Software Carpentry II: Programming Tools

Luís Pedro Coelho

Programming for Scientists

February 5, 2009

Version Control

If your laptop exploded, how many hours of work would you lose?

Advantages

- Maintain project history.
- Sync between computers.
- Sync between project members.
- . . .

Subversion

Subversion: model

1. Repository
2. Checkout
3. Commit

Example

1. Create a repository
2. Create a checkout
3. Edit
4. Commit

Alternative: Simply Copying

I can do this with file copying, no?

Alternative: Simply Copying

I can do this with file copying, no?

1. history
2. diffing
3. merging

Do You Keep Old Versions Around?

- bacteria.py
- bacteria1.py
- bacteria2.py
- bacteria3.py
- bacteria4.py
- bacteria5.py
- bacteriaold.py
- bacteriaold2.py
- bacteria.py2
- ...

Do you explain your changes in the code?

```
def function(x,y,z):
    ...
    # Added 2/4/92 by LPC
    x = 0
    # Added 3/3/93 by XYZ
    x += 1
    # Added 8/23/97 by XYZ
    # Added 18/4/00 by MH
    print x
```

Diff'ing

You have two versions of a file bacteria.py and bacteria2.py.

Find the changes.

Diffs & Patches

A diff is a patch.

hello.py (1)

```python
print 'Hello World'
```

hello.py (2)

```python
print 'Hello World'
print 'How are you?'
```

diff (1)

```
--- hello.py    2009-02-05 17:40:51.000000000 -0500
+++ hello2.py   2009-02-05 17:41:04.000000000 -0500
@@ -1 +1,2 @@
 print 'Hello World'
+print 'How are you doing?'
```

hello.py (1)

```python
print 'Hello World'
```

hello.py (2)

```python
print 'Hello World'
print 'How are you?'
```

diff (2)

```
--- hello2.py   2009-02-05 17:41:04.000000000 -0500
+++ hello.py    2009-02-05 17:40:51.000000000 -0500
@@ -1,2 +1 @@
 print 'Hello World'
-print 'How are you doing?'
```

hello.py (1)

```
print 'Hello World'
```

hello.py (2)

```
print 'Hello World'
print 'How are you?'
```

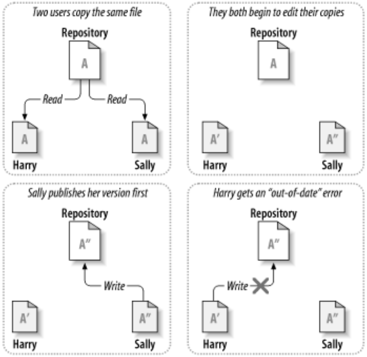
diff (1)

```
--- hello.py    2009-02-05 17:40:51.000000000 -0500
+++ hello2.py   2009-02-05 17:41:04.000000000 -0500
@@ -1 +1,2 @@
 print ’Hello World’
+print ’How are you doing?’
```

Example

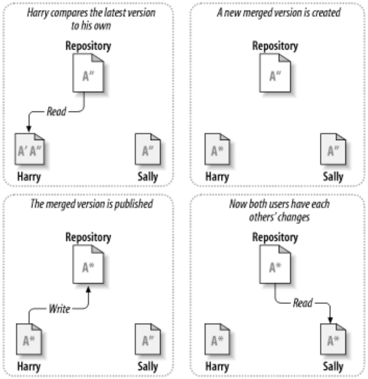
```
diff --git a/pyslic/preprocess/preprocesscollection.py
index f060b8a..a8cd9ec 100644
--- a/pyslic/preprocess/preprocesscollection.py
+++ b/pyslic/preprocess/preprocesscollection.py
@@ -120,7 +120,10 @@ class FixIllumination(object):
         @see see
         '''
         assert self.S is not None
-        self.S /= self.S.min()
+        Smin = self.S.min()
+        if Smin == 0:
+            Smin = 1
+        self.S /= Smin
         # float96 is not always very well supported a
         self.S = numpy.array(self.S,float)
```

Merging


Two users copy the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A'
A''
Harry
Sally
Sally publishes her version first
Repository
A''
Write
A'
A''
Harry
Sally
Harry gets an "out-of-date" error
Repository
A''
Write
A'
A''
Harry
Sally

(From the SVN Book — link on course webpage, under *Notes*)

Merging (II)


Harry compares the latest version to his own
Repository
A″
Read
A′ A″
A″
Harry
Sally
A new merged version is created
Repository
A″
A*
A″
Harry
Sally
The merged version is published
Repository
A*
Write
A*
A″
Harry
Sally
Now both users have each others' changes
Repository
A*
Read
A*
A*
Harry
Sally

(From the SVN Book — link on course webpage, under *Notes*)

Version Control Etiquette

- Don’t commit over my commit.
- Use the log.

Branches and Tags

- Tag: name for revision.
- Branch: multiple parallel tracks of development.

Defensive Programming

Defensive programming means writing code that will catch bugs early.

Assertions

```
def stddev(values):
    '''
    S = stddev(values)

    Compute standard deviation
    '''
    assert len(S) > 0, 'stddev: got empty list.'
    ...
```

Assertions

```
def stddev(values):
    '''
    S = stddev(values)

    Compute standard deviation
    '''
    if len(S) <= 0:
        raise AssertionError(
            'stddev: got empty list.')
    ...
```

```python
def factorial(N):
    '''
    fN = factorial(N)

    Returns the factorial of N.

    N must be equal or greater than zero.
    '''
    if N == 0:
        return 1.
    return N * factorial(N-1)
```

Preconditions

In computer programming, a precondition is a condition or predicate that must always be true just prior to the execution of some section of code.

(Wikipedia)

Preconditions

Other Languages

- C/C++ #include <assert.h>
- Java assert *pre-condition*
- Matlab assert() (in newer versions)
- … …

Assertions Are Not Error Handling!

- Error handling protect against outside events.
- Assertions should never be false.

Testing

Do you test your code?

Unit Testing

```
def test_stddev_const():
    assert stddev([1]*100) < 1e-3

def test_stddev_positive():
    assert stddev(range(20)) > 0.
```

Nosetest

Nose software testing framework:

- Tests are named test_*something*.
- Conditions are asserted.

Software Testing Philosophies

1. Test everything. Test it twice.
2. Write tests first.
3. Regression testing.

Regression Testing

Make sure bugs only appear once!