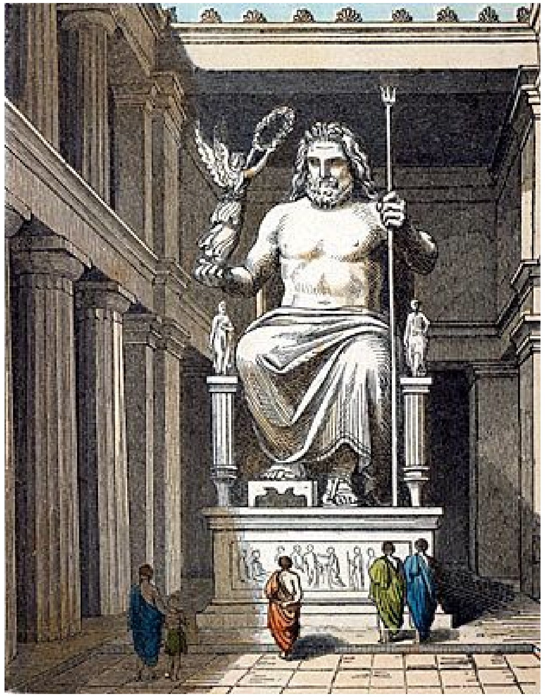
Principles of Computer Science II

Development Tools

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 3



Development Tools

Programming Tool

A programming tool or software development tool is a computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications.

- Source Code Editor
- Debugger or Profiler
- Bug Tracking System
- Documentation Generators
- Revision Control
- Performance Analysis
- Collaborative Programming
- Cloud-based IDEs

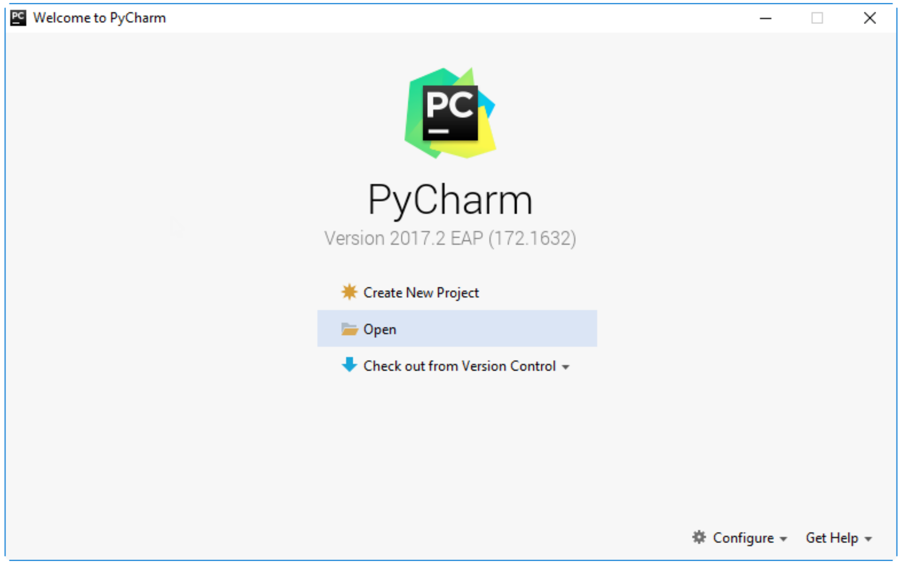
Integrated Development Environment (IDE)

