 245 222 179 wheat 244 164 96 sandy brown 244 164 96 SandyBrown 210 180 140 tan 210 105 30 chocolate 178 34 34 firebrick 165 42 42 brown 233 150 122 dark salmon 233 150 122 DarkSalmon 250 128 114 salmon 255 160 122 light salmon 255 160 122 LightSalmon 255 165 0 orange 255 140 0 dark orange 255 140 0 DarkOrange 255 127 80 coral 240 128 128 light coral 240 128 128 LightCoral 255 99 71 tomato 255 69 0 orange red 255 69 0 OrangeRed 255 0 0 red 255 105 180 hot pink 255 105 180 HotPink 255 20 147 deep pink 255 20 147 DeepPink 255 192 203 pink 255 182 193 light pink 255 182 193 LightPink 219 112 147 pale violet red 219 112 147 PaleVioletRed 176 48 96 maroon 199 21 133 medium violet red 199 21 133 MediumVioletRed 208 32 144 violet red 208 32 144 VioletRed 255 0 255 magenta 238 130 238 violet 221 160 221 plum 218 112 214 orchid 186 85 211 medium orchid 186 85 211 MediumOrchid 153 50 204 dark orchid 153 50 204 DarkOrchid 148 0 211 dark violet 148 0 211 DarkViolet 138 43 226 blue violet 138 43 226 BlueViolet 160 32 240 purple 147 112 219 medium purple 147 112 219 MediumPurple 216 191 216 thistle 255 250 250 snow1 238 233 233 snow2 205 201 201 snow3 139 137 137 snow4 255 245 238 seashell1 238 229 222 seashell2 205 197 191 seashell3 139 134 130 seashell4 255 239 219 AntiqueWhite1 238 223 204 AntiqueWhite2 205 192 176 AntiqueWhite3 139 131 120 AntiqueWhite4 255 228 196 bisque1 238 213 183 bisque2 205 183 158 bisque3 139 125 107 bisque4 255 218 185 PeachPuff1 238 203 173 PeachPuff2 205 175 149 PeachPuff3 139 119 101 PeachPuff4 255 222 173 NavajoWhite1 238 207 161 NavajoWhite2 205 179 139 NavajoWhite3 139 121 94 NavajoWhite4 255 250 205 LemonChiffon1 238 233 191 LemonChiffon2 205 201 165 LemonChiffon3 139 137 112 LemonChiffon4 255 248 220 cornsilk1 238 232 205 cornsilk2 205

