ITSE-1402 Intermediate Python

Class 1: Python Review | Course Introduction | Git Overview | IDE

June 1, 2017

Introductions

Instructor

Philip Ulrich

philip.ulrich@austincc.edu

https://www.telegram.me/PhilipUlrich

Education:
BS, Information Technology; Minor in Computer Science

Occupation:
Windows Administrator @ Rackspace

Experience:
- Programming as a hobbyist for over 8 years using a mixture of Bash, Java, PHP, Powershell, Python, Swift, VB Basic, and a little bit of Javascript.
- Currently am a lead developer on an internal tooling project at Rackspace.

Student Introductions

Possible Things You Can Mention:

- Name
- Occupation
- Experience with Programming
- Did you attend Introduction to Python?
- What do you hope to learn from this class?
- What do you hope to do with it afterwards?

Paperwork

- Class Roster
- Course Resources
- Syllabus
- Think Python, Allen B. Downey

Course Plan

Coursework

- In-Class Work
  - Each Class Except Project Work Days
  - 10 Points / Assignment | All or None
  - Reinforce Day’s Topic
- Projects
  - 3 Total - See Syllabus for Dates
  - 100 Points / Project
  - Cumulative Knowledge Application

Course Resources

- **GitHub**
- Cloud 9 IDE

GitHub

Website: https://github.com

Login:

1. You can either use your existing GitHub username or a new one. If you are using a new one, please go through the account sign up process as you normally would.

Purpose:

1. Code repository for the class. Class files will be cloned from here into your workspaces in Cloud 9.
2. (Optional) You can treat assignments/projects as if they were projects you are maintaining on GitHub. We will be maintaining them locally with git, but you also can push this to GitHub to maintain/better visualize.

Course Resources

- GitHub
- **Cloud 9 IDE**

Cloud 9 IDE

Website: https://c9.io

Login:

1. You will sign up for this service under an invite sent to your email. No CC is required (these are “student” accounts).
2. After you enter a couple pieces of info, a link will be sent to your email to set a password.
3. You will now have access to the class workspace and primary IDE we will be utilizing.

Purpose:

1. Primary working environment for the class. We will create a workspace for each class session.
2. Class files will be cloned from GitHub to a C9 workspace. From here, you will work your assignments and (optionally) push changes to your GH repo. Assignments will be graded based workspace contents. When turning in assignments, you will need to “Share” them with user “philipulrich”.

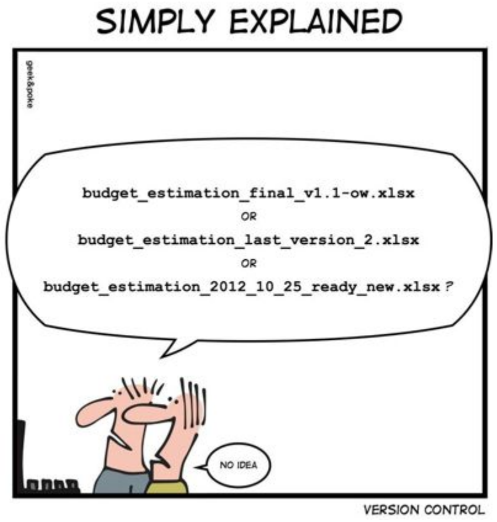
git

git

- What is version control?
- What is Git?
- What is GitHub?
- Using Git day-to-day
- Collaborating online with Github

git

- **What is version control?**
- What is Git?
- What is GitHub?
- Using Git day-to-day
- Collaborating with Github


SIMPLY EXPLAINED
geek&poke
budget_estimation_final_v1.1-ow.xlsx
OR
budget_estimation_last_version_2.xlsx
OR
budget_estimation_2012_10_25_ready_new.xlsx ?
NO IDEA
VERSION CONTROL

