Python Debugging Fundamentals

Saturday, October 20, 2012

PyCarolinas

UNC School of Pharmacy

Chapel Hill, NC

Chris Calloway

University of North Carolina

Department of Marine Sciences

What Is Debugging

A method for isolating program errors

What Is Debugging

- Execute program one statement at a time
- Inspect the state of objects bound to identifiers
- Lather, rinse, repeat

Python's pdb module to the rescue!

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py
```

Invoking Python's Debugger

Run a module as a script

```
$ python -m pdb fizzbuzz.py
```

Invoking Python's Debugger

Module to run
as a script

```
$ python -m pdb fizzbuzz.py
```

Invoking Python's Debugger

Argument to pdb:
script to debug

```
$ python -m pdb fizzbuzz.py
```

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Full path to script being debugged

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Line number of next statement to execute

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Type of object last evaluated

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Instruction pointer to next statement to execute

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Debugger prompt

Python's Debugger Prompt

```
(Pdb) help

Documented commands (type help <topic>):
========================================
EOF    cl         disable  interact  next     return  u          where
a      clear      display  j         p        retval  unalias
alias  commands   down     jump      pp       run     undisplay
args   condition  enable   l         print    rv      unt
b      cont       exit     list      q        s       until
break  continue   h        ll        quit     source  up
bt     d          help     longlist  r        step    w
c      debug      ignore   n         restart  tbreak  whatis

Miscellaneous help topics:
==========================
exec  pdb

(Pdb)
```

Python's Debugger Prompt

```
(Pdb) help list
l(ist) [first [,last] | .]

        List source code for the current file.  Without arguments,
        list 11 lines around the current line or continue the previous
        listing.  With . as argument, list 11 lines around the current
        line.  With one argument, list 11 lines starting at that line.
        With two arguments, list the given range; if the second
        argument is less than the first, it is a count.

        The current line in the current frame is indicated by "->".
        If an exception is being debugged, the line where the
        exception was originally raised or propagated is indicated by
        ">>", if it differs from the current line.
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) h l
l(ist) [first [,last] | .]

        List source code for the current file.  Without arguments,
        list 11 lines around the current line or continue the previous
        listing.  With . as argument, list 11 lines around the current
        line.  With one argument, list 11 lines starting at that line.
        With two arguments, list the given range; if the second
        argument is less than the first, it is a count.

        The current line in the current frame is indicated by "->".
        If an exception is being debugged, the line where the
        exception was originally raised or propagated is indicated by
        ">>", if it differs from the current line.
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) l
  2     Generate the first n Fizz Buzz answers.
  3
  4     Usage:
  5
  6     > python fizzbuzz.py n
  7  -> """
  8
  9     import sys
 10
 11     def fizzbuzz(n):
 12         """
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) l 6
  1       """
  2       Generate the first n Fizz Buzz answers.
  3
  4       Usage:
  5
  6       > python fizzbuzz.py n
  7    -> """
  8
  9       import sys
 10
 11       def fizzbuzz(n):
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) l .
  2     Generate the first n Fizz Buzz answers.
  3
  4     Usage:
  5
  6     > python fizzbuzz.py n
  7  -> """
  8
  9     import sys
 10
 11     def fizzbuzz(n):
 12         """
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) 3 ** (1 / 2)
1.7320508075688772
(Pdb) dir()
['__builtins__', '__file__', '__name__']
(Pdb) print(__name__)
'__main__'
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) !list
<class 'list'>
(Pdb)
```

What Is Debugging

- Execute program one statement at a time
- Inspect the state of objects bound to identifiers
- Lather, rinse, repeat

Single Stepping

```
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(9)<module>()
-> import sys
(Pdb)
```

Single Stepping

```
(Pdb) l
  4     Usage:
  5
  6     > python fizzbuzz.py n
  7     """
  8
  9  -> import sys
 10
 11     def fizzbuzz(n):
 12         """
 13         fizzbuzz(n) -> [first n Fizz Buzz answers]
 14         """
(Pdb)
```

Single Stepping

```
(Pdb) dir()
['__builtins__', '__doc__', '__file__', '__name__']
(Pdb) !print(__doc__)

Generate the first n Fizz Buzz answers.

