
konch

Release 4.5.0

unknown

Jan 17, 2024

CONTENTS

1	About	3
2	Install/Upgrade	5
3	Usage	7
4	Configuration	11
5	Setup and Teardown Functions	13
6	IPython Extras	15
7	ptpython support	17
8	Using <code>.konchrc.local</code>	19
9	Programmatic Usage	21
10	Project Info	23

Release v4.5.0

changelog // github // pypi // issues

- *About*
- *Install/Upgrade*
- *Usage*
- *Configuration*
- *Setup and Teardown Functions*
- *IPython Extras*
- *ptpython support*
- *Using .konchrc.local*
- *Programmatic Usage*
- *Project Info*

ABOUT

konch is a CLI and configuration utility for the Python shell, optimized for simplicity and productivity.

- **Automatically import** any object upon startup
- **Simple**, per-project configuration in a single file (it's just Python code)
- **No external dependencies**
- Uses **IPython**, **BPython**, or **ptpython** if available, and falls back to built-in interpreter
- Automatically load **IPython extensions**

1.1 Example Use Cases

- If you're building a web app, you can have all your models in your shell namespace without having to import them individually. You can also run any necessary database or app setup.
- If you're building a Python package, you can automatically import all of its modules.
- In a live demo, skip the imports.
- Immediately have test objects to work with in interactive sessions.

INSTALL/UPGRADE

```
$ pip install -U konch
```

Supports Python>=3.8. There are no external dependencies.

Note: If you are using Python 2, `konch<4.0` will be installed with the above command.

3.1 \$ `konch init [<config_file>]`

creates a `.konchrc` file in your current directory.

`.konchrc` is a Python file that calls the `konch.config(config_dict)` function. See the [Configuration](#) section for a listing of options.

Here is an example `.konchrc` file that includes some functions from the [requests](#) library in its context.

```
# .konchrc
import konch
import requests

konch.config(
    {
        "context": [requests.get, requests.post, requests.put, requests.delete],
        "banner": "A humanistic HTTP shell",
    }
)
```

This would produce the following shell:

```
konch > konch
3.6.5 (default, Jul  9 2018, 09:55:35)
[GCC 4.2.1 Compatible Apple LLVM 9.1.0 (clang-902.0.39.2)]

A humanistic HTTP shell

Context:
delete: <function delete at 0x1050861e0>
get: <function get at 0x105085e18>
post: <function post at 0x105086048>
put: <function put at 0x1050860d0>

>>> get('https://httpbin.org/get').json()
{'args': {}, 'headers': {'Accept': '*/.*', 'Accept-Encoding': 'gzip',
'connection': 'close', 'Host': 'httpbin.org', 'User-Agent': 'python-
0'}, 'origin': '71.62.148.25', 'url': 'https://httpbin.org/get'}
>>> █
```

See also:

For more examples, see the [example_rcfiles](#) directory.

Note: `.konchrc` files created with `konch init` are automatically authorized.

Note: You can override the default file contents of `.konchrc` by creating a `~/.konchrc.default` file in your home directory.

3.2 \$ `konch edit [<config_file>]`

opens the current config file in your editor. Automatically authorizes the file when editing is finished.

Checks the `KONCH_EDITOR`, `VISUAL` and `EDITOR` environment variables (in that order) to determine which editor to use.

3.3 \$ `konch [-f <config_file>]`

runs the shell. Uses `.konchrc` as the configuration file, by default.

3.4 \$ `konch -s <shell>`

overrides the default shell. Choose between `ipy`, `bpy`, `py`, `ptpy`, `ptipy`, or `auto`.

3.5 \$ `konch allow [<config_file>]`

authorizes a config file. **You MUST authorize new or edited config files before executing them.** This is a security mechanism to prevent execution of untrusted code.

By default, the auth file is stored in `~/.local/share/konch_auth`. You can change the location by setting the `KONCH_AUTH_FILE` environment variable.

3.6 \$ `konch deny [<config_file>]`

removes authorization for a config file.

3.7 \$ `konch -n <name>`

selects a *named config*.

