

CONCEPTS OF PROGRAMMING

© T.A. BINKOWSKI, 2020

MODULE 4
MPCS 50101



THE UNIVERSITY OF
CHICAGO

© T.A. BINKOWSKI, 2016

OUTLINE

- Class News
- Review
 - Lecture Quiz
 - Unix Challenge
- Breakout
- Module Preview
- Assignment Preview

Module 4: Overview Outline

- Python in VSCode
- Debugging Practices and Tools
 - Python Debugger
 - Debugging with VSCode
- Modules
 - Organizing Modular Code
- Packages
- Anatomy of a File (Module)
- A Few Useful Modules
- With
- Recursion

Reading List

Recommended Reading

- [Learn Python the Hard Way](#) ↗ Exercises 47.
- Think Python - Chapter 4, 5.8
- [Python is compiled language or interpreted language ?](#) ↗

Links

CLASS NEWS

© T.A. BINKOWSKI, 2020

MODULE 4
MPCS 50101



THE UNIVERSITY OF
CHICAGO

CLASS NEWS

- Read the problems carefully
 - User input only if its stated
 - Creating a function (show test snippet using it)
- There are many ways to solve the problems, use the best way for you
- Watch what function `return` versus modify

CLASS NEWS

- Best practices are always implied
 - Docstrings
 - Variable names
 - Comments



RUBRIC WILL HAVE BEST
PRACTICES FOR EVERY
PROBLEM

REVIEW

MODULE 4
MPCS 50101

© T.A. BINKOWSKI, 2020



THE UNIVERSITY OF
CHICAGO

UNIX CHALLENGE

UNIX CHALLENGE

Unix Challenge

```
cat speech.txt | tr -d '[:punct:]'
```

UNIX CHALLENGE

NAME

tr -- translate characters

SYNOPSIS

```
tr [-Ccsu] string1 string2
tr [-Ccu] -d string1
tr [-Ccu] -s string1
tr [-Ccu] -ds string1 string2
```



DESCRIPTION

The **tr** utility copies the standard input to the standard output with substitution or deletion of selected characters.

The following options are available:

- C** Complement the set of characters in string1, that is ``**-C** ab'' includes every character except for `a' and `b'.
- c** Same as **-C** but complement the set of values in string1.
- d** Delete characters in string1 from the input.
- s** Squeeze multiple occurrences of the characters listed in the last operand (either string1 or string2) in the input into a single instance of the character. This occurs after all deletion and translation is completed.

LECTURE QUIZ

REVIEW

LECTURE QUIZ

Question

What will be printed by the following statement?

```
x = "once upon a time"  
print(x.title())
```

Answers

Once Upon A Time

Question 1

The `try/except` is the only way to check for runtime errors?

True

False

Answer

Question 2

The `finally` statement is required at the end of a `try/except` block.

True

False

Answer

LECTURE QUIZ

frozenset() in Python

Last Updated : 19 Dec, 2018

The **frozenset()** is an inbuilt function in Python which takes an iterable object as input and makes them immutable. Simply it freezes the iterable objects and makes them unchangeable.

In Python, frozenset is same as set except its elements are immutable. This function takes input as any iterable object and converts them into immutable object. The order of element is not guaranteed to be preserved.

Question 3

A **set** is a collection type in Python. Which of the following are true for objects of Python's **set** type:

- Sets can only be used with number types.
- The elements in a set are unique.
- A sets items can be accessed by index.
- The items in a set are ordered.

Correct Answer

Question

In Python, **frozenset** is the same as **set** except that **frozenset**s are immutable (ie. elements from the **frozenset** cannot be removed once created).

Correct Answer

- True
- False

Question 4

Which of the following defines the set `{'a', 'b', 'c'}`:

Correct Answer

- `s = {'a', 'b', 'c'}`

Correct Answer

- `s = set(['a', 'b', 'c'])`

Correct Answer

- `s = set('abc')`

Correct Answer

- `s = set(['a', 'b', 'c'])`

- `s = {('a', 'b', 'c')}`

LECTURE QUIZ

Question 5

Given the dictionary :

```
x = [ 'a', 'b',
{ 'foo': 1,
'bar': {
'x' : 10,
'y' : 20,
'z' : 30 },
'baz': 3 },
'c',
'd' ]
```

What is a single line expression will access the value 20?