- It's a time machine for files
- Checkpoints through a project
- Also known as revision control

http://blog.erichradstake.nl/post/35111755837/simply-explained-version-control

git

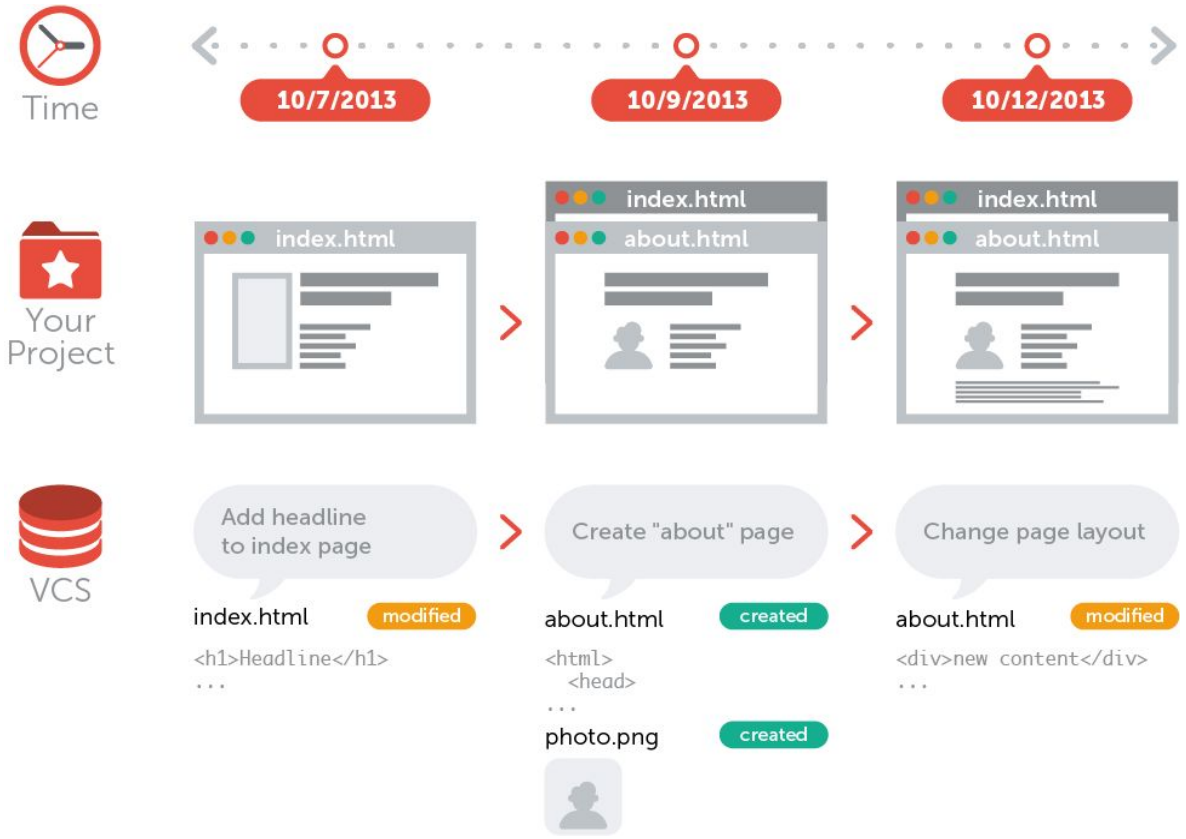
- **What is version control?**
- What is Git?
- What is GitHub?
- Using Git day-to-day
- Collaborating with Github

Used For:

- Types of Files
  - Text-Based Files (scripts, configs, etc.)
  - Binary Files (with exceptions)
  - Images (with exceptions)
- Files Shared With Others
  - Keeps Track of Who Did What
  - Allows Reverting to Previous Versions
- Code Deployment (Or Any Deployment)
  - Ensures the Correct Version is Deployed

git

- **What is version control?**
- What is Git?
- What is GitHub?
- Using Git day-to-day
- Collaborating with Github



https://www.git-tower.com/learn/git/ebook/en/command-line/basics/what-is-version-control

- What is version control?
- **What is Git?**
- What is GitHub?
- Using Git day-to-day
- Collaborating with Github

Definition from the Internet:

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Installing Git

- Git is available for all the major OSes
  - Linux, Windows, OS X
- http://git-scm.com/downloads

* git will already be installed in our workspaces

- What is version control?
- **What is Git?**
- What is GitHub?
- Using Git day-to-day
- Collaborating with Github

What is a Repository?

