Debugging Tips

http://bit.ly/pdbtips

Roy Hyunjin Han

@crosscompute

PyCon Asia Pacific 2016 Seoul, South Korea

Debugging scenarios

- 1. Prototype
- 2. Development
- 3. Production

Examine exception while running interpreter

Pinpoint variables in scope

Step through execution at breakpoint

Examine exception while running interpreter

jupyter notebook debug

Pinpoint variables in scope

Step through execution at breakpoint

Examine exception while running interpreter jupyter notebook debug

Pinpoint variables in scope import IPython; IPython.embed() whos

Step through execution at breakpoint

```
import sys
def f(x, y):
    import IPython; IPython.embed()
    return x + y
a, b = sys.argv[1:]
print(f(int(a), int(b)))
Use
         In [1]: whos
         Variable Type
whos
                    module
                              <module
         IPython
for
                    int
                    int
scope
```

Examine exception while running interpreter jupyter notebook debug

Pinpoint variables in scope import IPython; IPython.embed()

Step through execution at breakpoint import pudb; pudb.set_trace()

```
PuDB 2016.2 - ?:help n:next
                             s:step into b:breakpoin
 import sys
                                   Variables:
 def f(x, y):
 import pudb; pudb.set trace()
 a, b = sys.argv[1:]
                                   Stack:
 print(f(int(a), int(b)))
                                   >> f breakpoint.py
                                      <module> breakp
>>> X
             Step through
>>> f. module
                                   Breakpoints:
                  execution
  main
>>> len(sys.argv)
                        < Clear
```

Drop into debugger without modifying code

Trace on CTRL-C

Debug threads or Docker containers

Drop into debugger without modifying code pudb -- xyz.py arg1 arg2

Trace on CTRL-C

Debug threads or Docker containers

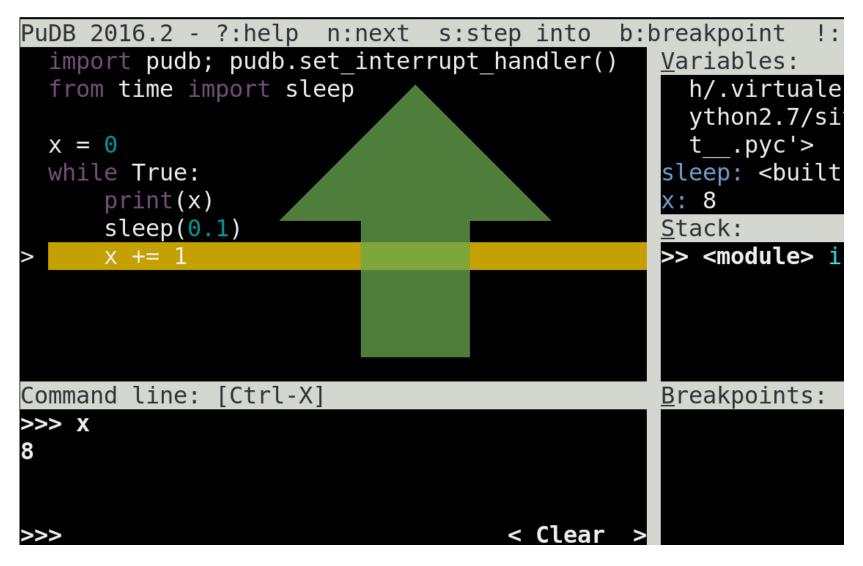
```
PuDB 2016.2 - ?:help n:next s:step into
                                            b:breakpoint
                                                          !:python
  import sys
                                              Variables:
                                              x: 1
  def f(x, y):
      return x + y
                                              Stack:
                                              >> f run.py:4
  a, b = sys.argv[1:]
  print(f(int(a), int(b)))
                                                 <module> run.py:7
Command line: [Ctrl-X]
>>> sys.argv
                                              Breakpoints:
['run.py', '1', '2']
                                   < Clear
```

Debug without modifying code pudb -- xyz.py arg1 arg2

Drop into debugger without modifying code pudb -- xyz.py arg1 arg2

Trace on CTRL-C import pudb; pudb.set_interrupt_handler()

