Lab 03: Version Control

CMPT 145

Laboratory 03 Overview

Part 1 : Pre-Lab Reading (Slide 4)
Part 2 : Laboratory Activities (Slide 15)
Hand In : A screenshot of the History tab showing your work with Git (Slide 38).

Part I

Pre-Lab Reading

Version Control

Version control for everyone

- Version control was invented by programmers, for software development.
- But version control can be applied to any kind of project you create with a computer. E.g.
  - Microsoft Word has version control software built-in (Review tab)
  - PyCharm has multiple version control tools available.
  - Every UNIX system (Mac, Linux, etc) provides version control software by default, usable from the command-line.

What is a version?

- A version is
  - The contents of all the files in your project
  - At a point in time.
  - You decide when to call it a version.
- To make a version:
  - Tell the version control software to backup the current state of the project.
  - It's more helpful to make versions at short intervals.
- Versions are stored in your project folder.
  - Out of sight; it doesn't clutter your project.
  - In a stack. The most recent version is on top.

Without version control

- Computer file systems are not your friend.
  - When you save a change to a project document, you destroy the previous contents.
- If you decide that your changes made your project worse, you cannot go back.
  - Unless you saved your previous version!
  - That's what version control does.
- Gamers will recognize the advantage of having a *saved game*:
  - You can go back to a previous point in the game, and try again, do better!

Do-it-yourself Version Control

You may already be doing version control without any help:

- Using the UNDO/REDO function.
- Maintaining your own backups my making folders called `'old'`, `'older'`, `'olderer'`, etc.
- Remembering the changes you made.

Might work for trivial projects, but not larger ones.

How Version Control Software Helps

- Backs up your work.
- Documents your back up copies.
- Allows you to
  - Return to a previous version.
  - Develop multiple versions in parallel.
  - Switch between versions.
  - Collaborate with multiple collaborators on multiple projects.

Basic Version Control: a walk-through

- Start a new project, assignment, essay:
  - Initialize version control for that project.
- Begin work on the project.
  - Add files to your new project.
- Complete a task for the project:
  - Use version control to make a documented backup (commit).
- Want to return to a previous state?
  - Identify the version you want, and return to it (checkout).

Version Control Software Packages

There are many packages that implement version control.

- Git
- CVS
- SVN
- Mercurial
- Bazaar
- Darcs
- …and many more

We'll use Git, but the concepts are transferable.

Learning Git

- Human nature resists using a new tool.
- Until using Git is a habit, you'll feel it is wasting your time.
- We'll start with simple uses, just for practice.
- Making Version Control a habit is the main purpose.
- In later labs, we'll show more functionality.

The Web has tons of tutorials

- You only need to know what CMPT 145 labs teach. Nothing more is expected.
- But of course, there's a lot more to learn, if you want to go beyond CMPT 145.
- A very good Git tutorial is: `http://www-cs-students.stanford.edu/~blynn/gitmagic`
- Note: most tutorials present Git using command-line tools. We're starting with PyCharm.

Part II

Laboratory Activities

Version Control in PyCharm

Basic Version Control: a walk-through

- Start a new project, assignment, essay:
  - Initialize version control for that project.
- Begin work on the project.
  - Add files to your new project.
- Complete a task for the project:
  - Use version control to make a documented backup (commit).
- Want to return to a previous state?
  - Identify the version you want, and return to it (revert).

Initializing a new project

ACTIVITY:

1. Create a new PyCharm project, called Lab03.
2. Find the VCS menu in PyCharm.
3. Select Enable Version Control Integration...
4. Select Git from the drop-down menu.
5. Click OK

You'll notice PyCharm's interface has changed slightly.

Begin work on your project

ACTIVITY:

1. Create a new Python file, called `fact.py`.
   - PyCharm will ask you about Add files to Git
   - Keep the defaults, and click OK
2. At the bottom of your IDE window, a new tab appears called Version Control.
   - Note: You may need to click on a tiny icon in the bottom left corner of PyCharm window frame to reveal these tabs.
3. Click and explore, but don't do any actions.
4. Notice the buttons on the left side of the Version Control panel.

Complete a task for the project

ACTIVITY: Add some Python code to `fact.py`

```
# CMPT 145 Lab03: Version Control
#

def factorial(n):
    """
    Purpose:
         Calculate the factorial of a non-negative integer
    Pre-conditions:
         n: non-negative integer
    Return:
        a non-negative integer
    """
    pass
```

Save your changes using PyCharm as normal.

Frequent small versions of your project

- Your script and function don't do anything yet, but it's a start.
- Run your script.
- Be sure there are no errors!
- Let's call this a version!