- A repository is the place that stores all the data for your version controlled directory.
  - More or less, it is the database for all the versions, metadata, file contents, etc.
- A repository can live on a local machine, cloud server, or really any computer.
- Repositories can be cloned and shared

git

Definition from the Internet:

GitHub is a web-based Git or version control repository and Internet hosting service. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.

https://en.wikipedia.org/wiki/GitHub

- GitHub is a web-based Git repository hosting service
  - Centralized repository

- Social coding features
  - Pull requests
  - Documentation (markdown)
  - Wikis
  - Issue tracking
  - Small websites
  - History browsing
  - Code reviews

- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github

Git Commands:

- ***git init***: Create repository
  - **Ex:** git init .
- ***git add***: add file(s) to commit
  - **Ex:** git add text-file.txt image.png
- ***git commit***: commit changes that were added
  - **Ex:** git commit -m “a comment about things”
- ***git push***: push changes to remote (github, gitlab, etc)
  - **Ex:** git push

git

Git Commands (cont.):

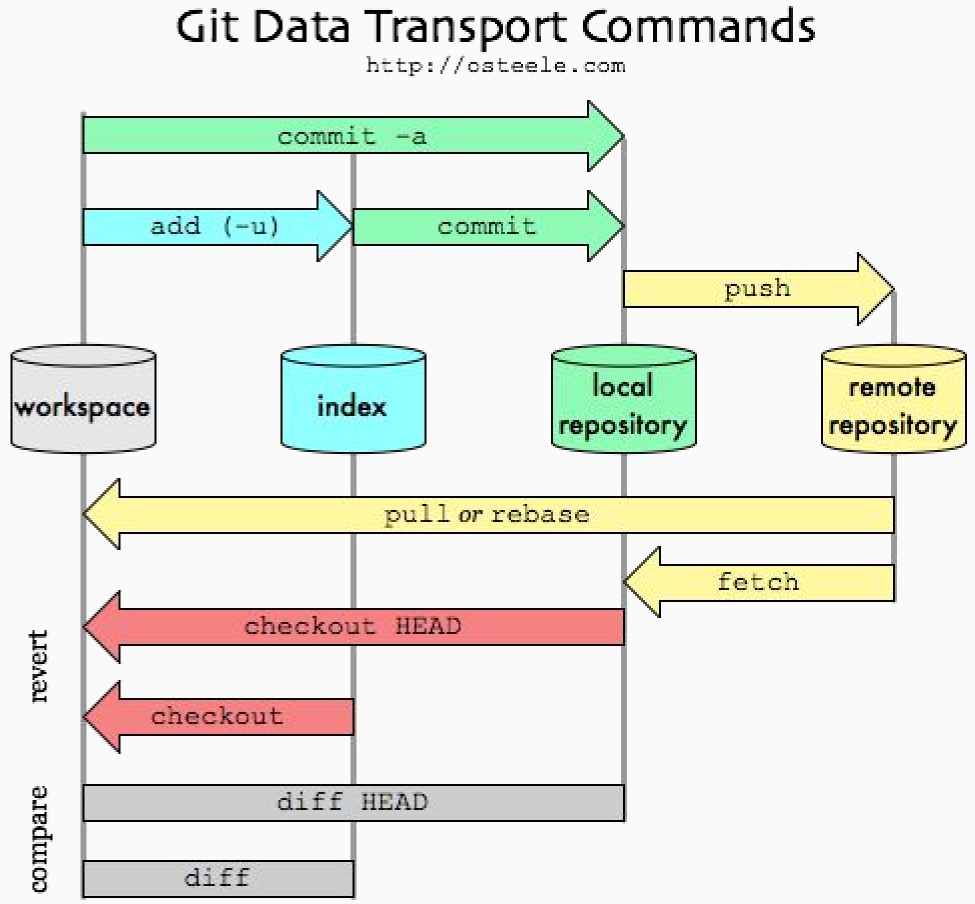
- ***git fetch***: grab latest changes from origin/master
    - **Ex:** git fetch
    - *You likely won't be using this from the get-go*
- ***git pull***: essentially does a git fetch and git merge
    - **Ex:** git pull
    - *More likely to be used. Updates local branch.*
- ***git status***: tells current status of the repo
    - **Ex:** git status

Git Commands (cont.):

