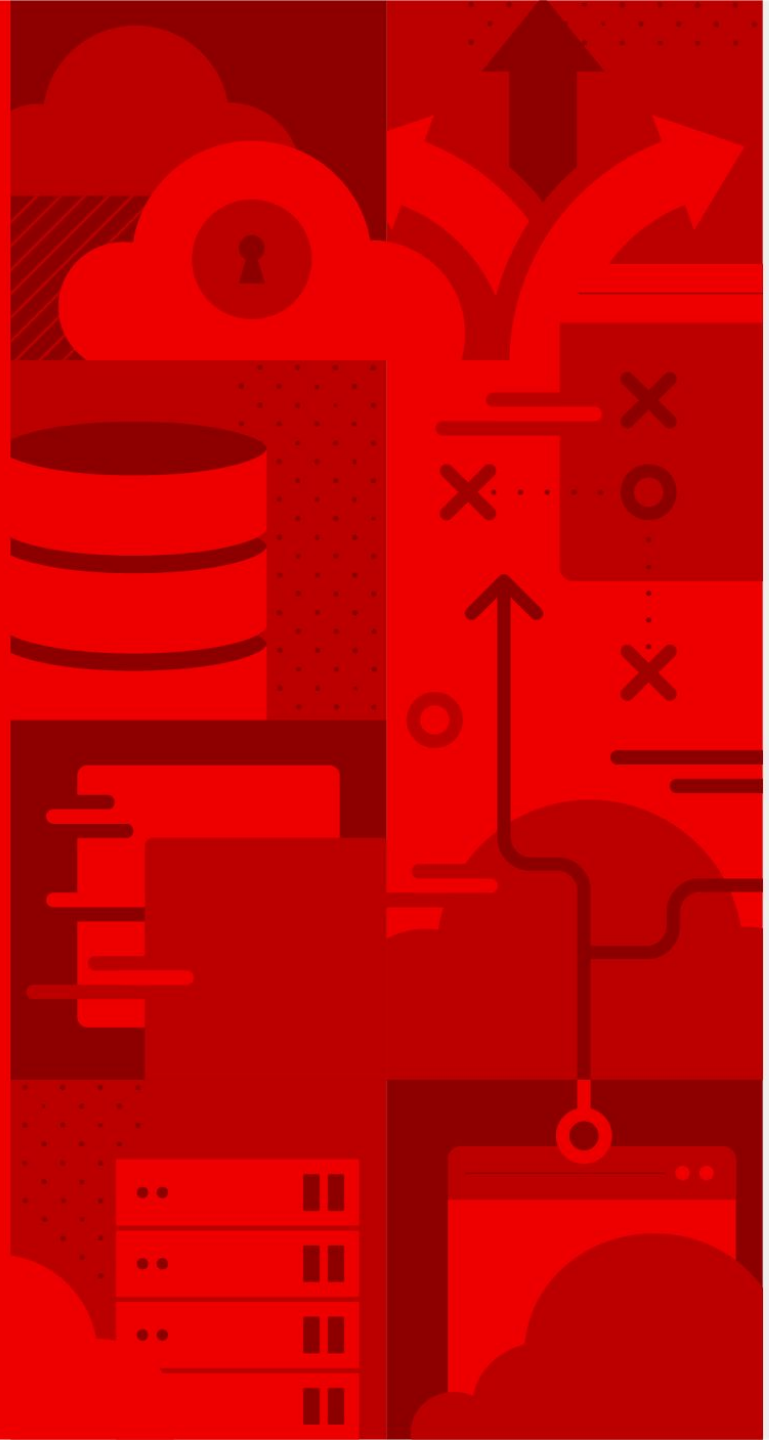


Testing and debugging Python Flask applications

Giulia Naponiello
Software Engineer

Red Hat
PnT DevOps Team

Who am I?

Software engineer at Red Hat

Python developer

In love with open source and Red Hat

Working with Gating and Containers

Red Hat
PnT DevOps Team

Today's agenda

Flask: python web application framework

- What it is
- Flask demo

Testing

- What it is
- Why it's important
- Unit tests demo

Debugging

- How to properly debug
- pdb (The Python Debugger)
- gdb (GNU Debugger)
- More demos!