Correct Answers

- x[2]["bar"]["y"]
- x[2]['bar']['y']
- x[2].get('bar').get('y')
- x[2].get("bar").get("y")

Question 6

What is printed from the following code snippet?

```
list = [6,4,2,3]
sorted_list = list.sort()
print(sorted_list)
```

Correct Answers

- None

LECTURE QUIZ

Question 5

Given the dictionary :

```
x = [ 'a', 'b',
{ 'foo': 1,
'bar': {
'x' : 10,
'y' : 20,
'z' : 30 },
'baz': 3 },
'c',
'd' ]
```

What is a single line expression will access the value 20?

Correct Answers

- x[2]["bar"]["y"]
- x[2]['bar']['y']
- x[2].get('bar').get('y')
- x[2].get("bar").get("y")

Question 6

What is printed from the following code snippet?

```
list = [6,4,2,3]
sorted_list = list.sort()
print(sorted_list)
```

Correct Answers

- None

LECTURE QUIZ

Question 7

Evaluate the following statement:

$(1,5,100,10,1) < (1,5,10,100,1)$

True

Correct Answer

False

Question 8

Given a text file named file.txt with the following lines of data:

1
2
3
4
5
6

What is printed by the following code snippet?

```
f = open('infile.txt', 'r')
file_text = f.readline().strip()
file_text = f.readline()
file_text = f.readline().strip()
print(file_text)
f.close()
```

Correct Answers

3

LECTURE QUIZ

Question 9

1 pts

In general, comments are for the developers and docstrings are for users.

Correct Answer

True

False

Question 10

1 pts

What is the value of string after the following code snippet is run?

```
string = "Computer Science is no more about computers than astronomy is about telescopes.\n\t- Edsger W. Dijkstra"
string = string.rstrip()
string = string.split('c')
string = string[3].split()[2]
```