- ***git rm***: removes file(s) from the repo
  - **Ex:** git rm text-file.txt image.png
- ***git reset***: removes a file added (git add) before commit
  - **Ex:** git reset other-file.txt
- ***git checkout -- changed-file.txt***
  - Reverts all edits to a specific file

git

- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github



http://blog.osteele.com/2008/05/my-git-workflow/

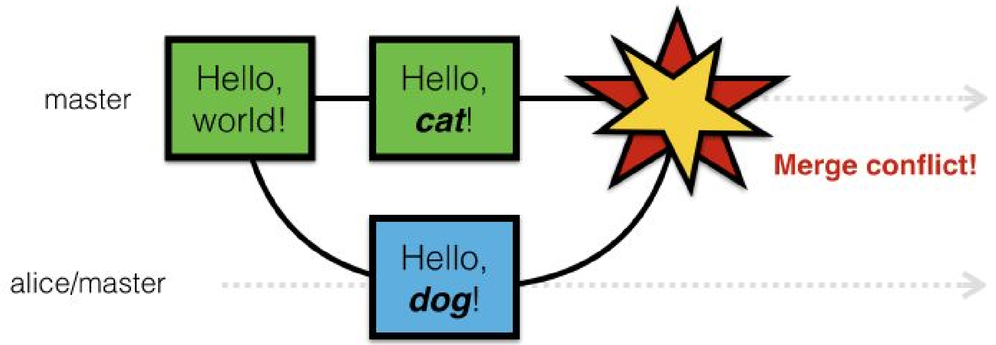
git

Lab 1

git

- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github

Versioning isn't always smooth sailing...



It also isn't always linear.

https://developer.atlassian.com/blog/2015/01/a-better-pull-request/

- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github

Intermediate Git Concepts

Branching

Merging

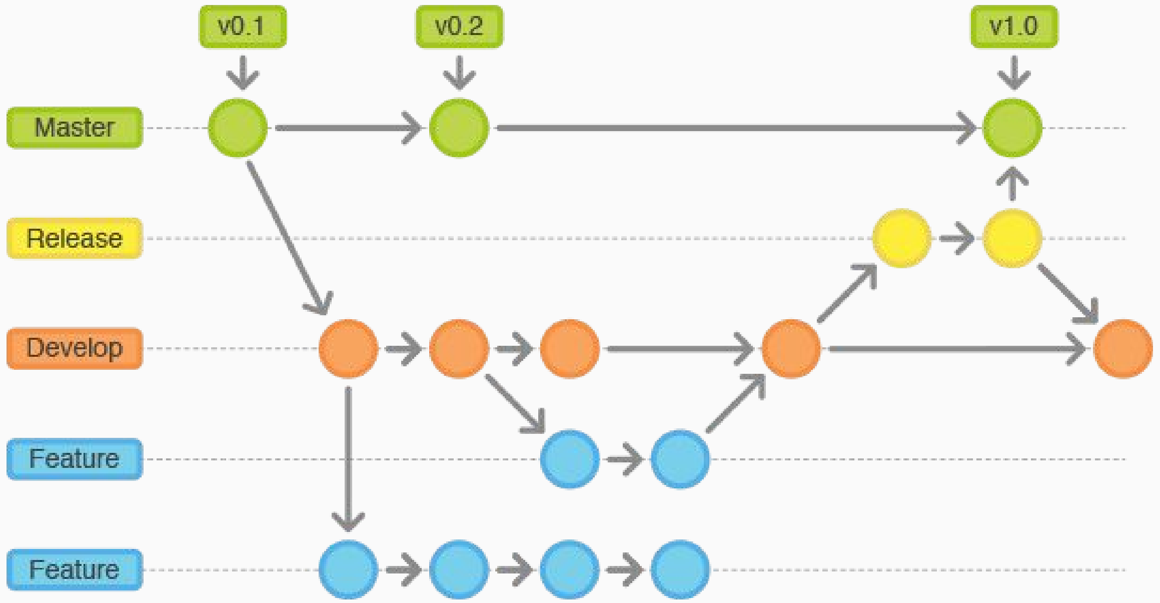
Conflicts

Branches

- A git branch is a separate line of development from the master branch and other branches.
- Branches are typically used for different features or versions of the content being developed.