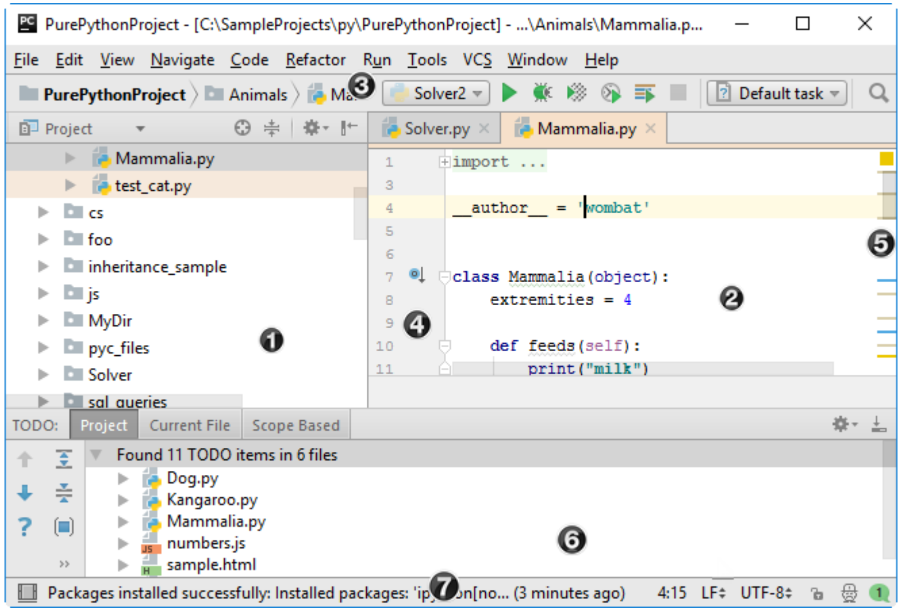
A programming tool or software development tool is a computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications. The IDE is meant to make programming a more productive process.

- Organize project files
- Searching
- Source Code Editor
- Debugger
- Tasks & Annotations related to code
- Documentation Generators
- Revision Control
- Code Analysis

pyCharm: Python IDE for Professional Developers

- Keyboard-centric approach
- Smart assistance
- Code quality tools
- Cross technology development
- Navigation and Refactoring
- Database support
- Scientific tools



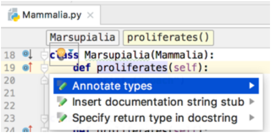


```
import ...

__author__ = 'wombat'


class Mammalia(object):
    extremities = 4

    def feeds(self):
        print("milk")
```

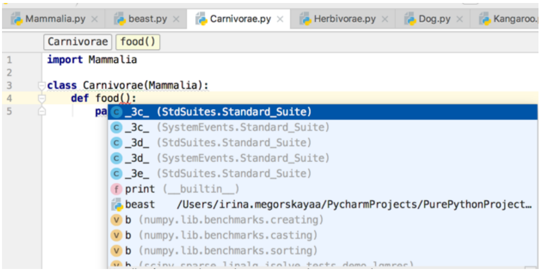
Found 11 TODO items in 6 files

Packages installed successfully: Installed packages: 'ip on[no... (3 minutes ago)

Code with smart assistance


Mammalia.py
Marsupialia
proliferates()
Annotate types
Insert documentation string stub
Specify return type in docstring

```
class Marsupialia(Mammalia):
    def proliferates(self):
```

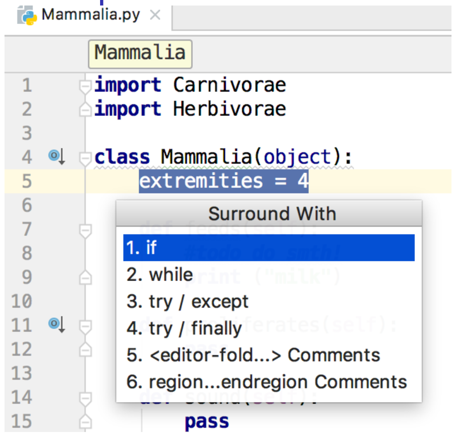


```
import Mammalia

class Carnivorae(Mammalia):
    def food():
        pa
```

```
_3c_  (StdSuites.Standard_Suite)
_3c_  (SystemEvents.Standard_Suite)
_3d_  (StdSuites.Standard_Suite)
_3d_  (SystemEvents.Standard_Suite)
_3e_  (StdSuites.Standard_Suite)
print (__builtin__)
beast   /Users/irina.megorskayaa/PycharmProjects/PurePythonProject…
b (numpy.lib.benchmarks.creating)
b (numpy.lib.benchmarks.casting)
b (numpy.lib.benchmarks.sorting)
```

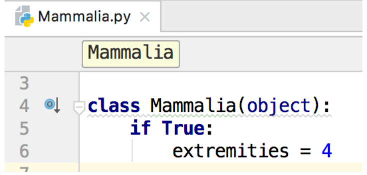
- Intention Action – indicated with a bulb ALT+Enter
  - Suggestions based on the action that you do that intend to save time.
  - Remark that the code needs to be correct for this feature to work.
- Code completion
  - Auto-complete function/variable names.