Usage:

> python fizzbuzz.py n

(Pdb)
```

Single Stepping

```
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(11)<module>()
-> def fizzbuzz(n):
(Pdb) dir()
['__builtins__', '__doc__', '__file__', '__name__', 'sys']
(Pdb)
```

Single Stepping

```
(Pdb) l
  6     > python fizzbuzz.py n
  7     """
  8
  9     import sys
 10
 11  -> def fizzbuzz(n):
 12         """
 13         fizzbuzz(n) -> [first n Fizz Buzz answers]
 14         """
 15
 16         answers = []
(Pdb)
```

Single Stepping

```
(Pdb) l
 17             for x in range(1,n+1):
 18                 answer = ""
 19                 if not x%3:
 20                     answer += "Fizz"
 21                 if not x%5:
 22                     answer += "Buzz"
 23                 if not answer:
 24                     answer = x
 25                 answers.append(answer)
 26             return answers
 27
(Pdb)
```

Single Stepping

```
(Pdb) l .
  6         > python fizzbuzz.py n
  7         """
  8
  9         import sys
 10
 11   -> def fizzbuzz(n):
 12             """
 13             fizzbuzz(n) -> [first n Fizz Buzz answers]
 14             """
 15
 16             answers = []
(Pdb)
```

Single Stepping

```
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(28)<module>()
-> if __name__ == '__main__':
(Pdb) dir()
['__builtins__', '__doc__', '__file__',
 '__name__', 'fizzbuzz', 'sys']
(Pdb)
```

Single Stepping

```
(Pdb) l
 23                 if not answer:
 24                     answer = x
 25                 answers.append(answer)
 26             return answers
 27
 28  -> if __name__ == '__main__':
 29             try:
 30                 if len(sys.argv) != 2:
 31                     raise ValueError("Incorrect number of arguments")
 32                 answers = fizzbuzz(int(sys.argv[1]))
 33                 print(" ".join([str(answer) for answer in answers]))
(Pdb)
```

Single Stepping

```
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(29)<module>()
-> try:
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(30)<module>()
-> if len(sys.argv) != 2:
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(31)<module>()
-> raise ValueError("Incorrect number of arguments")
(Pdb) s
ValueError: Incorrect number of arguments
> /Users/cbc/pycarolinas/fizzbuzz.py(31)<module>()
-> raise ValueError("Incorrect number of arguments")
(Pdb)
```

Single Stepping

```
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(34)<module>()
-> except:
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(35)<module>()
-> print(__doc__)
(Pdb)
```

SINGLE STEPPING

```
(Pdb) s

Generate the first n Fizz Buzz answers.

Usage:

> python fizzbuzz.py n

--Return--
> /Users/cbc/pycarolinas/fizzbuzz.py(35)<module>()->None
-> print(__doc__)
(Pdb)
```

Single Stepping

```
(Pdb) s
--Return--
> <string>(1)<module>()->None
(Pdb) l
[EOF]
(Pdb)
```

Single Stepping

```
(Pdb) s
> /opt/python330/lib/python3.3/bdb.py(409)run()
-> self.quitting = True
(Pdb) s
The program finished and will be restarted
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb) dir()
['__builtins__', '__file__', '__name__']
(Pdb)
```

Single Stepping

```
(Pdb) q
$
```

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py 100
```

Invoking Python's Debugger

```
$ python -m pdb fizzbuzz.py 100
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) l
  2       Generate the first n Fizz Buzz answers.
  3
  4       Usage:
  5
  6       > python fizzbuzz.py n
  7   ->  """
  8
  9       import sys
 10
 11       def fizzbuzz(n):
 12           """
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) l
 13             fizzbuzz(n) -> [first n Fizz Buzz answers]
 14             """
 15
 16             answers = []
 17             for x in range(1,n+1):
 18                 answer = ""
 19                 if not x%3:
 20                     answer += "Fizz"
 21                 if not x%5:
 22                     answer += "Buzz"
 23                 if not answer:
(Pdb)
```

Python's Debugger Prompt

```
(Pdb) l
 24                   answer = x
 25               answers.append(answer)
 26           return answers
 27
 28       if __name__ == '__main__':
 29           try:
 30               if len(sys.argv) != 2:
 31                   raise ValueError("Incorrect number of arguments")
 32               answers = fizzbuzz(int(sys.argv[1]))
 33               print(" ".join([str(answer) for answer in answers]))
 34           except:
(Pdb)
```