Correct Answers

astronomy

BREAKOUT EXERCISES

© T.A. BINKOWSKI, 2020

MODULE 4
MPCS 50101

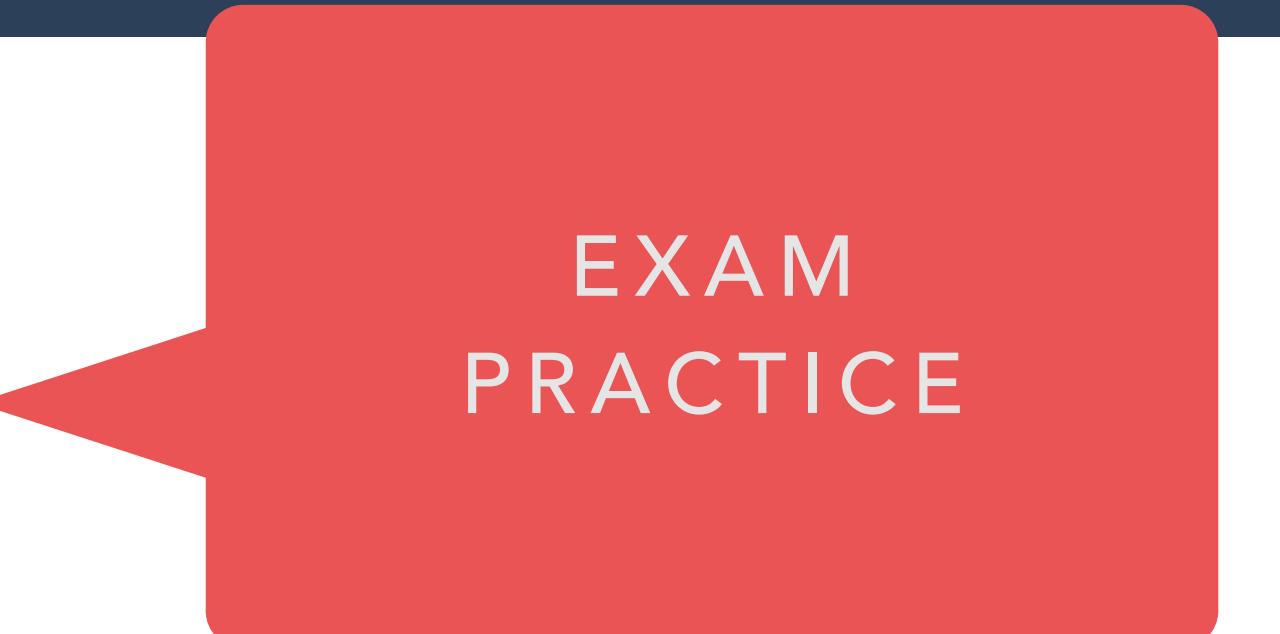


THE UNIVERSITY OF
CHICAGO

BREAKOUT EXERCISES

Module 4: Breakout Exercises

Exercise 1



EXAM
PRACTICE

In preparation for your upcoming exam, try to work these problems out manually. You may check your answers once you have agreed on a solution.

A. What is printed to the screen?

```
x = ['hello', 'world']
for i in x:
    i.lower()
print(x)
```

B. What is the output of the following code snippet?

DEBUGGING PRACTICES AND TOOLS

© T.A. BINKOWSKI, 2020

MODULE 4
MPCS 50101



THE UNIVERSITY OF
CHICAGO

DEBUGGING

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

- Brian W. Kernighan, computer scientist, 1942-

DEBUGGING

- Testing a program to find errors can take more time than writing the code
- "Debugger" is a program that overrides the execution of a program to allow you to inspect code
- Python Debugger (PDB)
- Others...

```
In [1]: run -d debugging1.py
Breakpoint 1 at /Users/tabinkowski/Google Drive/mpcs50101-2017-winter/lectures/session9/debugging1.py:1
NOTE: Enter 'c' at the ipdb> prompt to continue.
> /Users/tabinkowski/Google Drive/g-Teach/2017-winter/lectures/session9/debugging1.py(1)
1----> 1 def bubbleSort(alist):
2         for passnum in range(len(alist)):
3             for i in range(passnum):
4                 if alist[i]>alist[i+1]:
5                     temp = alist[i]
ipdb> s
> /Users/tabinkowski/Google Drive/g-Teach/2017-winter/lectures/session9/debugging1.py(9)
9
10
---> 11 def print_list(list):
12     print list
13
ipdb> s
> /Users/tabinkowski/Google Drive/g-Teach/2017-winter/lectures/session9/debugging1.py(12)
12     print list
13
---> 14 alist = [5,4,3,2,1]
15 bubbleSort(alist)
16 print(alist)
ipdb>
```

DEBUGGING

- "Debugger" allows you to go through your code and observe
 - Start/stop code
 - Print variables
 - Execute statements one-by-one

PYTHON DEBUGGER

DEBUGGING

pdb – The Python Debugger

Source code: [Lib/pdb.py](#)

The module `pdb` defines an interactive source code debugger for Python programs. It supports setting (conditional) breakpoints and single stepping at the source line level, inspection of stack frames, source code listing, and evaluation of arbitrary Python code in the context of any stack frame. It also supports post-mortem debugging and can be called under program control.

The debugger is extensible – it is actually defined as the class `Pdb`. This is currently undocumented but easily understood by reading the source. The extension interface uses the modules `bdb` and `cmd`.

The debugger's prompt is `(Pdb)`. Typical usage to run a program under control of the debugger is:

- `pdb` - an interactive source code debugger for Python programs

```
<>> import pdb
<>> import mymodule
<>> pdb.run('mymodule.test()')
> <string>(0)?()
(Pdb) continue
> <string>(1)?()
(Pdb) continue
NameError: 'spam'
> <string>(1)?()
(Pdb)
```

Changed in version 3.3: Tab-completion via the `readline` module is available for commands and command arguments, e.g. the current global and local names are offered as arguments of the `p` command.

