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

Prototype debugging

Examine exception while running interpreter

Pinpoint variables in scope

Step through execution at breakpoint

Prototype debugging

Examine exception while running interpreter

jupyter notebook
debug

Pinpoint variables in scope

Step through execution at breakpoint

```
In [1]: def f(x):
            return 100 / x

In [2]: f(0)

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-421db934c3f8> in <module>()
----> 1 f(0)

<ipython-input-1-87ea133d1dd7> in f(x)
      1 def f(x):
----> 2     return 100 / x

ZeroDivisionError: integer division or modulo by zero

In [3]: debug
> <ipython-input-1-87ea133d1dd7>(2)f()
      1 def f(x):
----> 2     return 100 / x

ipdb> x + 1
1
ipdb> q
```

Examine exception while running interpreter

Prototype debugging

Examine exception while running interpreter

jupyter notebook

debug

Pinpoint variables in scope

import IPython; IPython.embed()

whos

Step through execution at breakpoint

```python
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 whos for scope

```
In [1]: whos
Variable   Type       Data/Info
-----------------------------------
IPython    module     <module 'I
x          int        1
y          int        2
```

Prototype debugging

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

  def f(x, y):
>     return x + y

 *import pudb; pudb.set_trace()
  a, b = sys.argv[1:]
  print(f(int(a), int(b)))

Command line: [Ctrl-X]
>>> x
1
>>> f.__module__
'__main__'
>>> len(sys.argv)
3
>>>                          < Clear  >
```

```
Variables:
x: 1
y: 2

Stack:
>> f breakpoint.py
   <module> breakp

Breakpoints:
```

Step through execution

Development debugging

Drop into debugger without modifying code

Trace on CTRL-C

Debug threads or Docker containers

Development debugging

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):                                y: 2
>     return x + y
                                              Stack:
  a, b = sys.argv[1:]                         >> f run.py:4
  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

Development debugging

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

```
PuDB 2016.2 - ?:help  n:next  s:step into  b:breakpoint  !:
  import pudb; pudb.set_interrupt_handler()
  from time import sleep

  x = 0
  while True:
      print(x)
      sleep(0.1)
>     x += 1
Command line: [Ctrl-X]
>>> x
8

>>>                                          < Clear  >
```

```
Variables:
  h/.virtuale
  ython2.7/si
  t__.pyc'>
sleep: <built
x: 8
Stack:
>> <module> i
Breakpoints:
```

Trace on CTRL-C

Development debugging

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()
```

```
1 from threading
2
3 def work(x):
4     print(x)
➤     import wdb
6
7 for i in range
8     Thread(tar
```

```
>>> __return__
None
>>> x
0
>>>
```

```
1 from threading
2
3 def work(x):
4     print(x)
➤     import wdb
6
7 for i in range
8     Thread(tar
```

```
>>> __return__
None
>>> x
1
>>>
```

localhost:1984/debug/session/1f6a29ea-c0e1-4fcf-bb93-c94e0bfda5fc

```
1 from threading import Thread
2
3 def work(x):
4     print(x)
➤     import wdb; wdb.set_trace()
6
7 for i in range(3):
8     Thread(target=work, args=(i,)).start()
```

```
>>> __return__
None
>>> x
2
>>>
```

138µ

Use wdb to debug threads

```yaml
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

```python
import logging

logger = logging.getLogger(__name__)
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 <<  27%
import xyz

import logging
logging.getLogger('xyz').setLevel(logging.DEBUG)
logging.basicConfig()

xyz.f()
 <ugging/7/logs.py    pyt… < crosscompute <<  57%
```

Use namespaced logger

Production debugging

Use namespaced logger with rsyslog server

logging.getLogger(__name__)

Debug tests

pip install pdbpp; py.test --pdb

```
============================= test session starts ==============================
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
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> traceback >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

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
E       assert '3' == 3

examples/test_examples.py:16: AssertionError
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> entering PDB >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
[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.test --pdb