Setting a Breakpoint

```
(Pdb) b 30
Breakpoint 1 at /Users/cbc/pycarolinas/fizzbuzz.py:30
(Pdb) l 30
 25                 answers.append(answer)
 26             return answers
 27
 28     if __name__ == '__main__':
 29         try:
 30 B           if len(sys.argv) != 2:
 31                 raise ValueError("Incorrect number of arguments")
 32             answers = fizzbuzz(int(sys.argv[1]))
 33             print(" ".join([str(answer) for answer in answers]))
 34         except:
 35             print(__doc__)
(Pdb)
```

Setting a Breakpoint

```
(Pdb) l .
  2       Generate the first n Fizz Buzz answers.
  3
  4       Usage:
  5
  6       > python fizzbuzz.py n
  7   ->  """
  8
  9       import sys
 10
 11       def fizzbuzz(n):
 12           """
(Pdb)
```

Setting a Breakpoint

```
(Pdb) c
> /Users/cbc/pycarolinas/fizzbuzz.py(30)<module>()
-> if len(sys.argv) != 2:
(Pdb) l
 25                 answers.append(answer)
 26             return answers
 27
 28     if __name__ == '__main__':
 29         try:
 30 B->         if len(sys.argv) != 2:
 31                 raise ValueError("Incorrect number of arguments")
 32             answers = fizzbuzz(int(sys.argv[1]))
 33             print(" ".join([str(answer) for answer in answers]))
 34         except:
 35             print(__doc__)
(Pdb)
```

Stepping Into a Function

```
(Pdb) len(sys.argv)
2
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(32)<module>()
-> answers = fizzbuzz(int(sys.argv[1]))
(Pdb)
```

Stepping Into a Function

```
(Pdb) s
--Call--
> /Users/cbc/pycarolinas/fizzbuzz.py(11)fizzbuzz()
-> def fizzbuzz(n):
(Pdb) l
  6     > python fizzbuzz.py n
  7     """
  8
  9     import sys
 10
 11  -> def fizzbuzz(n):
 12         """
 13         fizzbuzz(n) -> [first n Fizz Buzz answers]
 14         """
 15
 16         answers = []
(Pdb)
```

Stepping Into a Function

```
(Pdb) p n
100
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(16)fizzbuzz()
-> answers = []
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(17)fizzbuzz()
-> for x in range(1,n+1):
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(18)fizzbuzz()
-> answer = ""
(Pdb)
```

Stepping Into a Function

```
(Pdb) l
 13             fizzbuzz(n) -> [first n Fizz Buzz answers]
 14             """
 15
 16             answers = []
 17             for x in range(1,n+1):
 18   ->            answer = ""
 19                 if not x%3:
 20                     answer += "Fizz"
 21                 if not x%5:
 22                     answer += "Buzz"
 23                 if not answer:
(Pdb)
```

Stepping Into a Function

```
(Pdb) dir()
['answers', 'n', 'x']
(Pdb) where
  /opt/python330/lib/python3.3/bdb.py(405)run()
-> exec(cmd, globals, locals)
  <string>(1)<module>()
  /Users/cbc/pycarolinas/fizzbuzz.py(32)<module>()
-> answers = fizzbuzz(int(sys.argv[1]))
> /Users/cbc/pycarolinas/fizzbuzz.py(18)fizzbuzz()
-> answer = ""
(Pdb)
```

Stack Frames

```
(Pdb) up
> /Users/cbc/pycarolinas/fizzbuzz.py(32)<module>()
-> answers = fizzbuzz(int(sys.argv[1]))
(Pdb) dir()
['__builtins__', '__doc__', '__file__',
 '__name__', 'fizzbuzz', 'sys']
(Pdb) down
> /Users/cbc/pycarolinas/fizzbuzz.py(18)fizzbuzz()
-> answer = ""
(Pdb) down
*** Newest frame
(Pdb)
```

Stepping Into a Function

```
(Pdb) l
 13             fizzbuzz(n) -> [first n Fizz Buzz answers]
 14             """
 15
 16             answers = []
 17             for x in range(1,n+1):
 18   ->            answer = ""
 19                 if not x%3:
 20                     answer += "Fizz"
 21                 if not x%5:
 22                     answer += "Buzz"
 23                 if not answer:
(Pdb)
```