Web application framework

Easy and quick to get start

Supports extensions to add functionalities

Live demo...

What can go wrong?

Testing

Why do I need testing?

You might think...

Why do I need testing?

You might think...

I don't have time.

Why do I need testing?

You might think...

I don't have time.
It's faster to manually test my code.

Why do I need testing?

You might think...

I don't have time.
It's faster to manually test my code.
I don't know how to write tests.

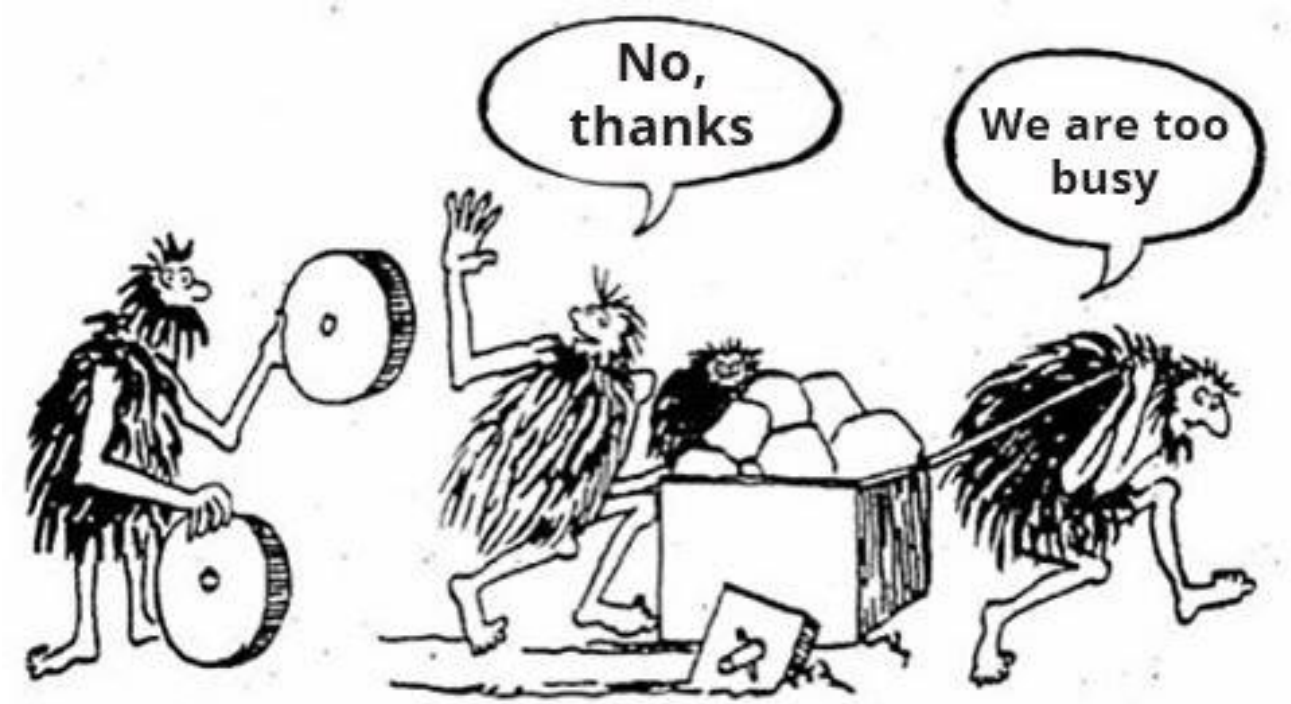
Why do I need testing?

You might think...

I don't have time.
It's faster to manually test my code.
I don't know how to write tests.

...let's just test in production!

Why do I need testing?



Why do I need testing?

Tests will:

- Save your time.

Why do I need testing?

Tests will:

- Save your time.
- Help you preventing errors.

Why do I need testing?

Tests will:

- Save your time.
- Help you preventing errors.
- Help new developers to start contributing.

Why do I need testing?