Live Templates



```
import Carnivorae
import Herbivorae

class Mammalia(object):
    extremities = 4
```

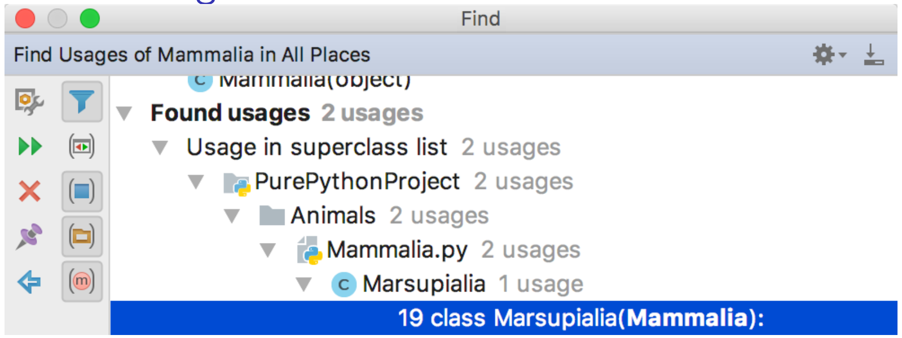
```
        pass
```



```
class Mammalia(object):
    if True:
        extremities = 4
```

- Live Template CTRL+J produce entire code constructs.
- A library of ready-to-use templates.

Search for Usages



Find

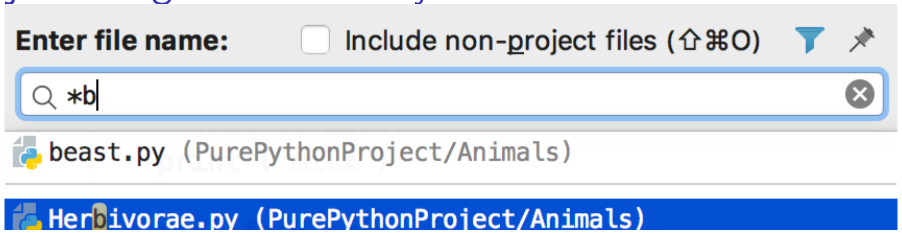
Find Usages of Mammalia in All Places

Mammalia(object)

- Found usages 2 usages
  - Usage in superclass list 2 usages
    - PurePythonProject 2 usages
      - Animals 2 usages
        - Mammalia.py 2 usages
          - Marsupialia 1 usage
            - 19 class Marsupialia(Mammalia):

- As the project grows, or when you work with someone else's code.
- To find where a particular symbol is used, ALT+F7
  - All files are searched.

Project navigation – Find by name



Enter file name: ☐ Include non-project files (⇧⌘O)

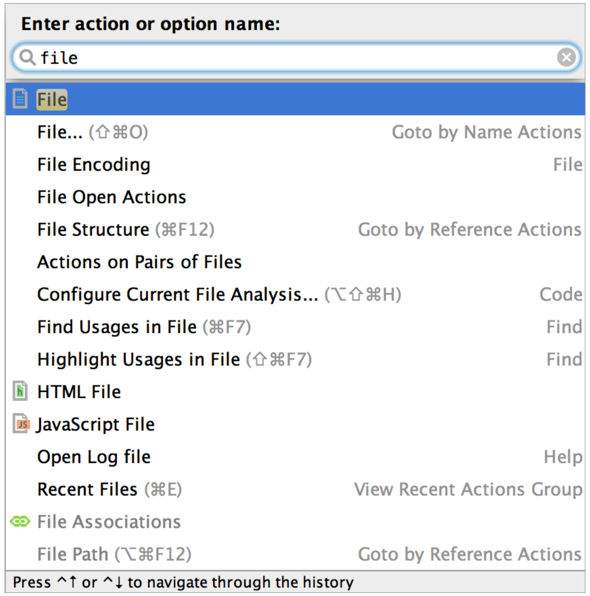
*b

beast.py (PurePythonProject/Animals)