`pdb.py` can also be invoked as a script to debug other scripts. For example:

```
python3 -m pdb myscript.py
```

When invoked as a script, `pdb` will automatically enter post-mortem debugging if the program being

DEBUGGING

- Access PDB debugger
 - Programmatic
 - Command line
 - Integrated in IDE

```
import pdb

def iter_hello(i):
    pdb.set_trace()
    for n in range(i):
        print "hello"
```

```
x = 10
iter_hello(x)
```

DEBUGGING WITH PDB

- Functionality is the same
- Usability is very different

```
[483 % python debug_pdb.py
> /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs50101-
2017-winter/lectures/session9/debug_pdb.py(6)iter_hello()
-> for n in range(i):
(Pdb) l
 1
 2     import pdb
 3
 4     def iter_hello(i):
 5         pdb.set_trace()
 6     ->     for n in range(i):
 7             print "hello"
 8
 9
10
11     x = 10
(Pdb) s
> /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs50101-
2017-winter/lectures/session9/debug_pdb.py(7)iter_hello()
-> print "hello"
(Pdb) print n
0
(Pdb) s
hello
> /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs50101-
2017-winter/lectures/session9/debug_pdb.py(6)iter_hello()
-> for n in range(i):
(Pdb) print n
0
(Pdb) s
```

DEBUGGING WITH VS CODE

DEBUGGING

A screenshot of the Visual Studio Code (VS Code) interface, specifically demonstrating the Python debugger. The title bar shows "demo.py — mpcs50101-2021-winter-canvas". The left sidebar has icons for file operations, search, and other tools. The main area shows the code editor with "Extension: Python" selected. A red callout box points to the "DEBUG" button in the toolbar, which is highlighted with a yellow background. The code in "demo.py" is:

```
1
2
3 numbers = [1,2,3,4,5,6]
4
5 for number in numbers:
6     print(number)
7
```

The line "print(number)" is highlighted with a green background, indicating it is the current line of execution. The variable sidebar on the left shows the "Locals" section with "number" set to 6.

DEBUGGING

A screenshot of a code editor interface, likely Visual Studio Code, illustrating the concept of debugging. The main window shows a Python file named `debug_words.py` with the following code:

```
1
● 2  tagged_words = []
3
4  string = "... This is a test of debugging! ..."
5  clean_string = string.strip()
6  normalize = clean_string.lower()
7  words = normalize.split()
8  for word in words:
9      word = ">> " + word
10     tagged_words.append(word)
11
12 print(tagged_words)
13
14
```

The second line of code, `tagged_words = []`, has a red dot next to it, indicating it is a breakpoint. A red callout bubble with the text "BREAKPOINT WILL STOP YOUR CODE" points to this breakpoint. The code editor's interface includes a top bar with tabs for "DEBU...", "debug_words.py", and "debug_timer.py", and a sidebar on the left with icons for file operations, variables, and other settings.