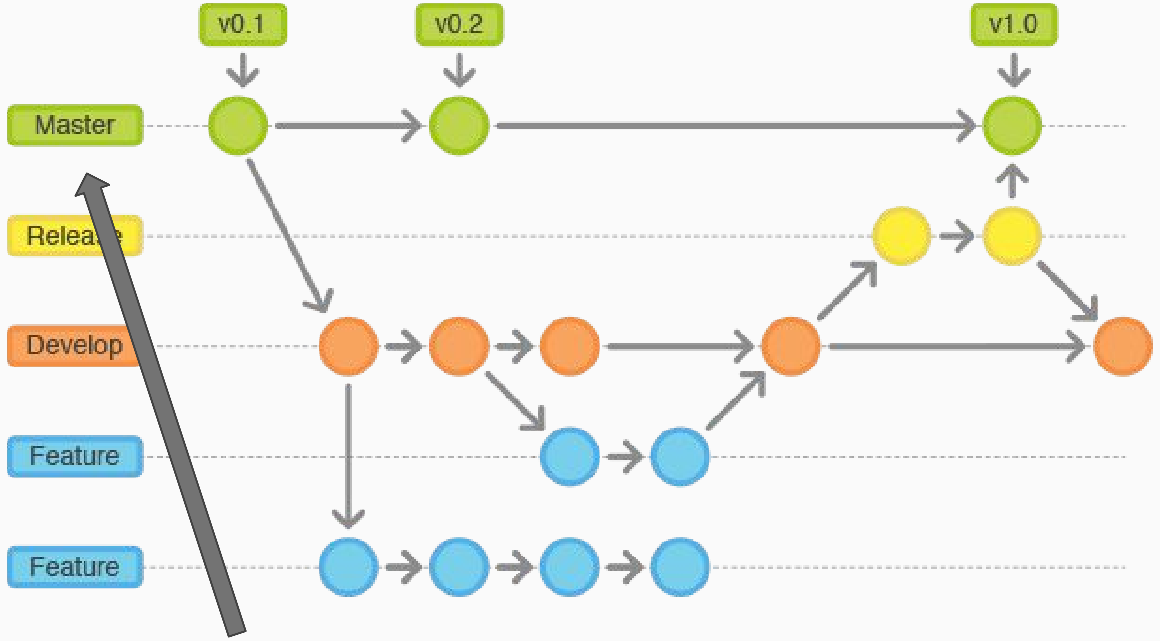
git

- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github



https://leanpub.com/git-flow/read

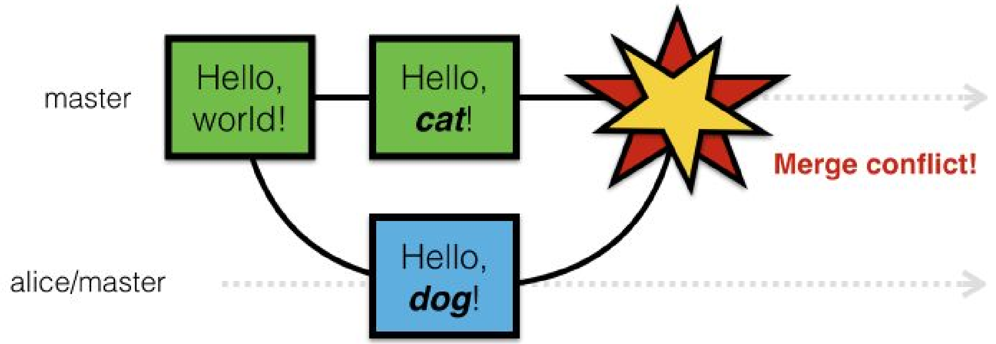
- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github



- The master branch is special and is regarded as the current state of the repository.
- It is considered the core branch that changes are merged into.
- Branches are not strictly required to be merged into the master.
- While the master branch is considered special and the current state of the repo, git sees it as any other branch.

git

Conflicts



- A merge conflict happens when you to merge conflicting changes between branches.
- Example: in your README file
  - master: Hello, cat!
  - alice/master: Hello, dog!
- Git can't tell which one it should go with, so it tells you to pick

- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github

Git Commands:

- ***git branch***: Create branch
  - **Ex:** git branch my_branch
- ***git checkout***: switch to a branch
  - **Ex:** git checkout my_branch
- ***git checkout -b***: shortcut to create and switch
  - **Ex:** git checkout -b my_branch

How to merge:

- Merging branches is when you take the changes from one branch and put them in another.