Herbivorae.py (PurePythonProject/Animals)

- Search only Classes by name, CTRL+N
- Search only based on filenames, CTRL+Shift+N
- Search Variable, CTRL+Shift+ALT+N
- Search Declaration, CTRL+B
- Search Class/Function, CTRL+U

Find Action – CTRL+Shift+A

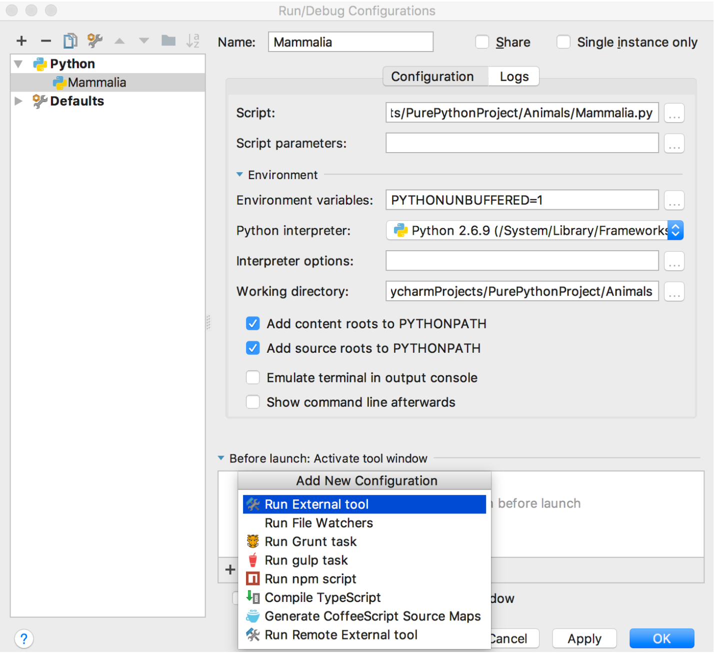


Enter action or option name:

file

File	
File... (⇧⌘O)	Goto by Name Actions
File Encoding	File
File Open Actions	
File Structure (⌘F12)	Goto by Reference Actions
Actions on Pairs of Files	
Configure Current File Analysis... (⌥⇧⌘H)	Code
Find Usages in File (⌘F7)	Find
Highlight Usages in File (⇧⌘F7)	Find
HTML File	
JavaScript File	
Open Log file	Help
Recent Files (⌘E)	View Recent Actions Group
File Associations	
File Path (⌥⌘F12)	Goto by Reference Actions

Press ^↑ or ^↓ to navigate through the history



Run/Debug Configurations

- Python
  - Mammalia
- Defaults

Name: Mammalia ☐ Share ☐ Single instance only

Configuration | Logs

Script: ts/PurePythonProject/Animals/Mammalia.py

Script parameters:

Environment

Environment variables: PYTHONUNBUFFERED=1

Python interpreter: Python 2.6.9 (/System/Library/Frameworks

Interpreter options:

Working directory: ycharmProjects/PurePythonProject/Animals

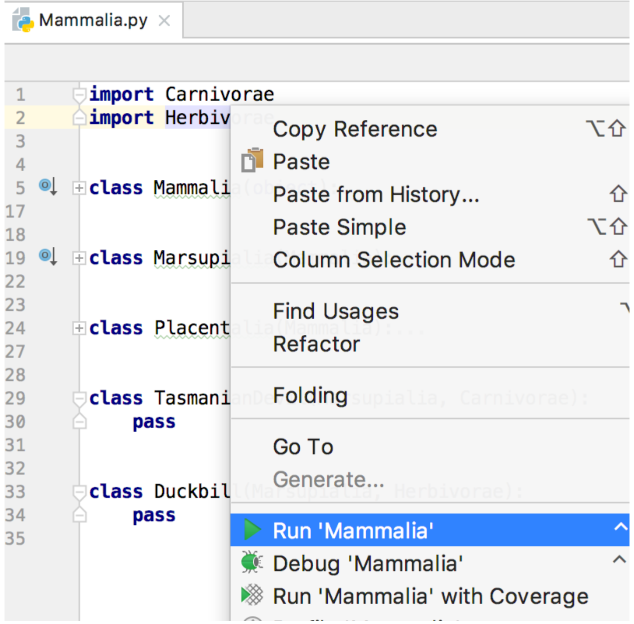
☑ Add content roots to PYTHONPATH

☑ Add source roots to PYTHONPATH

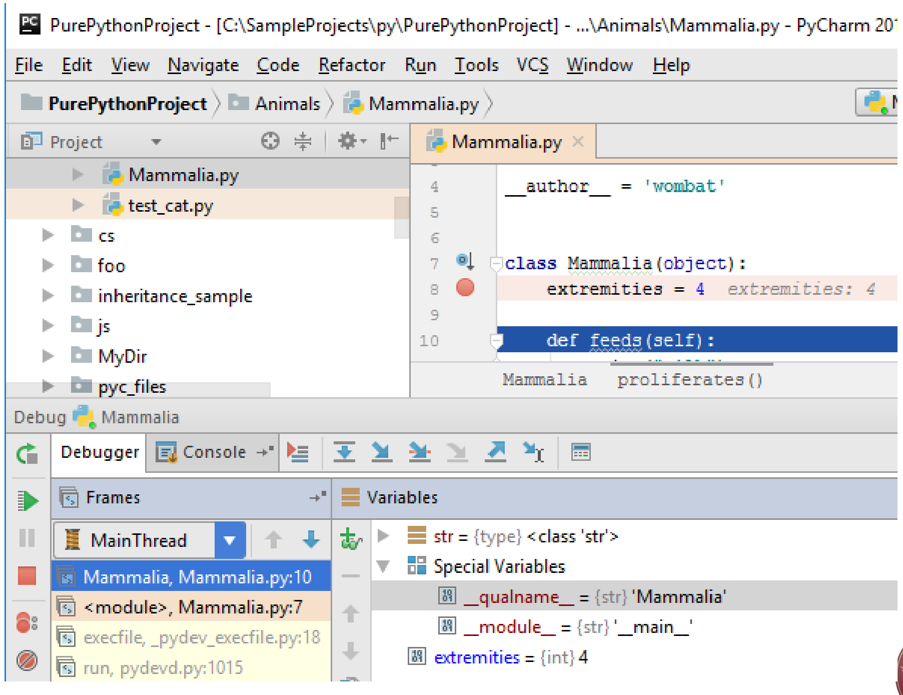
☐ Emulate terminal in output console

☐ Show command line afterwards

Before launch: Activate tool window



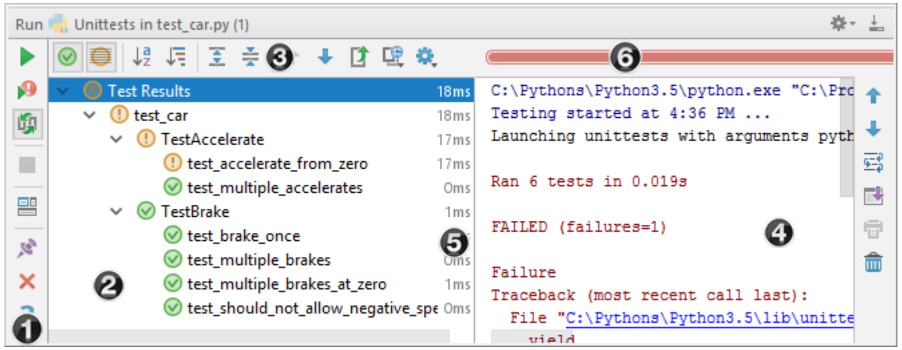
```
import Carnivorae
import Herbiv

class Mammali

class Marsupi

class Placent

class Tasmani
    pass

class Duckbil
    pass
```



```
__author__ = 'wombat'

class Mammalia(object):
    extremities = 4   extremities: 4

    def feeds(self):
```

