Debugging Python code

Different levels of debugging

1. **The bottom of the barrel**: Send <enter person here> an email with the message "*it doesn't work*" & a screenshot of the error message.
2. **Rudimentary debugging**: Inspect variables by adding print statements to the code.
    - Fast & easy, but not very powerful.
3. **Acceptable debugging**: Use a debugger, e.g. pdb or ipdb, to enter the code (e.g. script or method) & execute & inspect selected lines.

Helpful tools

For the editor

- **Pyflakes**: a static tool that checks Python source code for errors.
- **Flake8** (pyflakes + PEP8)

Debuggers

- **pdb** -- The Python Debugger: An interactive source code debugger. Installed by default.
- **ipdb** -- Ipython pdb: An enhanced version of pdb. Install it with `pip install ipdb`

Print statements

```
init_learning_rate = 1.0
learning_rate_decay = 0.1
for step in range(20):
     print('step:', i)
     learning_rate = init_learning_rate * (1. / (1 + learning_rate_decay * step))
     print('learning_rate:', learning_rate)
```

The debugger

example.py

```
def main():
    """
    Main function
    """
    x = 34
    y = 3
    z = y/x
    print(x,y,z)

if __name__ == '__main__':
    main()
```

```
ipdb example.py

> /home/example.py(1)<module>()
----> 1 def main():
      2     """
      3     Main function
```

The debugger

```
ipdb> ?

Documented commands (type help <topic>):
========================================
EOF    bt         cont      enable  jump  pdef    psource  run      unt
a      c          continue  exit    l     pdoc    q        s        until
alias  cl         d         h       list  pfile   quit     step     up
args   clear      debug     help    n     pinfo   r        tbreak   w
b      commands   disable   ignore  next  pinfo2  restart  u        whatis
break  condition  down      j       p     pp      return   unalias  where

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

Undocumented commands:
======================
retval  rv
```

The debugger

```
ipdb> ?s
s(tep)
Execute the current line, stop at the first possible occasion
(either in a function that is called or in the current function).
```

The debugger

Set a breakpoint

```
ipdb> b 8
Breakpoint 1 at /home/example.py:8
```

Execute until the breakpoint

```
c
```

Quit the debugger

```
q
```

If you know what line you're interested in

```
def fcn():
    """
    Main function
    """
    x = 34
    y = 3
    import ipdb
    ipdb.set_trace()
    z = y/x
    print(x,y,z)
```

```
In [2]: fcn()

>
<ipython-input-1-fc4b7d1479ea>(9)fcn
()
     8    ipdb.set_trace()
----> 9   z = y/x
     10   print(x,y,z)
```