- Example:
  git checkout master
  git merge my_branch

- What is version control?
- What is Git?
- What is GitHub?
- **Using Git day-to-day**
- Collaborating with Github

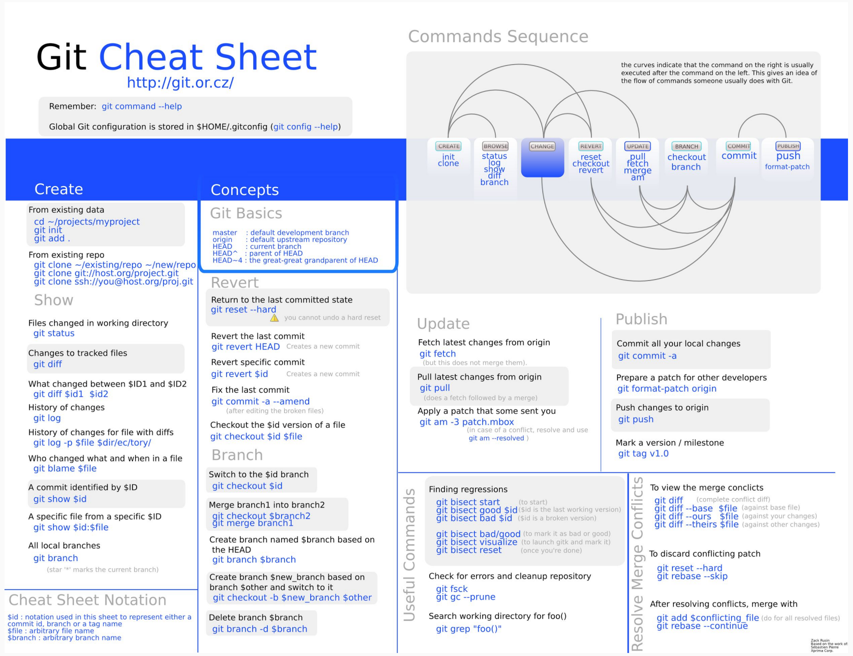
Lab 2

git

- What is version control?
- What is Git?
- What is GitHub?
- Using Git day-to-day
- **Collaborating with Github**

Lab 3

Desk Reference



Git Cheat Sheet
http://git.or.cz/

Remember: git command --help

Global Git configuration is stored in $HOME/.gitconfig (git config --help)

Create

From existing data
```
cd ~/projects/myproject
git init
git add .
```

From existing repo
```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git
```

Show

Files changed in working directory
`git status`

Changes to tracked files
`git diff`

What changed between $ID1 and $ID2
`git diff $id1  $id2`

History of changes
`git log`

History of changes for file with diffs
`git log -p $file $dir/ec/tory/`

Who changed what and when in a file
`git blame $file`

A commit identified by $ID
`git show $id`

A specific file from a specific $ID
`git show $id:$file`

All local branches
`git branch`
(star '*' marks the current branch)

Cheat Sheet Notation

$id : notation used in this sheet to represent either a commit id, branch or a tag name
$file : arbitrary file name
$branch : arbitrary branch name

Concepts

Git Basics

```
master    : default development branch
origin    : default upstream repository
HEAD      : current branch
HEAD^     : parent of HEAD
HEAD~4 : the great-great grandparent of HEAD
```

Revert

Return to the last committed state
`git reset --hard`
⚠ you cannot undo a hard reset

Revert the last commit
`git revert HEAD` Creates a new commit

Revert specific commit
`git revert $id` Creates a new commit

Fix the last commit
`git commit -a --amend`
(after editing the broken files)

Checkout the $id version of a file
`git checkout $id $file`

Branch

Switch to the $id branch
`git checkout $id`

Merge branch1 into branch2
```
git checkout $branch2
git merge branch1
```

Create branch named $branch based on the HEAD
`git branch $branch`

Create branch $new_branch based on branch $other and switch to it
`git checkout -b $new_branch $other`

Delete branch $branch
`git branch -d $branch`

Commands Sequence

the curves indicate that the command on the right is usually executed after the command on the left. This gives an idea of the flow of commands someone usually does with Git.

CREATE: init, clone
BROWSE: status, log, show, diff, branch
CHANGE
REVERT: reset, checkout, revert
UPDATE: pull, fetch, merge, am
BRANCH: checkout, branch
COMMIT: commit
PUBLISH: push, format-patch

Update

Fetch latest changes from origin
`git fetch`
(but this does not merge them).