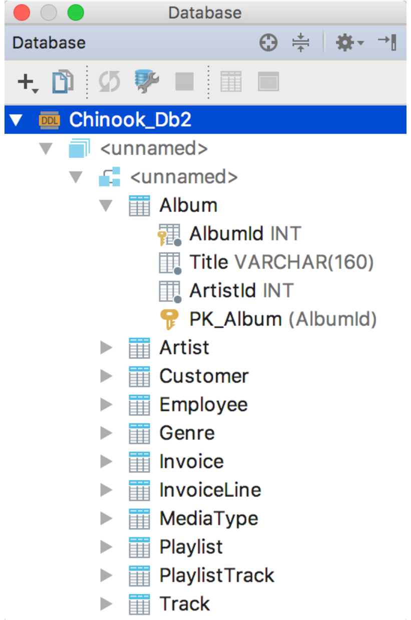
```
Mammalia, Mammalia.py:10
<module>, Mammalia.py:7
execfile, _pydev_execfile.py:18
run, pydevd.py:1015
```

```
str = {type} <class 'str'>
Special Variables
    __qualname__ = {str} 'Mammalia'
    __module__ = {str} '__main__'
extremities = {int} 4
```



```
Test Results                          18ms
  test_car                            18ms
    TestAccelerate                    17ms
      test_accelerate_from_zero       17ms
      test_multiple_accelerates        0ms
    TestBrake                          1ms
      test_brake_once
      test_multiple_brakes             0ms
      test_multiple_brakes_at_zero     1ms
      test_should_not_allow_negative_spe 0ms
```

```
C:\Pythons\Python3.5\python.exe "C:\Prc
Testing started at 4:36 PM ...
Launching unittests with arguments pyth

Ran 6 tests in 0.019s

FAILED (failures=1)

Failure
Traceback (most recent call last):
  File "C:\Pythons\Python3.5\lib\unitte
    yield
```



```
Chinook_Db2
  <unnamed>
    <unnamed>
      Album
        AlbumId INT
        Title VARCHAR(160)
        ArtistId INT
        PK_Album (AlbumId)
      Artist
      Customer
      Employee
      Genre
      Invoice
      InvoiceLine
      MediaType
      Playlist
      PlaylistTrack
      Track
```

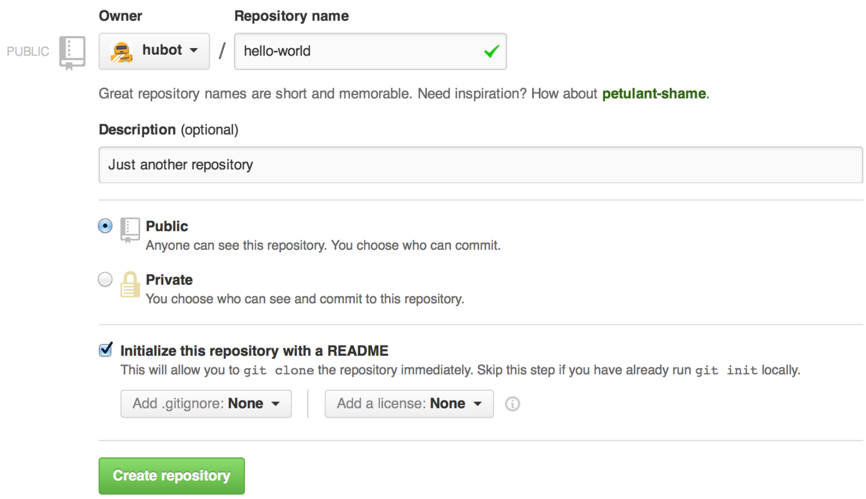
github
SOCIAL CODING

- Code Hosting Platform
  - Version Control, Bug Tracking & Todo list, Wiki, Collaboration, . . .
- Public + Private Projects
- Cloud-based or Private Storage
- Alternatives:
  - BitBucket, SourceFourge, Team Foundation Server, SVN, CVS

First steps on Github

- Repository-oriented Family of Services
  - Repository: group of files relevant to a specific project.
  - Not necessarily related to coding.
- Each member of the project needs a separate account.
- Repositories are owned by an account.
  - Organizations are also allowed to own repositories.
- Repositories are created via the Website.
- Repositories can be browsed/modified via the Web or via broad range of client applications.