Tests will:

- Save your time.
- Help you preventing errors.
- Help new developers to start contributing.
- Make your code more elegant.

> Code without tests is **broken** by design.

Jacob Kaplan-Moss
Co-creator of Django

Live demo...

...again.

Debugging

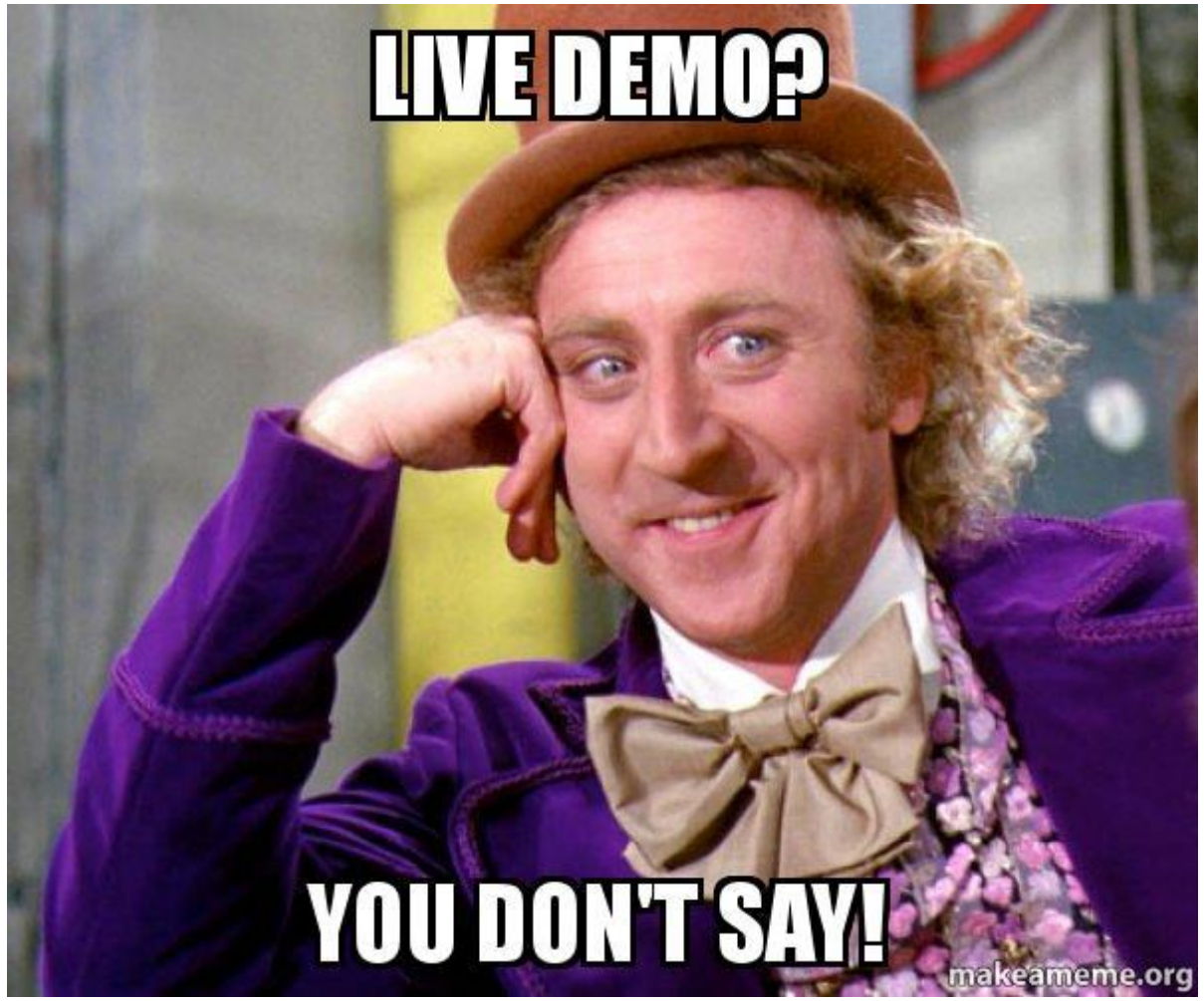
pdb - The Python Debugger

- Interactive source code debugger for Python programs.
- Supports breakpoints: `import pdb; pdb.set_trace()`
- Supports:
  - Stepping at the source line level.
  - Inspection of stack frames.
  - Source code listing.
  - Evaluation of arbitrary Python code.