Debug threads or Docker containers

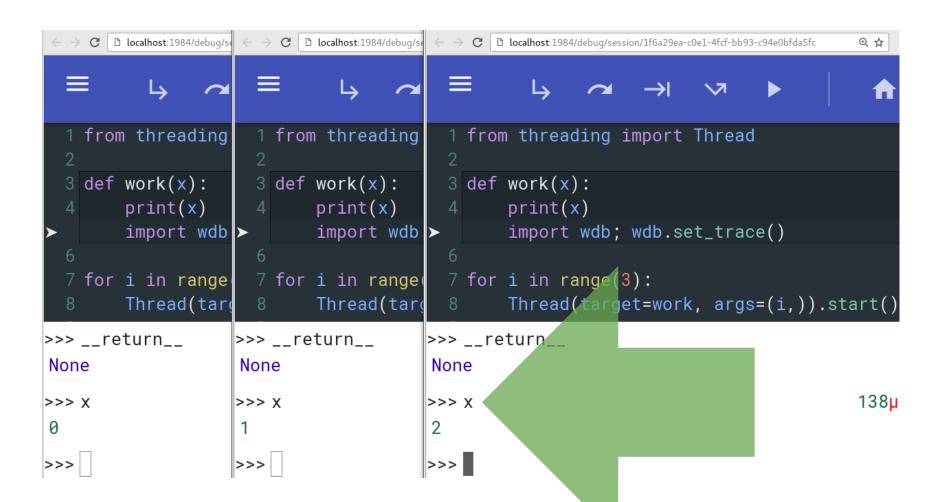


Trace on CTRL-C

Drop into debugger without modifying code pudb -- xyz.py arg1 arg2

Trace on CTRL-C import pudb; pudb.set_interrupt_handler()

Debug threads or Docker containers import wdb; wdb.set_trace()



Use wdb to debug threads

```
script:
  build:
 links:
  - wdb
  environment:
   WDB SOCKET SERVER: wdb
    WDB NO BROWSER AUTO OPEN: "True"
wdb:
 image: kozea/wdb-server
  ports:
   - "1984:1984"
```

Use wdb for Docker containers

Production debugging

Use namespaced logger with rsyslog server

Debug tests

Production debugging

Use namespaced logger with rsyslog server

logging.getLogger(__name__)

Debug tests

```
import logging
logger.addHandler(logging.NullHandler())
def f():
   logger.debug('a')
   logger.info('b')
   logger.warning('c')
   logger.error('d')
   logger.critical('e')
NORMAL >> <xyz.py     pyt... < crosscompute <<</pre>
                                           27%
import xyz
import logging
logging.getLogger('xyz').setLevel(logging.DEBUG)
logging.basicConfig()
xyz.f()
```

Use namespaced logger

Production debugging

```
Use namespaced logger with rsyslog server logging.getLogger(__name__)
```

Debug tests

pip install pdbpp; py.test --pdb

```
platform linux2 -- Python 2.7.12, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /home/rhh/Projects/crosscompute, inifile:
plugins: cov-2.2.1
collected 16 items
examples/test examples.py .F
tmpdir = local('/tmp/pytest-of-rhh/pytest-2/test find primes0')
  def test find primes(tmpdir):
     args = str(tmpdir), 'find-primes', {'x_integer': 2016}
     r = run(*args)
     assert r['standard_outputs']['unique_factor_count'] == 3
     assert '3' == 3
examples/test examples.py:16: AssertionError
[21] > /home/rhh/Projects/crosscompute/examples/test_examples.py(16)test find p
rimes()
-> assert r['standard_outputs']['unique factor count'] == 3
(Pdb++) r.keys()
```

Debug tests

pip install pdbpp; py.test --pdb

bit.ly/pdbtips

Examine exception from interpreter	jupyter notebook debug
Pinpoint variables in scope	import IPython; IPython.embed() whos
Set breakpoint	import pudb; pudb.set_trace()
Run with debugger	pudb xyz.py arg1 arg2
Start debugger on CTRL-C	import pudb pudb.set_interrupt_handler()
Trace threads	import wdb; wdb.set_trace()
Record logs	logging.getLogger(name)
Debug tests	pip install pdbpp; py.testpdb