DEBUGGING

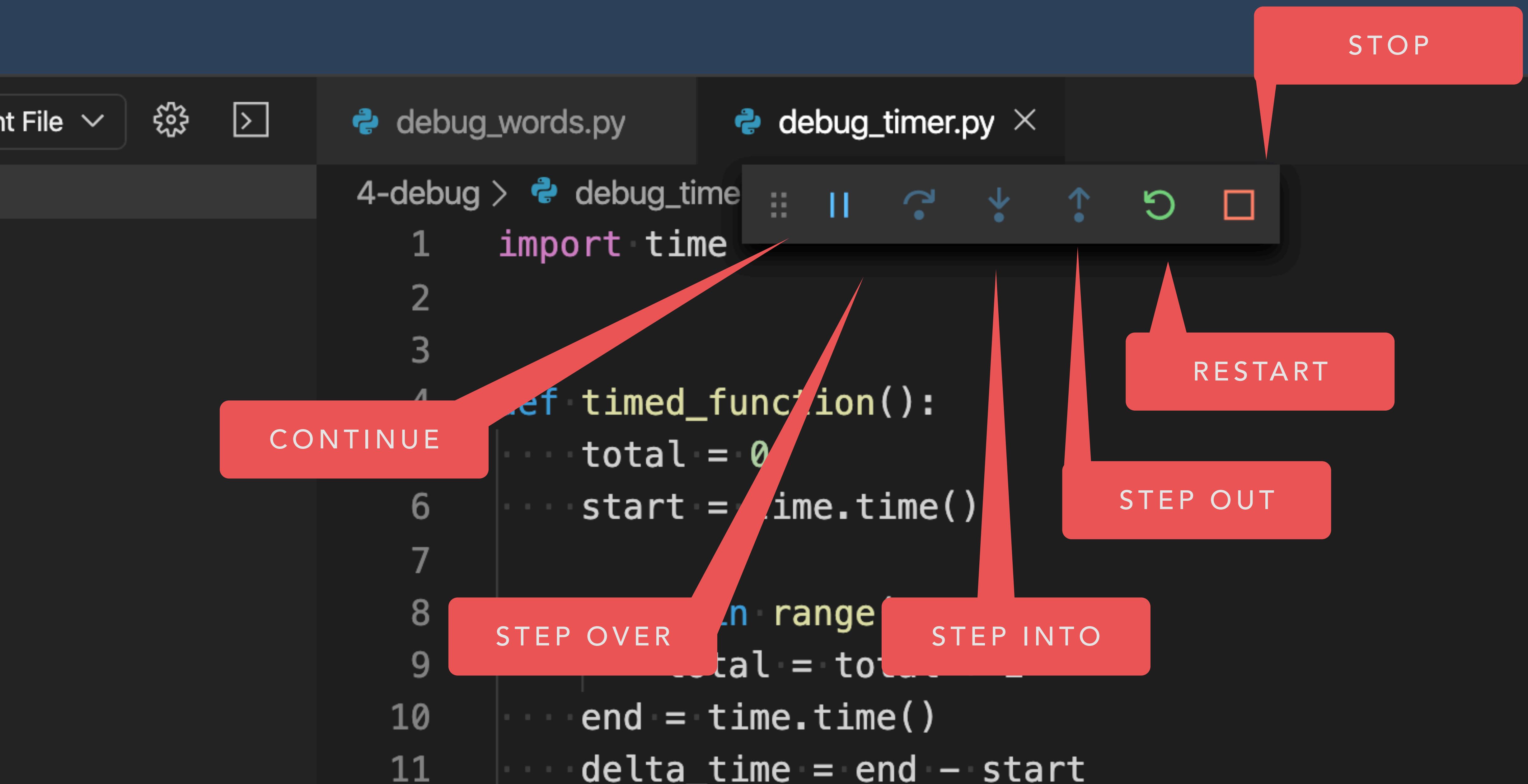
The screenshot shows a Python debugger interface with the following details:

- Title Bar:** debug_words.py — 2020-session-4
- Toolbar:** DEBUG AND RUN (Pyt), Settings, Help.
- Variables View:** Shows the **Locals** scope with the following variables:
 - clean_string: 'This is a test o...
 - string: ' This is a test of d...
 - tagged_words: []
 - __builtins__: {'Arithmeti...
 - __cached__: None
 - __doc__: None
 - __file__: '/Users/tabinkowski/G...
 - __loader__: None
 - __name__: '__main__'
 - __package__: ''
 - __spec__: None
- Code Editor:** Displays the `debug_words.py` file with the following code:

```
1
2     tagged_words = []
3
4     string = "... This is a test of debugging! ..."
5     clean_string = string.strip()
6     normalize = clean_string.lower()
7     words = normalize.split()
8     for word in words:
9         word = ">>" + word
10    tagged_words.append(word)
11
```

The line `normalize = clean_string.lower()` is highlighted with a yellow background.
- Status Bar:** Shows the current state as "4-debug > debug_words.py".
- Bottom Text:** VARIABLES WHEN STOPPED (with a red arrow pointing from the text to the Variables view).

DEB U G G I N G



DEBUGGING

- Continue
- Step Over - next line
- Step Into - go into function
- Step Out - exit function
- Restart
- End

```
> ⚙ debug_time :: || ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉
```

```
import time
```

```
def timed_function():
    total = 0
    start = time.time()

    for i in range(1000):
        total = total + i
    end = time.time()
    delta_time = end - start
    return delta_time

how_long = timed_function()
print(how_long)
```

DEBUGGING

```
> time: <module 'time' (built-in)>      2
> timed_function: <function timed...      3
> __builtins__: {'ArithmetError...      4
__cached__: None
__doc__: None
__file__: '/Users/tabinkowski/G...
__loader__: None
__name__: '__main__'
__package__: ''
__spec__: None
def timed_function():
    total = 0
    start = time.time()
    for i in range(1000):
        total = total + i
    end = time.time()
    delta_time = end - start
    return delta_time
how_long = timed_function()
print(how_long)