Named configs allow you to have multiple configurations in the same `.konchrc` file.

```
# .konchrc
import konch
import os
import sys
import requests
import flask
```

(continues on next page)

(continued from previous page)

```
# The default config
konch.config({"context": [os, sys]})

konch.named_config("http", {"context": [requests.get, requests.post]})

konch.named_config(
    "flask", {"context": [flask.Flask, flask.url_for, flask.render_template]}
)
```

To use the flask config, you would run:

```
$ konch -n flask
```

You can also pass multiple names to named_config:

```
# konch -n flask
# OR
# konch -n fl
konch.named_config(
    ["flask", "fl"], {"context": [flask.Flask, flask.url_for, flask.render_template]}
)
```


CONFIGURATION

- **context**: A dictionary or list of objects that will be available in your shell session. May also be a callable that returns a dictionary or list.
- **shell**: Default shell. May be 'ipy', 'bpy', 'ptpy', 'ptipy', 'py', or 'auto' (default). You can also pass a `Shell` class directly, such as `konch.IPythonShell`, `konch.BPythonShell`, `konch.PtPythonShell`, `konch.PtIPythonShell`, `konch.PythonShell`, or `konch.AutoShell`.
- **banner**: Custom banner text.
- **prompt**: The input prompt (not supported with BPython).
- **output**: (IPython and ptypython only) The output prompt.
- **context_format**: Format to display context. May be 'full', 'short', 'hide', or a function that receives the context dictionary as input and returns a string.
- **ipy_extensions**: (IPython and ptypython only) IPython extensions to load at startup.
- **ipy_autoreload**: (IPython and ptypython only) Whether to enable the IPython autoreload extension.
- **ipy_colors**: (IPython only) Color scheme.
- **ipy_highlighting_style**: (IPython only) Syntax highlighting style.
- **ptpy_vi_mode**: (ptpython and ptypython only) Whether to enable vim bindings.

SETUP AND TEARDOWN FUNCTIONS

You can optionally define `setup()` and/or `teardown()` functions which will execute immediately before and after running the shell, respectively.

```
from pathlib import Path
import shutil
import konch

def setup():
    Path("my_temp_dir").mkdir()

def teardown():
    shutil.rmtree("my_temp_dir")

konch.config({"context": [Path]})
```


IPYTHON EXTRAS

`konch` provides a few IPython-specific options.

6.1 Loading Extensions

The `ipy_extensions` option is used to automatically load IPython extensions at startup.

```
import konch

konch.config(
    {
        # ...
        "shell": "ipython",
        "ipy_extensions": ["autoreload", "rpy2.ipython"],
    }
)
```

6.2 Autoreload

The `ipy_autoreload` option enables and initializes the IPython `autoreload` extension at startup.

```
import konch

konch.config(
    {
        # ...
        "shell": "ipython",
        # Automatically reload modules
        "ipy_autoreload": True,
    }
)
```

This is equivalent to running:

```
% load_ext autoreload
% autoreload 2
```

6.3 Colors

The `ipy_colors` and `ipy_highlighting_style` options are used to configure colors in the IPython shell. `ipy_colors` sets the color of tracebacks and object info (the output of e.g. `zip?`). `ipy_highlighting_style` sets colors for syntax highlighting.

```
import konch

konch.config(
    {
        # ...
        "shell": "ipython",
        # 'linux' is optimized for dark terminal backgrounds
        "ipy_colors": "linux",
        "ipy_highlighting_style": "monokai",
    }
)
```

See the IPython docs for more information and valid values for these options: <https://ipython.readthedocs.io/en/stable/config/details.html#terminal-colors>

PTPYTHON SUPPORT

`konch` supports both `ptpython` and `ptipython`. If either is installed in your current environment, running `konch` will run the available shell.

`konch` provides a few `ptpython`-specific options.

To use `ptpython`'s vi-style bindings, set the `ptpy_vi_mode` option in your `.konchrc`. You can also use the `ipy_extensions` option to load IPython extensions at startup (must be using `ptipython`).

```
import konch

konch.config(
    {
        # ...
        "shell": "ptipython",
        "ptpy_vi_mode": True,
        "ipy_extensions": ["autoreload"],
    }
)
```


USING .KONCHRC.LOCAL

If you're distributing your `.konchrc` in a git repo, you may want to allow collaborators to extend your configuration in an unversioned `.konchrc.local` file.

First, add `.konchrc.local` to `.gitignore`.

```
# .gitignore
.konchrc.local
```

Then add the following to your `.konchrc`:

```
# .konchrc
from pathlib import Path

# konch.config(...)

if Path(".konchrc.local").exists():
    konch.use_file(".konchrc.local", trust=True)
```

Note: The context in `.konchrc.local` will be merged with the context in `.konchrc`.