200 177 cornsilk3 139 136 120 cornsilk4 255 255 240 ivory1 238 238 224 ivory2 205 205 193 ivory3 139 139 131 ivory4 240 255 240 honeydew1 224 238 224 honeydew2 193 205 193 honeydew3 131 139 131 honeydew4 255 240 245 LavenderBlush1 238 224 229 LavenderBlush2 205 193 197 LavenderBlush3 139 131 134 LavenderBlush4 255 228 225 MistyRose1 238 213 210 MistyRose2 205 183 181 MistyRose3 139 125 123 MistyRose4 240 255 255 azure1 224 238 238 azure2 193 205 205 azure3 131 139 139 azure4 131 111 255 SlateBlue1 122 103 238 SlateBlue2 105 89 205 SlateBlue3 71 60 139 SlateBlue4 72 118 255 RoyalBlue1 67 110 238 RoyalBlue2 58 95 205 RoyalBlue3 39 64 139 RoyalBlue4 0 0 255 blue1 0 0 238 blue2 0 0 205 blue3 0 0 139 blue4 30 144 255 DodgerBlue1 28 134 238 DodgerBlue2 24 116 205 DodgerBlue3 16 78 139 DodgerBlue4 99 184 255 SteelBlue1 92 172 238 SteelBlue2 79 148 205 SteelBlue3 54 100 139 SteelBlue4 0 191 255 DeepSkyBlue1 0 178 238 DeepSkyBlue2 0 154 205 DeepSkyBlue3 0 104 139 DeepSkyBlue4 135 206 255 SkyBlue1 126 192 238 SkyBlue2 108 166 205 SkyBlue3 74 112 139 SkyBlue4 176 226 255 LightSkyBlue1 164 211 238 LightSkyBlue2 141 182 205 LightSkyBlue3 96 123 139 LightSkyBlue4 198 226 255 SlateGray1 185 211 238 SlateGray2 159 182 205 SlateGray3 108 123 139 SlateGray4 202 225 255 LightSteelBlue1 188 210 238 LightSteelBlue2 162 181 205 LightSteelBlue3 110 123 139 LightSteelBlue4 191 239 255 LightBlue1 178 223 238 LightBlue2 154 192 205 LightBlue3 104 131 139 LightBlue4 224 255 255 LightCyan1 209 238 238 LightCyan2 180 205 205 LightCyan3 122 139 139 LightCyan4 187 255 255 PaleTurquoise1 174 238 238 PaleTurquoise2 150 205 205 PaleTurquoise3 102 139 139 PaleTurquoise4 152 245 255 CadetBlue1 142 229 238 CadetBlue2 122 197 205 CadetBlue3 83 134 139 CadetBlue4 0 245 255 turquoise1 0 229 238 turquoise2 0 197 205 turquoise3 0 134 139 turquoise4 0 255 255 cyan1 0 238 238 cyan2 0 205 205 cyan3 0 139 139 cyan4 151 255 255 DarkSlateGray1 141 238 238 DarkSlateGray2 121 205 205 DarkSlateGray3 82 139 139 DarkSlateGray4 127 255 212 aquamarine1 118 238 198 aquamarine2 102 205 170 aquamarine3 69 139 116 aquamarine4 193 255 193 DarkSeaGreen1 180 238 180 DarkSeaGreen2 155 205 155 DarkSeaGreen3 105 139 105 DarkSeaGreen4 84 255 159 SeaGreen1 78 238 148 SeaGreen2 67 205 128 SeaGreen3 46 139 87 SeaGreen4 154 255 154 PaleGreen1 144 238 144 PaleGreen2 124 205 124 PaleGreen3 84 139 84 PaleGreen4 0 255 127 SpringGreen1 0 238 118 SpringGreen2 0 205 102 SpringGreen3 0 139 69 SpringGreen4 0 255 0 green1 0 238 0 green2 0 205 0 green3 0 139 0 green4 127 255 0 chartreuse1 118 238 0 chartreuse2 102 205 0 chartreuse3 69 139 0 chartreuse4 192 255 62 OliveDrab1 179 238 58 OliveDrab2 154 205 50 OliveDrab3 105 139 34 OliveDrab4 202 255 112 DarkOliveGreen1 188 238 104 DarkOliveGreen2 162 205 90 DarkOliveGreen3 110 139 61 DarkOliveGreen4 255 246 143 khaki1 238 230 133 khaki2 205 198 