```

> WATCH

> CALL STACK

PAUSED ON STEP

✓ BREAKPOINTS

STEP INTO

DEBUGGING

```
i: 5
start: 1580246421.10399
total: 15
2
3
4 def timed_function():
5     total = 0
6     start = time.time()
7
8     for i in range(1000):
9         total = total + i
10        end = time.time()
11        delta_time = end - start
12        return delta_time
13
14 how_long = timed_function()
15 print(how_long)
16
17
18
```

ADD ANOTHER
BREAKPOINT

> CALL STACK

PAUSED ON STEP

✗ BREAKPOINTS

Raised Exceptions

Uncaught Exceptions

CONTINUE

VERSION CONTROL

© T.A. BINKOWSKI, 2020

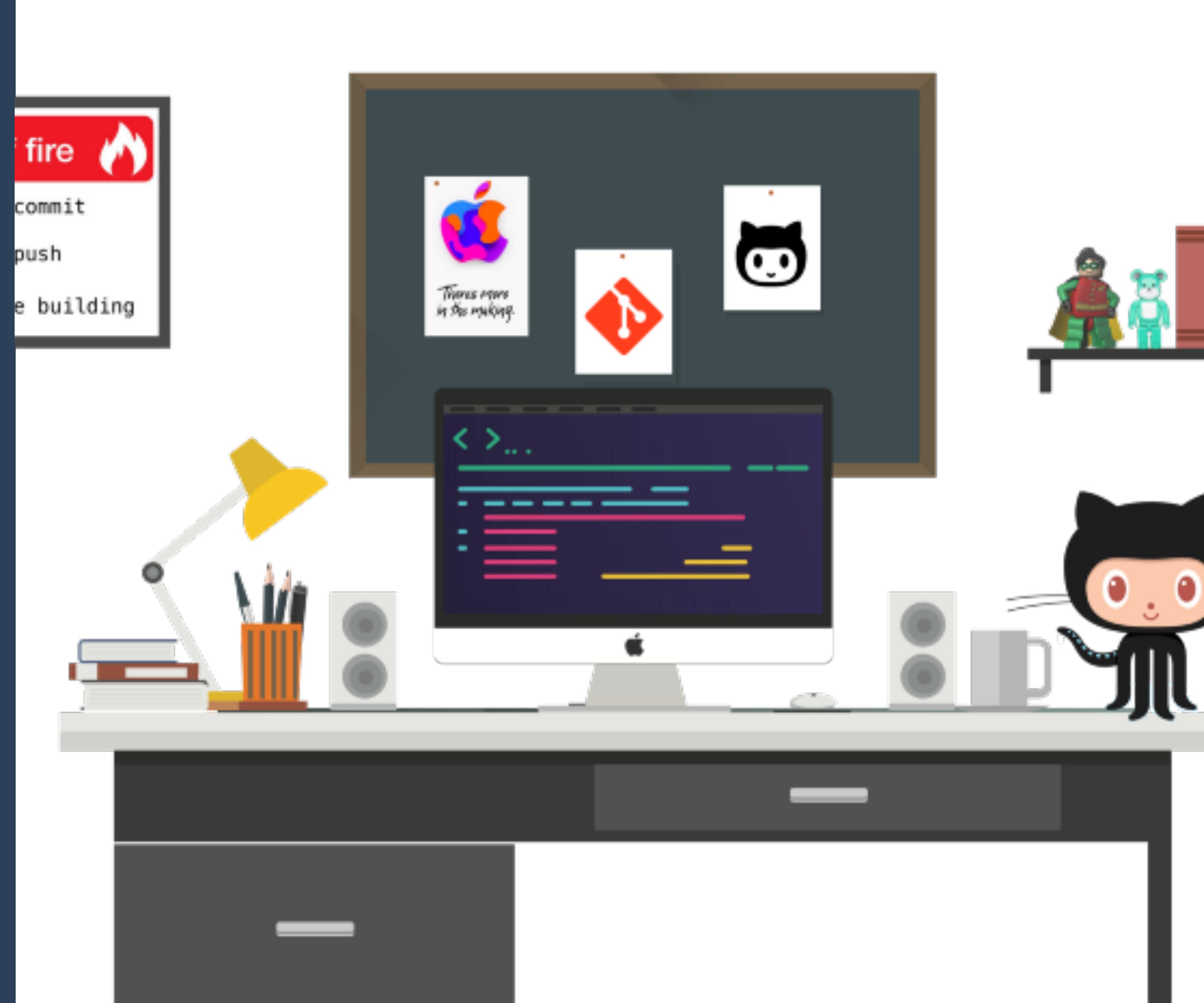
MODULE 3
MPCS 50101



THE UNIVERSITY OF
CHICAGO

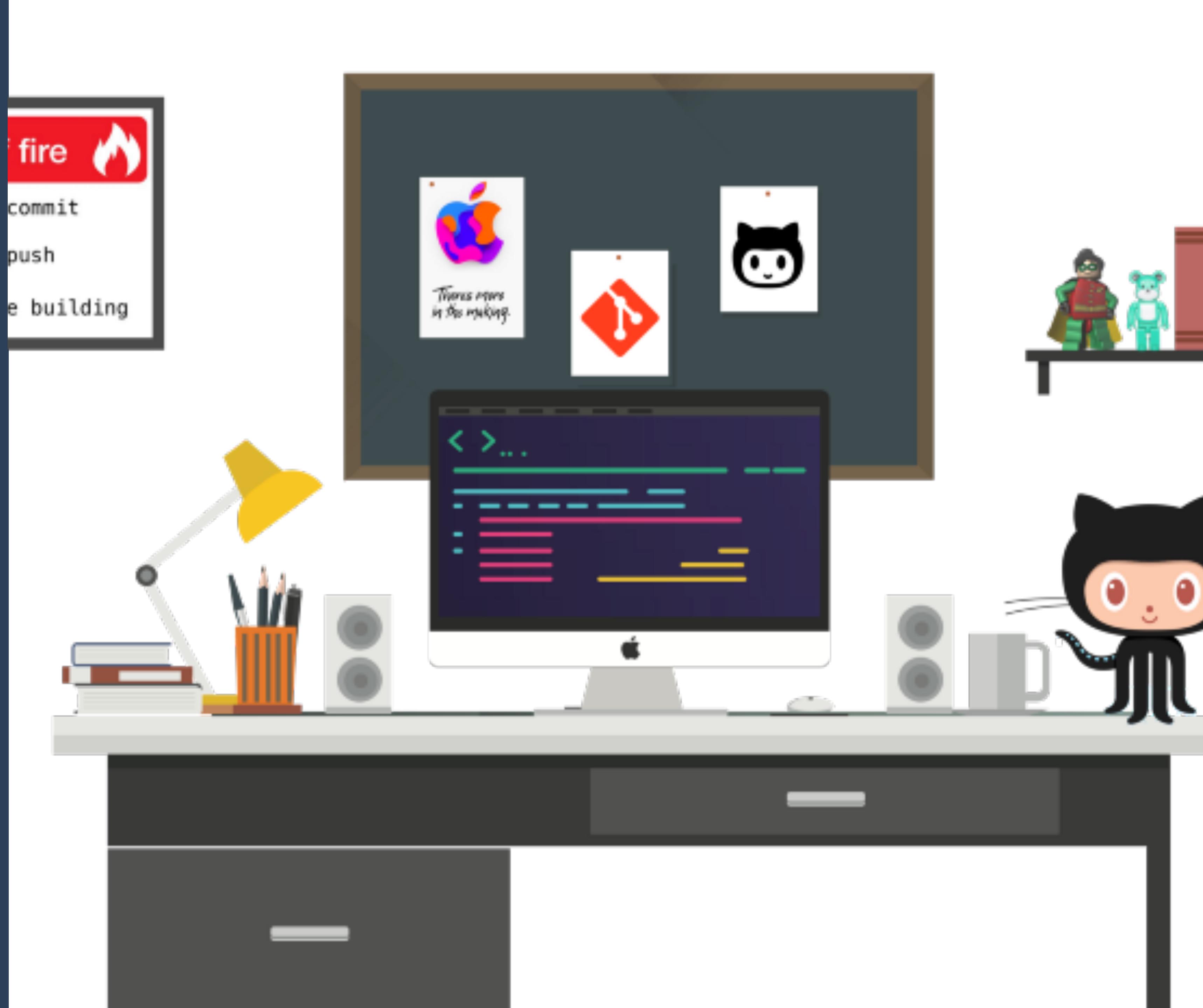
VERSION CONTROL

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later