Note: Passing `trust=True` allows `.konchrc.local` to be edited without requiring approval. **This is safe if (and only if) `.konchrc.local` is not added to source control.**

Note: `setup()` and `teardown()` cannot be used in `.konchrc.local`.

PROGRAMMATIC USAGE

Want to use konch within a Python script? konch exposes many of its high-level functions.

```
import konch
from mypackage import cheese

# Start the shell
konch.start(context={"cheese": cheese}, shell=konch.AutoShell)
```

To use a config file:

```
import konch

konch.use_file("~/path/to/.mykonchrc")
konch.start()
```

Get command-line arguments using `konch.parse_args()`. konch uses `docopt` for arguments parsing.

```
import konch
from myapp import app, db

args = konch.parse_args()
if args["--name"] == "db":
    # ...expensive database setup...
    konch.start(context={"db": db, "app": app})
else:
    konch.start(context={"app": app})
```

You can also use shell objects directly:

```
import konch

my_shell = konch.AutoShell(context={"foo": 42}, banner="My foo shell")
my_shell.start()
```

“Praise the Magic Conch!”

PROJECT INFO

10.1 License

Copyright 2014-2024 Steven Loria

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

10.2 Changelog

10.2.1 4.5.0 (2024-01-10)

Features:

- Install *konch* within a *konch* package. This prevents *docopt.py* from polluting the site packages directory.
- Add *py.typed* marker file for PEP 561 compatibility.

Other changes:

- Drop support for Python 3.7 (EOL).
- Test against Python 3.11 and Python 3.12.

10.2.2 4.4.0 (2022-01-12)

Bug fixes:

- Fix compatibility with IPython 8.0 (#211).

Other changes:

- Drop support for Python 3.6 (EOL).
- Test against Python 3.10.

10.2.3 4.3.0 (2020-11-29)

Features:

- Follow XDG standard in config file discovery (#177). Thanks @rpdelaney for PR and the suggestion.

Other changes:

- Test against Python 3.8 and 3.9.

10.2.4 4.2.2 (2019-06-19)

Bug fixes:

- Remove usage of deprecated `imp` module.
- Handle `AttributeError` when object in context list has no `__name__` attribute (#105). Thanks @brl0 for reporting.

10.2.5 4.2.1 (2019-03-16)

Bug fixes:

- Prevent error when ptpython config file exists (#84). Thanks @svanburen.

10.2.6 4.2.0 (2019-03-12)

Features:

- Add BPython Curses shell (#78). Thanks @goandbuild for the suggestion and PR.

10.2.7 4.1.1 (2019-02-28)

Bug fixes:

- Fix casing when using `"context_format": "short"`.

10.2.8 4.1.0 (2019-01-24)

Features:

- Use a [y/N] prompt for authorizing a config file instead of requiring user to run `konch allow`.

10.2.9 4.0.1 (2019-01-19)

Bug fixes:

- Fix behavior when `konch edit` is passed a file that does not exist.

Other changes:

- Various documentation improvements and updates.

10.2.10 4.0.0 (2019-01-19)

Features:

- Add ANSI coloring and improve messaging (#67).
- `konch.config()` will shallow-merge `context` when called multiple times.
- `konch edit` may be passed a filename to edit.
- `konch.main` accepts an `argv` argument.
- Add `trust` parameter to `konch.use_file()`.

Other changes:

- Python 2 is no longer supported. Python>=3.6 is officially supported.

10.2.11 3.2.1 (2019-01-17)

Bug fixes:

- Fix handling of nested modules when passing a list to `context`.

10.2.12 3.2.0.post0 (2019-01-13)

- Add `python_requires` to PyPI metadata.

3.2.x will be the last release line to support Python 2.

10.2.13 3.2.0 (2019-01-11)

- Show preview when `.konchrc` file has changed.

10.2.14 3.1.0 (2018-12-29)

- Show preview of unauthorized files.

10.2.15 3.0.0.post0 (2018-12-22)

- Distribute a universal wheel.

10.2.16 3.0.0 (2018-12-22)

Features:

- Config files must be approved before executing them. Use `konch allow` to authorize a config file. This is a security mechanism to prevent executing untrusted Python code (#47). Thanks [@hartwork](#) for the suggestion.
- Allow customizing the editor to use for `konch edit` via the `KONCH_EDITOR` environment variable.
- `konch init` only adds the encoding pragma (`# -*- coding: utf-8 -*-\n`) on Python 2.
- Raise error when an invalid `--name` is passed.