pdb - The Python Debugger

(my favourite) commands:

- `n(ext)` → executes until the next line in the current function is reached or it returns
- `s(tep)` → executes the current line, stop at the first possible occasion
- `l(ist)` → lists source code for the current file
- `c(ont(inue))` → continue execution, only stop when a breakpoint is encountered.
- `locals()` and `globals()` will display all the variables in scope with their values (actually this is not pdb… but it's so useful).


LIVE DEMO?
YOU DON'T SAY!
makeameme.org

gdb - GNU Debugger

Some bugs that are difficult to debug:

- segfaults
- hung processes
- out of control daemon processes

Fake demo this time...

That's our program "broken.py" this time.

```
import time

def wasting_time():
    a = "a variable"
    while(1):
        time.sleep(1)

wasting_time()
```


Let's run the script:

```
python3 broken.py
```

Get the pid of it:

```
~/proj/demo-py-meetup (master)$ 🦄 ps -aux | grep broken
gnaponie  203270  0.0  0.0 227840 10464 pts/8   S+   18:02   0:00 vim broken.py
gnaponie  204292  0.1  0.0 223364  8788 pts/9   S+   18:46   0:00 python3 broken.py
gnaponie  205547  0.0  0.0 216120   848 pts/7   S+   18:46   0:00 grep --color=auto broken
```

Get the gdb dump for it:

```
gcore 204292
```

Start gdb on it:

```
gdb python3 core.204292
```

Analyze the backtrace with `bt`:

```
bt
```


```
(gdb) bt
#0  0x00007fa2a4c94f5a in select () from /lib64/libc.so.6
#1  0x00007fa2a4a1643d in time_sleep () from /lib64/libpython3.7m.so.1.0
#2  0x00007fa2a495db13 in _PyMethodDef_RawFastCallKeywords () from /lib64/libpython3.7m.so.1.0
#3  0x00007fa2a495dd43 in _PyCFunction_FastCallKeywords () from /lib64/libpython3.7m.so.1.0
#4  0x00007fa2a4990203 in call_function () from /lib64/libpython3.7m.so.1.0
#5  0x00007fa2a49cae2d in _PyEval_EvalFrameDefault () from /lib64/libpython3.7m.so.1.0
#6  0x00007fa2a497e2d2 in _PyFunction_FastCallKeywords () from /lib64/libpython3.7m.so.1.0
#7  0x00007fa2a49900ef in call_function () from /lib64/libpython3.7m.so.1.0
#8  0x00007fa2a49c64ce in _PyEval_EvalFrameDefault () from /lib64/libpython3.7m.so.1.0
#9  0x00007fa2a497d430 in _PyEval_EvalCodeWithName () from /lib64/libpython3.7m.so.1.0
#10 0x00007fa2a497e1c9 in PyEval_EvalCodeEx () from /lib64/libpython3.7m.so.1.0
#11 0x00007fa2a4a0e0db in PyEval_EvalCode () from /lib64/libpython3.7m.so.1.0
#12 0x00007fa2a4a4fc43 in run_mod () from /lib64/libpython3.7m.so.1.0
#13 0x00007fa2a4a50197 in PyRun_FileExFlags () from /lib64/libpython3.7m.so.1.0
#14 0x00007fa2a4a5668a in PyRun_SimpleFileExFlags () from /lib64/libpython3.7m.so.1.0
#15 0x00007fa2a4a58411 in pymain_main () from /lib64/libpython3.7m.so.1.0
#16 0x00007fa2a4a585bc in _Py_UnixMain () from /lib64/libpython3.7m.so.1.0
#17 0x00007fa2a4bc31a3 in __libc_start_main () from /lib64/libc.so.6
#18 0x000055a8f9ee708e in _start ()
```


```
(gdb) bt
#0  0x00007fa2a4c94f5a in select () from /lib64/libc.so.6
```

Current instruction pointer.

