

---

# Bugjar Documentation

*Release 0.1.0*

**Russell Keith-Magee**

**Sep 27, 2017**



---

## Contents

---

<b>1</b>	<b>Bugjar</b>	<b>1</b>
1.1	Quickstart . . . . .	1
1.2	Documentation . . . . .	2
1.3	Community . . . . .	2
<b>2</b>	<b>Indices and tables</b>	<b>5</b>



# CHAPTER 1

---

## Bugjar

---

Bugjar is part of the [BeeWare suite](http://pybee.org/bugjar). The project website is <http://pybee.org/bugjar>.

Anyone who learned to code in the mid to late 80s probably spent some time with a Borland compiler – probably either Turbo Pascal or Turbo C. One of the best features of the Turbo compilers was their IDE – and in particular, a really good visual debugger that would let you inspect code while it was running.

Then we all moved to Unix, and somehow forgot what a good debugger was. GDB is perfectly functional, but isn't very intuitive. GDB gives you perfect control over the execution of your code, but bad contextual information to let you know what control you should be exercising.

Then came Python. Python's execution model contains excellent debugging hooks, and supplies PDB as a proof of concept. PDB is an interface that shares many similarities with GDB – text mode, fantastic control, but very bad contextual information.

So - enter `bugjar`. A graphical interface for debugging code. PDB, but with the context to help you step through code in a meaningful way.

## Quickstart

Bugjar can be installed with pip:

```
$ pip install bugjar
```

You can then debug a Python script by typing the following at a shell prompt:

```
$ bugjar myscript.py arg1 arg2
```

This will start a graphical interface, with `myscript.py` loaded into the source code window. You can set (or remove) breakpoints by clicking on line numbers; you can step through and into code; or you can set the program running unconstrained. Each time the debugger stops at a breakpoint, the inspector will be updated with the current contents of locals, globals, and builtins.

The Python script will run using your current environment; if you have an active `virtualenv`, that environment will be current.

When you quit the debugger, the script will be terminated.

## Documentation

Documentation for bugjar can be found on [Read The Docs](#).

## Community

Bugjar is part of the [BeeWare suite](#). You can talk to the community through:

- [@pybeeware](#) on Twitter
- The [BeeWare Users Mailing list](#), for questions about how to use the BeeWare suite.
- The [BeeWare Developers Mailing list](#), for discussing the development of new features in the BeeWare suite, and ideas for new tools for the suite.

Contents:

## Headless mode

Bugjar can also operate in a headless mode. This can be use to debug processes running on a remote machine (although it also works on local machines).

In headless mode, Bugjar is split into two parts:

- The **Net**: a headless backend responsible for debugging code
- The **Jar**: the GUI used to inspect code.

To debug in headless mode, you first start a headless debugger (the net) on the process that you want to debug:

```
$ bugjar-net myscript.py arg1 arg2
```

Then, on the machine that you want to visualize the debugging session, you start the user interface (the jar), and attach it to the net:

```
$ bugjar-jar --host example.com
```

If the net and the jar are running on the same machine, the `--host example.com` argument can be omitted.

Unlike local mode, when you quit the debugger, the script will *not* be terminated by closing the jar. If you close the jar, and reopen a new session, the GUI will resume where it left off. The net is responsible for running the script; when the net is stopped, the script will be terminated.

## Contributing to Bugjar

If you experience problems with bugjar, [log them on GitHub](#). If you want to contribute code, please [fork the code](#) and [submit a pull request](#).

## Setting up your development environment

The recommended way of setting up your development environment for bugjar is to install a virtual environment, install the required dependencies and start coding. Assuming that you are using `virtualenvwrapper`, you only have to run:

```
$ git clone git@github.com:pybee/bugjar.git
$ cd bugjar
$ mkvirtualenv bugjar
```

bugjar uses unittest (or unittest2 for Python < 2.7) for its own test suite as well as additional helper modules for testing. To install all the requirements for bugjar, you have to run the following commands within your virtual environment:

```
$ pip install -e .
$ pip install -r requirements_dev.txt
```

In case you are running a python version < 2.7 please use the `requirements_dev_python2.7.txt` instead because unittest2 is not part of the standard library for these version.

Now you are ready to start hacking! Have fun!

## Bugjar Roadmap

Bugjar is a new project - we have lots of things that we'd like to do. If you'd like to contribute, providing a patch for one of these features:

- Port to Python 3
- Add the ability to browse objects in the stack
- Add the ability to add user-defined expressions to watch.

## Release History

### 0.1.0 - August 31, 2013

Initial public release for DjangoCon US 2013.



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`