115 khaki3 139 134 78 khaki4 255 236 139 LightGoldenrod1 238 220 130 LightGoldenrod2 205 190 112 LightGoldenrod3 139 129 76 LightGoldenrod4 255 255 224 LightYellow1 238 238 209 LightYellow2 205 205 180 LightYellow3 139 139 122 LightYellow4 255 255 0 yellow1 238 238 0 yellow2 205 205 0 yellow3 139 139 0 yellow4 255 215 0 gold1 238 201 0 gold2 205 173 0 gold3 139 117 0 gold4 255 193 37 goldenrod1 238 180 34 goldenrod2 205 155 29 goldenrod3 139 105 20 goldenrod4 255 185 15 DarkGoldenrod1 238 173 14 DarkGoldenrod2 205 149 12 DarkGoldenrod3 139 101 8 DarkGoldenrod4 255 193 193 RosyBrown1 238 180 180 RosyBrown2 205 155 155 RosyBrown3 139 105 105 RosyBrown4 255 106 106 IndianRed1 238 99 99 IndianRed2 205 85 85 IndianRed3 139 58 58 IndianRed4 255 130 71 sienna1 238 121 66 sienna2 205 104 57 sienna3 139 71 38 sienna4 255 211 155 burlywood1 238 197 145 burlywood2 205 170 125 burlywood3 139 115 85 burlywood4 255 231 186 wheat1 238 216 174 wheat2 205 186 150 wheat3 139 126 102 wheat4 255 165 79 tan1 238 154 73 tan2 205 133 63 tan3 139 90 43 tan4 255 127 36 chocolate1 238 118 33 chocolate2 205 102 29 chocolate3 139 69 19 chocolate4 255 48 48 firebrick1 238 44 44 firebrick2 205 38 38 firebrick3 139 26 26 firebrick4 255 64 64 brown1 238 59 59 brown2 205 51 51 brown3 139 35 35 brown4 255 140 105 salmon1 238 130 98 salmon2 205 112 84 salmon3 139 76 57 salmon4 255 160 122 LightSalmon1 238 149 114 LightSalmon2 205 129 98 LightSalmon3 139 87 66 LightSalmon4 255 165 0 orange1 238 154 0 orange2 205 133 0 orange3 139 90 0 orange4 255 127 0 DarkOrange1 238 118 0 DarkOrange2 205 102 0 DarkOrange3 139 69 0 DarkOrange4 255 114 86 coral1 238 106 80 coral2 205 91 69 coral3 139 62 47 coral4 255 99 71 tomato1 238 92 66 tomato2 205 79 57 tomato3 139 54 38 tomato4 255 69 0 OrangeRed1 238 64 0 OrangeRed2 205 55 0 OrangeRed3 139 37 0 OrangeRed4 255 0 0 red1 238 0 0 red2 205 0 0 red3 139 0 0 red4 255 20 147 DeepPink1 238 18 137 DeepPink2 205 16 118 DeepPink3 139 10 80 DeepPink4 255 110 180 HotPink1 238 106 167 HotPink2 205 96 144 HotPink3 139 58 98 HotPink4 255 181 197 pink1 238 169 184 pink2 205 145 158 pink3 139 99 108 pink4 255 174 185 LightPink1 238 162 173 LightPink2 205 140 149 LightPink3 139 95 101 LightPink4 255 130 171 PaleVioletRed1 238 121 159 PaleVioletRed2 205 104 137 PaleVioletRed3 139 71 93 PaleVioletRed4 255 52 179 maroon1 238 48 167 maroon2 205 41 144 maroon3 139 28 98 maroon4 255 62 150 VioletRed1 238 58 140 VioletRed2 205 50 120 VioletRed3 139 34 82 VioletRed4 255 0 255 magenta1 238 0 238 magenta2 205 0 205 magenta3 139 0 139 magenta4 255 131 250 orchid1 238 122 233 orchid2 205 105 201 orchid3 139 71 137 orchid4 255 187 255 plum1 238 174 238 plum2 205 150 205 plum3 139 102 139 plum4 224 102 255 MediumOrchid1 209 95 238 MediumOrchid2 180 82 205 MediumOrchid3 122 55 139 MediumOrchid4 191 62 255 DarkOrchid1 178 58 238 DarkOrchid2 154