Stepping Into a Function

```
(Pdb) c
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17
Fizz 19 Buzz Fizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz 31 32
Fizz 34 Buzz Fizz 37 38 Fizz Buzz 41 Fizz 43 44 FizzBuzz 46 47
Fizz 49 Buzz Fizz 52 53 Fizz Buzz 56 Fizz 58 59 FizzBuzz 61 62
Fizz 64 Buzz Fizz 67 68 Fizz Buzz 71 Fizz 73 74 FizzBuzz 76 77
Fizz 79 Buzz Fizz 82 83 Fizz Buzz 86 Fizz 88 89 FizzBuzz 91 92
Fizz 94 Buzz Fizz 97 98 Fizz Buzz
The program finished and will be restarted
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

Listing Breakpoints

```
(Pdb) b
Num Type         Disp Enb   Where
1   breakpoint   keep yes   at /Users/cbc/pycarolinas/fizzbuzz.py:30
        breakpoint already hit 1 time
(Pdb) c
> /Users/cbc/pycarolinas/fizzbuzz.py(30)<module>()
-> if len(sys.argv) != 2:
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(32)<module>()
-> answers = fizzbuzz(int(sys.argv[1]))
(Pdb)
```

Stepping Over a Function

```
(Pdb) n
> /Users/cbc/pycarolinas/fizzbuzz.py(33)<module>()
-> print(" ".join([str(answer) for answer in answers]))
(Pdb) pp answers
[1,
 2,
 'Fizz',
...
 98,
 'Fizz',
 'Buzz']
(Pdb)
```

Stepping Over a Function

```
(Pdb) c
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17
Fizz 19 Buzz Fizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz 31 32
Fizz 34 Buzz Fizz 37 38 Fizz Buzz 41 Fizz 43 44 FizzBuzz 46 47
Fizz 49 Buzz Fizz 52 53 Fizz Buzz 56 Fizz 58 59 FizzBuzz 61 62
Fizz 64 Buzz Fizz 67 68 Fizz Buzz 71 Fizz 73 74 FizzBuzz 76 77
Fizz 79 Buzz Fizz 82 83 Fizz Buzz 86 Fizz 88 89 FizzBuzz 91 92
Fizz 94 Buzz Fizz 97 98 Fizz Buzz
The program finished and will be restarted
> /Users/cbc/pycarolinas/fizzbuzz.py(7)<module>()
-> """
(Pdb)
```

STEPPING OVER A FUNCTION

```
(Pdb) c
> /Users/cbc/pycarolinas/fizzbuzz.py(30)<module>()
-> if len(sys.argv) != 2:
(Pdb) b
Num Type         Disp Enb   Where
1   breakpoint   keep yes   at /Users/cbc/pycarolinas/fizzbuzz.py:30
        breakpoint already hit 3 times
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(32)<module>()
-> answers = fizzbuzz(int(sys.argv[1]))
(Pdb)
```

Stepping Over a Function

```
(Pdb) s
--Call--
> /Users/cbc/pycarolinas/fizzbuzz.py(11)fizzbuzz()
-> def fizzbuzz(n):
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(16)fizzbuzz()
-> answers = []
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(17)fizzbuzz()
-> for x in range(1,n+1):
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(18)fizzbuzz()
-> answer = ""
(Pdb)
```

Stepping Over a Function

```
(Pdb) r
--Return--
> /Users/cbc/pycarolinas/fizzbuzz.py(26)fizzbuzz()->
  [1, 2, 'Fizz', 4, 'Buzz', 'Fizz', ...]
-> return answers
(Pdb)
```

Stepping Over a List Comprehension

```
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(33)<module>()
-> print(" ".join([str(answer) for answer in answers]))
(Pdb) s
--Call--
> /Users/cbc/pycarolinas/fizzbuzz.py(33)<listcomp>()
-> print(" ".join([str(answer) for answer in answers]))
(Pdb) s
> /Users/cbc/pycarolinas/fizzbuzz.py(33)<listcomp>()
-> print(" ".join([str(answer) for answer in answers]))
(Pdb)
```

Stepping Over a List Comprehension

```
(Pdb) r
--Return--
> /Users/cbc/pycarolinas/fizzbuzz.py(33)<listcomp>()->
  ['1', '2', 'Fizz', '4', 'Buzz', 'Fizz', ...]
-> print(" ".join([str(answer) for answer in answers]))
(Pdb)
```

Clear Breakpoints