Making a version: commit

ACTIVITY:

1. Find the Commit Changes button  or  , on the left side of the Version Control pane.
   - Click the Commit Changes button.
   - A big window pops up, showing all changes since the previous version.
   - Look for a box called the Commit Message.
2. In Commit Message box, type

```
Added factorial stub function.
```

3. Click Commit at the bottom of the window.

Browsing your changes

ACTIVITY:

1. Find the Log tab in the Version Control panel.
2. You'll see the words HEAD and master and a few words from your commit message.
   - Git keeps a stack of backup versions.
   - HEAD indicates the most recent commit.
   - The name master is the default name for the initial branch created by Git.
   - We won't deal with branching yet, but you can have multiple branches, and each one will need a name.

Complete a task for your project

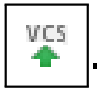
ACTIVITY: Add code to your function:

```
def factorial(n):
    """
    Purpose:
         Calculate the factorial of a non-negative integer
    Pre-conditions:
         n: non-negative integer
    Return:
        a non-negative integer
    """
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

Save your changes using PyCharm as normal.

Committing your new version

ACTIVITY:

1. You have made a change to your program.
2. Let's call this a new version!
3. Click the Commit Changes button  or  .
4. In Commit Message box, type

```
Coded up a recursive implementation of factorial.
```

5. Click Commit at the bottom of the window.

Advice

The better your message is, the more useful it is.

Browsing your changes

1. Find the Log tab in the Version Control panel.
2. The new version is on the top.
3. The HEAD and master have changed.
   - They both refer to the most recent version.
4. You can click on either version, and see the full commit message, and the date it was committed.

Complete a task for your project

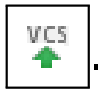
ACTIVITY: Recursion is fine, but you want to change to a loop. Change the code as follows:

```
def factorial(n):
    """
    Purpose:
         Calculate the factorial of a non-negative integer
    Pre-conditions:
         n: non-negative integer
    Return:
        a non-negative integer
    """
    prod = 1
    for i in range(1, n):
        prod = prod * i
    return prod
```

Save your changes using PyCharm as normal.

Committing your new version

ACTIVITY:

1. You have made a change to your program.
2. Let's call this a new version!
3. Click the Commit Changes button  or  .
4. In Commit Message box, type

```
Replaced the recursion in factorial() with a loop.
```

5. Click Commit at the bottom of the window.

Advice

The better your message is, the more useful it is.

Browsing your changes

ACTIVITY:

1. You'll see that the log has updated.
2. Click the top (most recent) version.
3. Find `fact.py` listed under version control (to the right of the log). Click on it.
4. Find the Compare versions button  or  . Click!
5. A window pops up showing two versions of the file:
   - The left side shows the previous version.
   - The right side shows your current version.
   - You can see exactly what changed!
6. Close the window when you're done.

Undoing your work

Advice

This is the important part of the lab.

- Because you have used Git, you have 3 different versions of the function, documented by commit messages.
- After any commit, you can make changes and experiment with ideas.
- If you're happy with your changes, commit  or  .
- If you're not happy, you can:
  - Discard all changes since the last version
  - Select changes to keep and discard.
  - Revert to any version in the past.

Discarding all changes

- You made some changes, but you're not happy.
- You want to return to the state of your project at your most recent commit.
- Click on `fact.py` in the Project pane.
- Find the VCS menu, select Git, select Revert .
- You're back to the most recent committed version.

Discarding some changes

- You made some changes, but you're not happy.
- You want to return to the state of your project at your most recent commit.
- Click on `fact.py` in the Project pane.
- Open the Compare versions window  or  .
- Differences are highlighted.
- Look for >> beside line numbers between the panels.
- Clicking >> moves the previous version into your current version.
- When you're done, close the window.
- If you're happy, commit the version  .

Revert to a version in the past.

- You made some changes, and some commits, but you're really not happy.
- You want to return to the state of your project some time in the past.
- Click on `fact.py` in the Project pane.
- Open the VCS panel, and click the History tab.
- The history of the file is shown, documented by your commit message.
- Click on any version in the past.
- Click on the Get button.

Version Control gives you control

- Remembering to save your work, and back it up, takes practice.
- Giving good commit messages takes practice.
- Using your judgement about when to return to a previous version takes experience.

How to use Version Control for assignments

- Create a new project for your assignment.
- Initialize version control right away, with the empty project.
- Save your work often.
- Commit your work whenever you reach a point where your program does something useful. This should be frequently!
- Never commit a broken program!
- Always give a good commit message. These will help you find what you look for when you need it.

How to use Version Control for debugging

- Commit your code after every bug you fix.
- Document the bug and the fix in the commit message.
- If you ever discover a new bug, you may find it helpful to compare versions, to see what changed.

Activity: Working with VCS

- Add some simple code to test the factorial function. (Hint: `factorial(5) == 120`)
- Save and commit your work at various points.
- Using the History of `fact`, show that the recursive version is correct, but the non-recursive version has a bug.
- Correct the bug in the non-recursive version, and commit your changes.

Part III

Hand In

What To Hand In

- Open the Log tab in the Version Control panel in PyCharm.
- It should show that you've made some commits to `fact.py`
- Take a screen shot of the Version Control panel from PyCharm.
- Submit the screenshot to Lab 03 on Moodle.