Bug fixes:

- Respect ptpython's user config file (`~/.ptpython/config.py`) (#16). Thanks [@nasyxx](#) for reporting and thanks [@pycadelic](#) for helping out with the implementation.
- Address a `DeprecationWarning` about importing from `collections.abc` on Python 3.7.

10.2.17 2.5.0 (2018-11-04)

- Update dev environment.
- Python 3.4 is no longer officially supported.
- Tested on Python 3.7.

10.2.18 2.4.0 (2017-04-29)

Features:

- Add basic tab-completion to plain Python shell.

10.2.19 2.3.0 (2016-12-23)

Features:

- Allow `context` to be a callable.
- Multiple names may be passed to `named_config`.

10.2.20 2.2.1 (2016-12-19)

Bug fixes:

- Fix error raised when some options are passed to `konch.named_config`.

10.2.21 2.2.0 (2016-07-21)

Features:

- Add `ipy_colors` and `ipy_highlighting_style` options for customizing IPython terminal colors.

10.2.22 2.1.0 (2016-07-18)

Features:

- Compatibility with IPython>=5.0.0.

Support:

- Update `tasks.py` for compatibility with `invoke`>=0.13.0.

10.2.23 2.0.0 (2016-06-01)

Features:

- Customizable context formatting via the `context_format` option.
- More CONCHES!

Deprecations/Removals:

- Remove `hide_context` option. Use the `context_format` option instead.
- Drop support for Python<=2.6 and <=3.3.

Bug fixes:

- Fix bug in checking availability of `PtlPython`.
- Fix bug in passing shell subclass as `shell` argument to `konch.start`.

10.2.24 1.1.2 (2016-05-24)

- `ShellNotAvailableErrors` no longer pollute tracebacks when using the `AutoShell`.

10.2.25 1.1.1 (2015-09-27)

- Remove deprecated import of `IPython.config`.

10.2.26 1.1.0 (2015-06-21)

- Add `ptpython` support.

10.2.27 1.0.0 (2015-02-08)

- Add support for `setup` and `teardown` functions in `.konchrc` files.
- If `~/ .konchrc.default` exists, use that file as the template for new `.konchrc` files created with `konch init`.
- Add `ipy_extensions` and `ipy_autoreload` options.
- Make sure that `vim` opens `.konchrc` files in Python mode.
- Drop Python 3.2 support.

10.2.28 0.4.2 (2014-07-12)

- “shell” option in `.konchrc` can be a string: either `'bpy'`, `'ipy'`, `'py'`, or `'auto'`.
- Fix error in “konch edit”.

10.2.29 0.4.1 (2014-06-23)

- Fix bug that caused `konch` to hang if no `.konchrc` file can be found.

10.2.30 0.4.0 (2014-06-10)

- Add `edit` command for editing `.konchrc` file.
- Properly output error messages to `stderr`.
- Tested on Python 3.4.

10.2.31 0.3.4 (2014-04-06)

- Fix bug that raised `SyntaxError` when executing `konch` on Windows.

10.2.32 0.3.3 (2014-03-27)

- Fix bug in `resolve_path` that caused infinite loop if config file not found.
- Fix bug with initializing `konch` in home directory.
- Add `hide_context` option.

10.2.33 0.3.2 (2014-03-18)

- Some changes to make it easier to use konch programatically.
- `konch.start()` can be called with no arguments.
- Expose docopt argument parsing via `konch.parse_args()`.

10.2.34 0.3.1 (2014-03-17)

- Doesn't change current working directory.
- Less magicks.
- Tested on Python 3.4.

10.2.35 0.3.0 (2014-03-16)

- Smarter path resolution. konch will search parent directories until it finds a `.konchrc` file to use.
- Make prompt configurable on IPython and built-in shell. Output template is also supported on IPython.
- *Backwards-incompatible*: Remove support for old ($\leq 0.10.x$ —released 3 years ago!) versions of IPython.

10.2.36 0.2.0 (2014-03-15)

- Fix bug with importing modules and packages in the current working directory.
- Introducing *named configs*.

10.2.37 0.1.0 (2014-03-14)

- First release to PyPI.