```
(Pdb) h clear
cl(ear) filename:lineno
cl(ear) [bpnumber [bpnumber...]]
        With a space separated list of breakpoint numbers, clear
        those breakpoints.  Without argument, clear all breaks (but
        first ask confirmation).  With a filename:lineno argument,
        clear all breaks at that line in that file.
(Pdb)
```

Disable Breakpoints

```
(Pdb) h disable
disable bpnumber [bpnumber ...]
        Disables the breakpoints given as a space separated list of
        breakpoint numbers.  Disabling a breakpoint means it cannot
        cause the program to stop execution, but unlike clearing a
        breakpoint, it remains in the list of breakpoints and can be
        (re-)enabled.
(Pdb)
```

Enable Breakpoints

```
(Pdb) h enable
enable bpnumber [bpnumber ...]
        Enables the breakpoints given as a space separated list of
        breakpoint numbers.
(Pdb)
```

Monitor Objects for Changes

```
(Pdb) h display
display [expression]

        Display the value of the expression if it changed, each time execution
        stops in the current frame.

        Without expression, list all display expressions for the current frame.
(Pdb)
```

Conditional Breakpoints

```
(Pdb) h condition
condition bpnumber [condition]
        Set a new condition for the breakpoint, an expression which
        must evaluate to true before the breakpoint is honored.  If
        condition is absent, any existing condition is removed; i.e.,
        the breakpoint is made unconditional.
(Pdb)
```

Skip Over Breakpoints

```
(Pdb) h ignore
ignore bpnumber [count]
        Set the ignore count for the given breakpoint number.  If
        count is omitted, the ignore count is set to 0.  A breakpoint
        becomes active when the ignore count is zero.  When non-zero,
        the count is decremented each time the breakpoint is reached
        and the breakpoint is not disabled and any associated
        condition evaluates to true.
(Pdb)
```

Debugger Macros

```
(Pdb) help alias
alias [name [command [parameter parameter ...] ]]
        Create an alias called 'name' that executes 'command'.  The
        command must *not* be enclosed in quotes.  Replaceable
        parameters can be indicated by %1, %2, and so on, while %* is
        replaced by all the parameters.  If no command is given, the
        current alias for name is shown. If no name is given, all
        aliases are listed.
```

One Time Breakpoints

```
(Pdb) h tbreak
tbreak [ ([filename:]lineno | function) [, condition] ]
        Same arguments as break, but sets a temporary breakpoint: it
        is automatically deleted when first hit.
(Pdb)
```

What Is Debugging

- Execute entire program one step at the time
- Execute only suspect portions of program
- Isolate suspect portions of program

Setting a Trace

```
if __name__ == '__main__':
    try:
        if len(sys.argv) != 2:
            import pdb; pdb.set_trace()
            raise ValueError("Incorrect number of arguments")
        answers = fizzbuzz(int(sys.argv[1]))
        print(" ".join([str(answer) for answer in answers]))
    except:
        print(__doc__)
```

Setting a Trace

```
$ python fizzbuzzNG.py
> /Users/cbc/pycarolinas/fizzbuzzNG.py(32)<module>()
-> raise ValueError("Incorrect number of arguments")
(Pdb)
```

Post-mortem Debugging

```
if __name__ == '__main__':
    if len(sys.argv) != 2:
        raise ValueError("Incorrect number of arguments")
    answers = fizzbuzz(int(sys.argv[1]))
    print(" ".join([str(answer) for answer in answers]))
```

Post-mortem Debugging

```
$ python -i fizzbuzzNG2.py
Traceback (most recent call last):
  File "fizzbuzzNG2.py", line 30, in <module>
    raise ValueError("Incorrect number of arguments")
ValueError: Incorrect number of arguments
>>>
```

Post-mortem Debugging

```
>>> import pdb; pdb.pm()
> /Users/cbc/pycarolinas/fizzbuzzNG2.py(30)<module>()
-> raise ValueError("Incorrect number of arguments")
(Pdb)
```

Debugger Runner

```
>>> import fizzbuzz
>>> import pdb
>>> pdb.run('fizzbuzz.fizzbuzz(100)')
> <string>(1)<module>()
(Pdb) s
--Call--
> /Users/cbc/pycarolinas/fizzbuzz.py(11)fizzbuzz()
-> def fizzbuzz(n):
(Pdb)
```

Questions?

cbc@chriscalloway.org

http://drunkenpython.org/pycarolinas.zip

http://drunkenpython.org/pycarolinas.tgz

http://docs.python.org/py3k/library/pdb.html#module-pdb