50 205 DarkOrchid3 104 34 139 DarkOrchid4 155 48 255 purple1 145 44 238 purple2 125 38 205 purple3 85 26 139 purple4 171 130 255 MediumPurple1 159 121 238 MediumPurple2 137 104 205 MediumPurple3 93 71 139 MediumPurple4 255 225 255 thistle1 238 210 238 thistle2 205 181 205 thistle3 139 123 139 thistle4 0 0 0 gray0 0 0 0 grey0 3 3 3 gray1 3 3 3 grey1 5 5 5 gray2 5 5 5 grey2 8 8 8 gray3 8 8 8 grey3 10 10 10 gray4 10 10 10 grey4 13 13 13 gray5 13 13 13 grey5 15 15 15 gray6 15 15 15 grey6 18 18 18 gray7 18 18 18 grey7 20 20 20 gray8 20 20 20 grey8 23 23 23 gray9 23 23 23 grey9 26 26 26 gray10 26 26 26 grey10 28 28 28 gray11 28 28 28 grey11 31 31 31 gray12 31 31 31 grey12 33 33 33 gray13 33 33 33 grey13 36 36 36 gray14 36 36 36 grey14 38 38 38 gray15 38 38 38 grey15 41 41 41 gray16 41 41 41 grey16 43 43 43 gray17 43 43 43 grey17 46 46 46 gray18 46 46 46 grey18 48 48 48 gray19 48 48 48 grey19 51 51 51 gray20 51 51 51 grey20 54 54 54 gray21 54 54 54 grey21 56 56 56 gray22 56 56 56 grey22 59 59 59 gray23 59 59 59 grey23 61 61 61 gray24 61 61 61 grey24 64 64 64 gray25 64 64 64 grey25 66 66 66 gray26 66 66 66 grey26 69 69 69 gray27 69 69 69 grey27 71 71 71 gray28 71 71 71 grey28 74 74 74 gray29 74 74 74 grey29 77 77 77 gray30 77 77 77 grey30 79 79 79 gray31 79 79 79 grey31 82 82 82 gray32 82 82 82 grey32 84 84 84 gray33 84 84 84 grey33 87 87 87 gray34 87 87 87 grey34 89 89 89 gray35 89 89 89 grey35 92 92 92 gray36 92 92 92 grey36 94 94 94 gray37 94 94 94 grey37 97 97 97 gray38 97 97 97 grey38 99 99 99 gray39 99 99 99 grey39 102 102 102 gray40 102 102 102 grey40 105 105 105 gray41 105 105 105 grey41 107 107 107 gray42 107 107 107 grey42 110 110 110 gray43 110 110 110 grey43 112 112 112 gray44 112 112 112 grey44 115 115 115 gray45 115 115 115 grey45 117 117 117 gray46 117 117 117 grey46 120 120 120 gray47 120 120 120 grey47 122 122 122 gray48 122 122 122 grey48 125 125 125 gray49 125 125 125 grey49 127 127 127 gray50 127 127 127 grey50 130 130 130 gray51 130 130 130 grey51 133 133 133 gray52 133 133 133 grey52 135 135 135 gray53 135 135 135 grey53 138 138 138 gray54 138 138 138 grey54 140 140 140 gray55 140 140 140 grey55 143 143 143 gray56 143 143 143 grey56 145 145 145 gray57 145 145 145 grey57 148 148 148 gray58 148 148 148 grey58 150 150 150 gray59 150 150 150 grey59 153 153 153 gray60 153 153 153 grey60 156 156 156 gray61 156 156 156 grey61 158 158 158 gray62 158 158 158 grey62 161 161 161 gray63 161 161 161 grey63 163 163 163 gray64 163 163 163 grey64 166 166 166 gray65 166 166 166 grey65 168 168 168 gray66 168 168 168 grey66 171 171 171 gray67 171 171 171 grey67 173 173 173 gray68 173 173 173 grey68 176 176 176 gray69 176 176 176 grey69 179 179 179 gray70 179 179 179 grey70 181 181 181 gray71 181 181 181 grey71 184 184 184 gray72 184 184 184 grey72 186 186 186 gray73 186 186 186 grey73 189 189 189 gray74 189 189 189 grey74 191 191 191 gray75 191 191 191 grey75 194 194 194 gray76 194 194 194 grey76 196 196 196 gray77 196 196 196 grey77 199 199 199 gray78 199 199 199 grey78 201 201 201 gray79 201 201 201 grey79 204 204 204 gray80 204 204 204 grey80 207 207 207 gray81 207 207 207 grey81 209 209 209 gray82 209 209 209 grey82 212 212 212 gray83 212 212 212 grey83 214 214 214 gray84 214 214 214 grey84 217 217 217 gray85 217 217 217 grey85 219 219 219 gray86 219 219 219 grey86 222 222 222 gray87 222 222 222 grey87 224 224 224 gray88 224 224 224 grey88 227 227 227 gray89 227 227 227 grey89 229 229 229 gray90 229 229 229 grey90 232 232 232 gray91 232 232 232 grey91 235 235 235 gray92 235 235 235 grey92 237 237 237 gray93 237 237 237 grey93 240 240 240 gray94 240 240 240 grey94 242 242 242 gray95 242 242 242 grey95 245 245 245 gray96 245 245 245 grey96 247 247 247 gray97 247 247 247 grey97 250 250 250 gray98 250 250 250 grey98 252 252 252 gray99 252 252 252 grey99 255 255 255 gray100 255 255 255 grey100 169 169 169 dark grey 169 169 169 DarkGrey 169 169 169 dark gray 169 169 169 DarkGray 0 0 139 dark blue 0 0 139 DarkBlue 0 139 139 dark cyan 0 139 139 DarkCyan 139 0 139 dark magenta 139 0 139 DarkMagenta 139 0 0 dark red 139 0 0 DarkRed 144 238 144 light green 144 238 144 LightGreen

X Window System License - X11R6.4

unittestgui.py is GUI framework and application for use with Python unit testing framework. It executes tests written using the framework provided by the 'unittest' module.

Based on the original by Steve Purcell, from:

http://pyunit.sourceforge.net/

Updated for unittest test discovery by Mark Roddy and Python 3 support by Brian Curtin.

For details on how to make your tests work with test discovery, and for explanations of the configuration options, see the unittest documentation:

`http://docs.python.org/library/unittest.html#test-discovery`

Table of Contents