Pull latest changes from origin
`git pull`
(does a fetch followed by a merge)

Apply a patch that some sent you
`git am -3 patch.mbox`
(in case of a conflict, resolve and use git am --resolved)

Publish

Commit all your local changes
`git commit -a`

Prepare a patch for other developers
`git format-patch origin`

Push changes to origin
`git push`

Mark a version / milestone
`git tag v1.0`

Useful Commands

Finding regressions
```
git bisect start       (to start)
git bisect good $id ($id is the last working version)
git bisect bad $id  ($id is a broken version)

git bisect bad/good (to mark it as bad or good)
git bisect visualize (to launch gitk and mark it)
git bisect reset      (once you're done)
```

Check for errors and cleanup repository
```
git fsck
git gc --prune
```

Search working directory for foo()
`git grep "foo()"`

Resolve Merge Conflicts

To view the merge conclicts
```
git diff      (complete conflict diff)
git diff --base  $file (against base file)
git diff --ours   $file (against your changes)
git diff --theirs $file (against other changes)
```

To discard conflicting patch
```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with
```
git add $conflicting_file (do for all resolved files)
git rebase --continue
```

Zack Rusin
Based on the work of:
Sébastien Pierre
Xprima Corp.

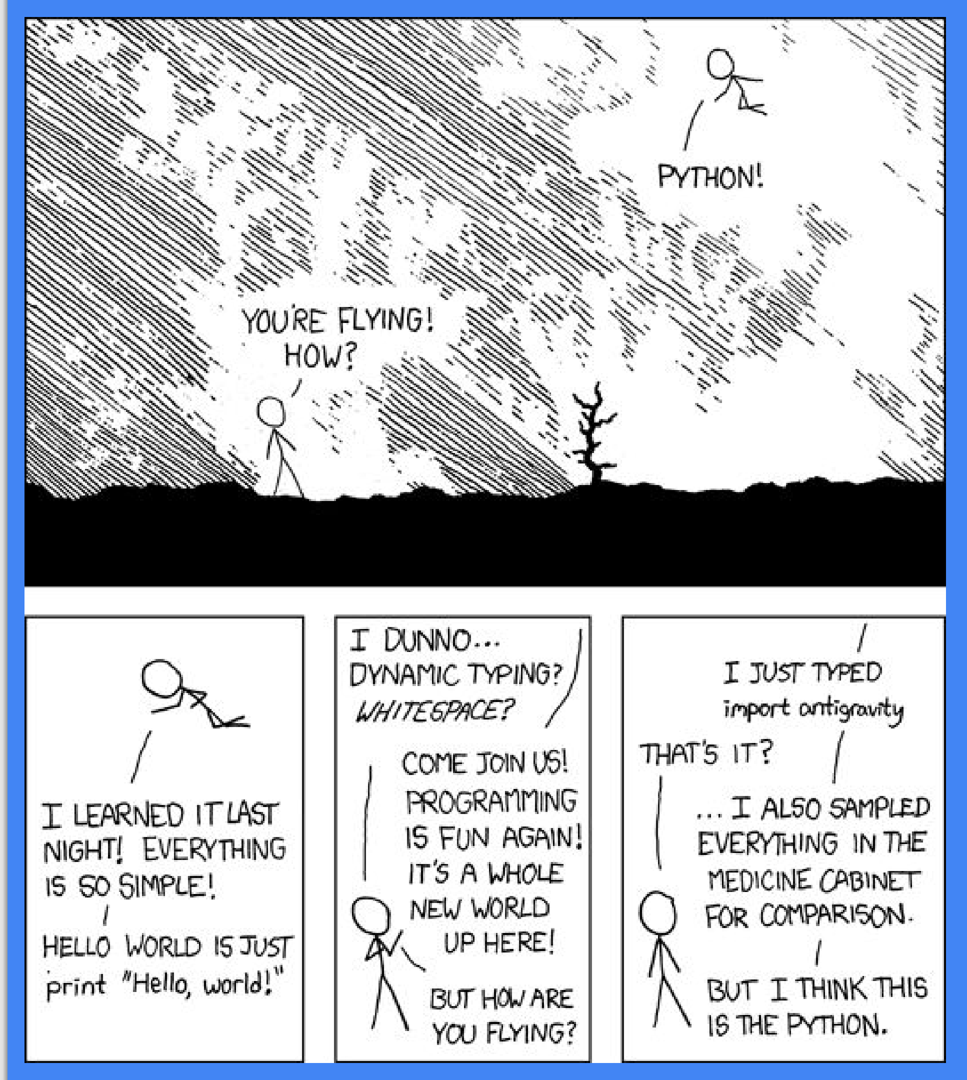
http://byte.kde.org/~zrusin/git/git-cheat-sheet-large.png

Caption:
"I wrote 20 short programs in Python yesterday. It was wonderful. Perl, I'm leaving you."

Python Review

Part 1

https://xkcd.com/353/


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.