Call to the kernel, waiting on the sleep…

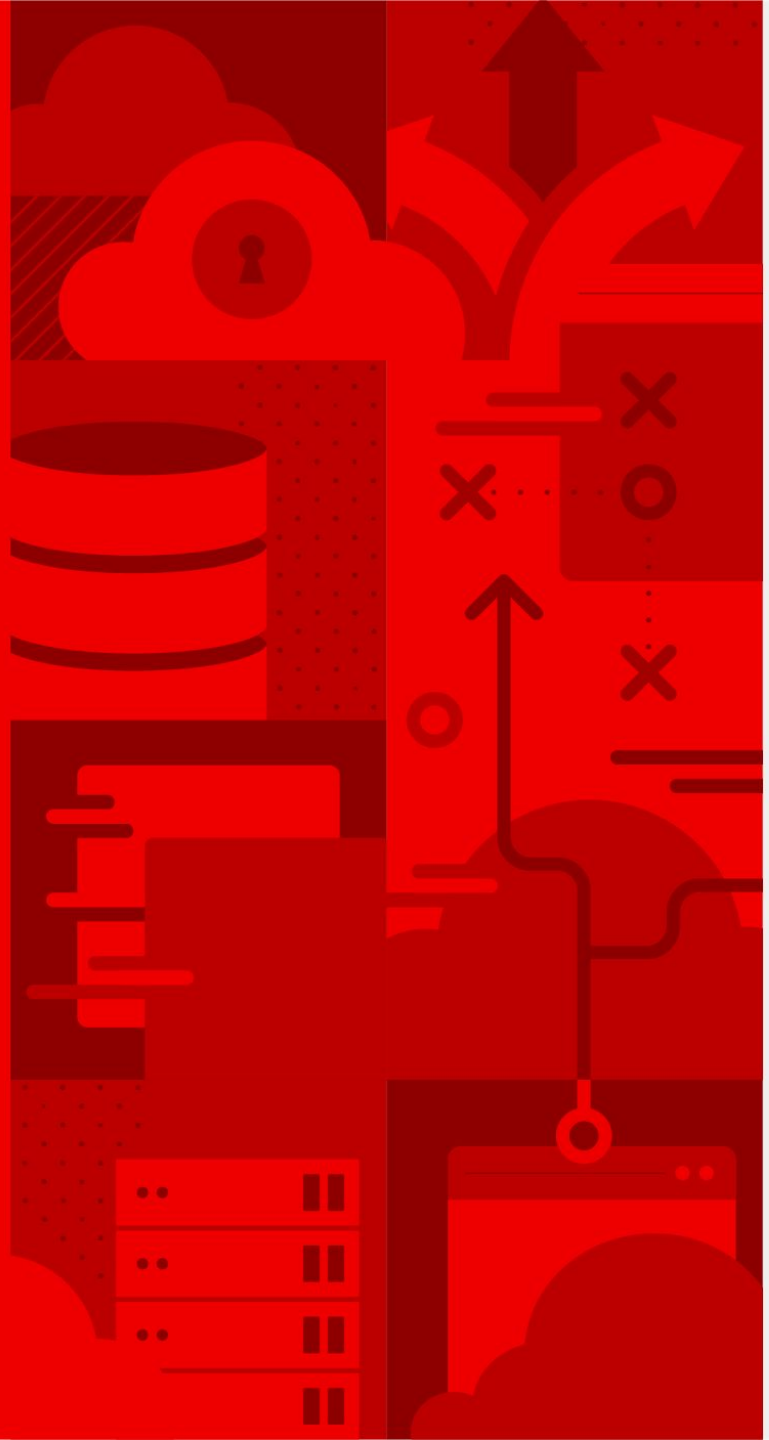
Select man page:

select() and pselect() allow a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready" for some class of I/O operation (e.g., input possible).

…you can do a lot of more fun stuff with these tool, but that's all we got for today.

We are hiring!

Red Hat
PnT DevOps Team



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat
PnT DevOps Team