Creating a new Repository



Owner **Repository name**

PUBLIC hubot ▾ / hello-world ✓

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

Description (optional)

Just another repository

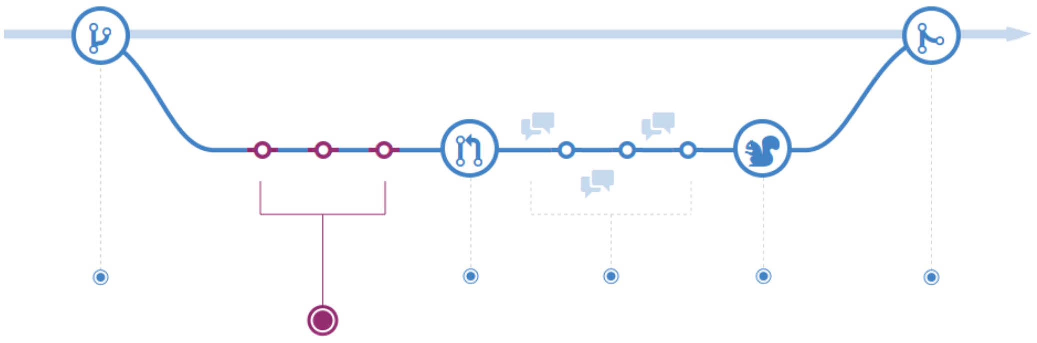
● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

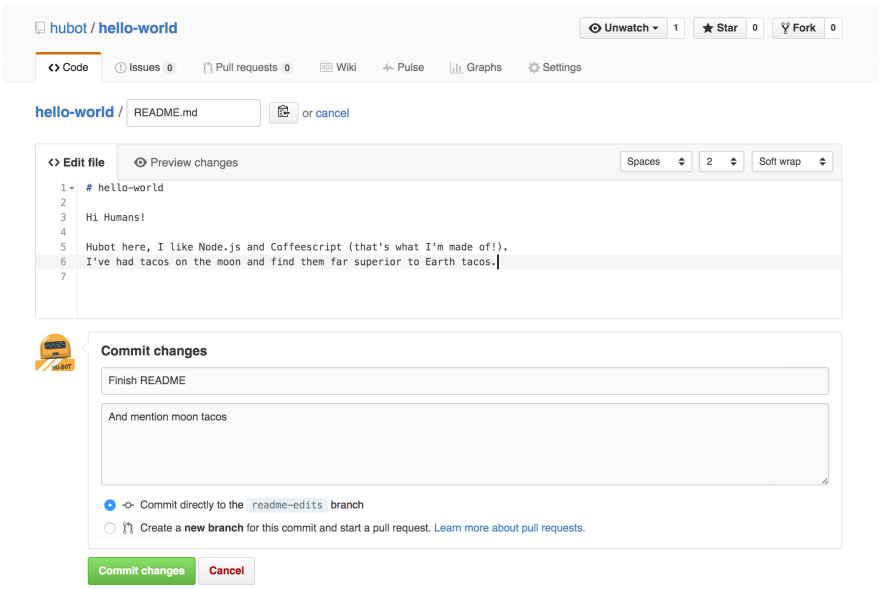
☑ **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** ▾ | Add a license: **None** ▾

Make and commit changes



- Whenever you add, edit, delete.
- Keeps track of progress.
- Easy to roll-back to previous states.



hubot / **hello-world**

hello-world / README.md or cancel

```
# hello-world

Hi Humans!

Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
I've had tacos on the moon and find them far superior to Earth tacos.
```

Commit changes

Finish README

And mention moon tacos

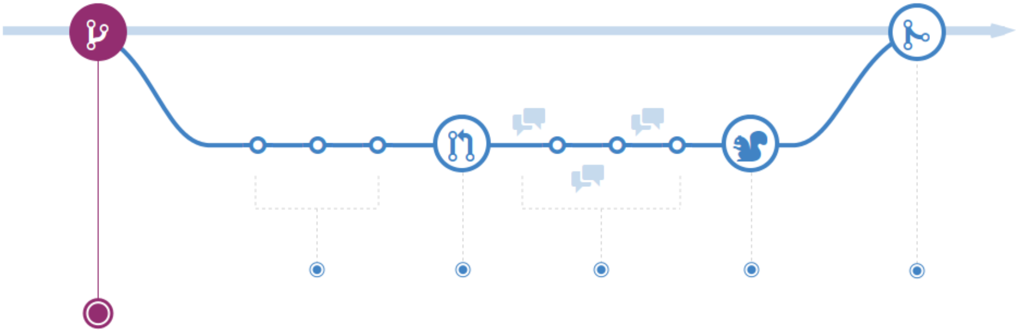
● Commit directly to the `readme-edits` branch

○ Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

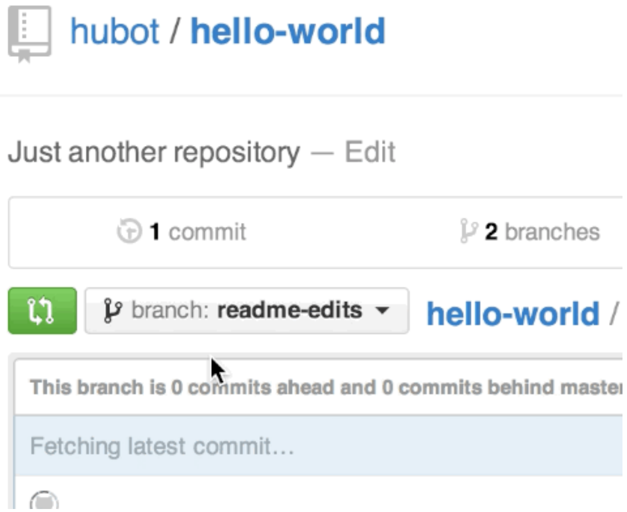
Real power of Github: Branching

- The most over-stressed functionality.
- Branching: work on different versions of a repository at one time.
- By default each repository has 1 branch: master
- When create a new branch off the master:
  - Make a copy of all contents.
  - Changes on new repository are separated.
  - Can pull changes from master at any point.
  - Can push changes to master at any point.

Branching



- Starting from the MASTER branch.
- We create the FEATURE branch.
- The new branch progresses independently.
- Eventually, it MERGES into MASTER.



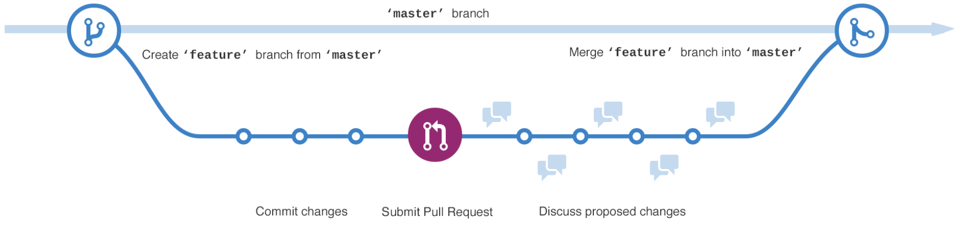
hubot / **hello-world**

Just another repository — Edit

hello-world /

This branch is 0 commits ahead and 0 commits behind maste

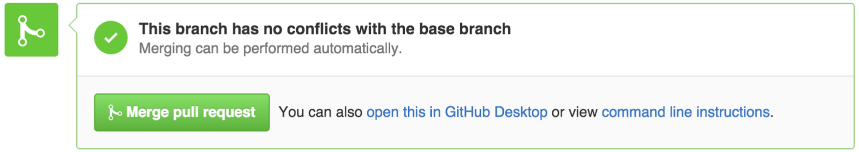
Fetching latest commit…


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

- Communicating changes to the other members of the team is done via PULL REQUESTS.
- Pull Requests are the heart of collaboration on GitHub.
- As soon as you make a commit:
  - open a pull request,
  - start a discussion!

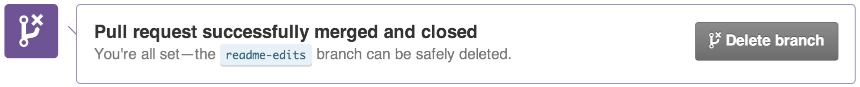
Merge Pull Requests

- The final step of bringing changes together.
- Merging 2 brunches.
- After confirming the merge, other branches can be deleted.



This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.



Pull request successfully merged and closed
You're all set—the `readme-edits` branch can be safely deleted.