VERSION CONTROL

- Track
- Blame
- Revert



GIT

GIT

FROM THE GUY
WHO BROUGHT
YOU LINUX



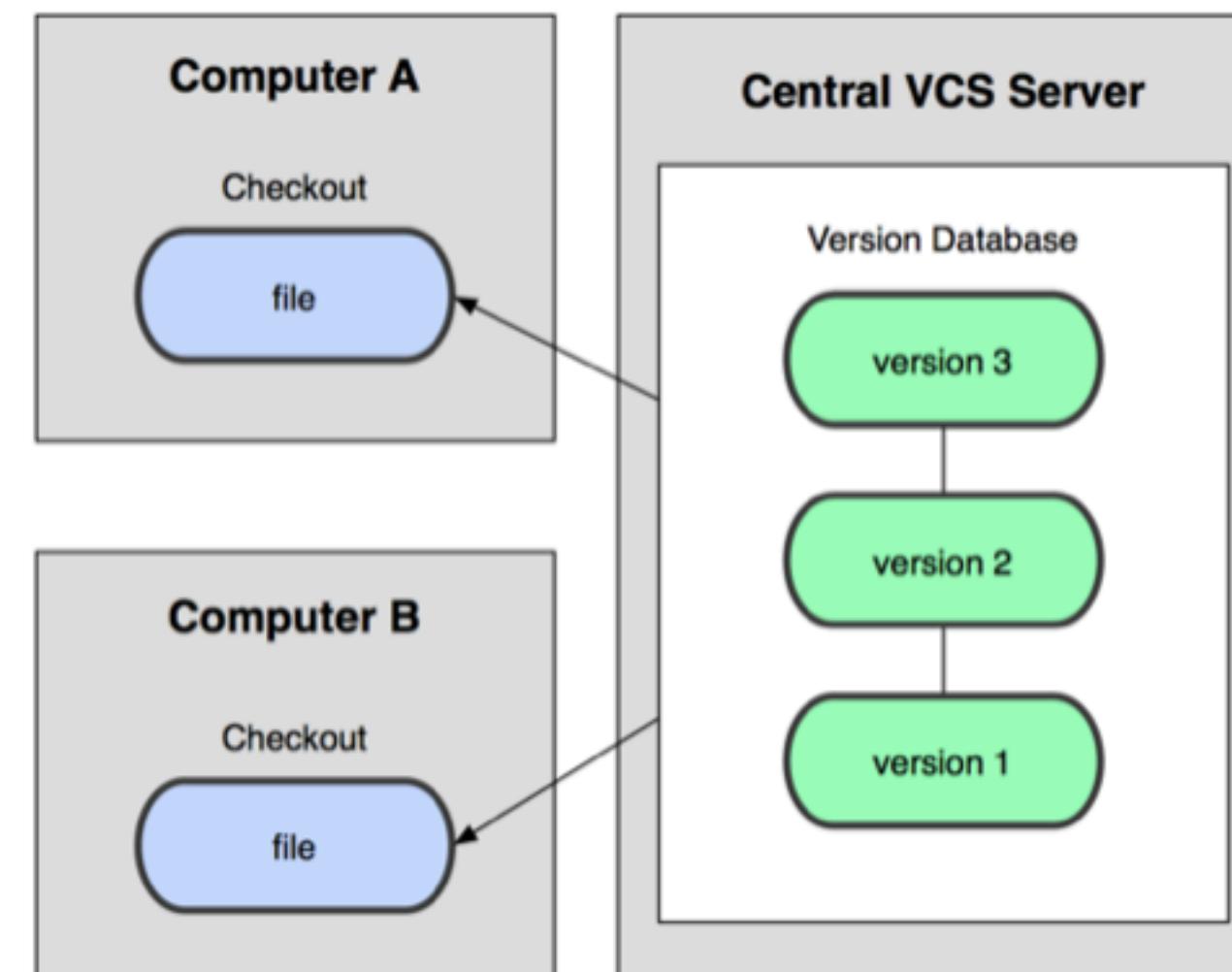
- Git is a free & open source, distributed version control system
 - Designed to handle everything from small to very large projects with speed and efficiency

GIT

- Every git repository contains the entire history of the project

SUBVERSION

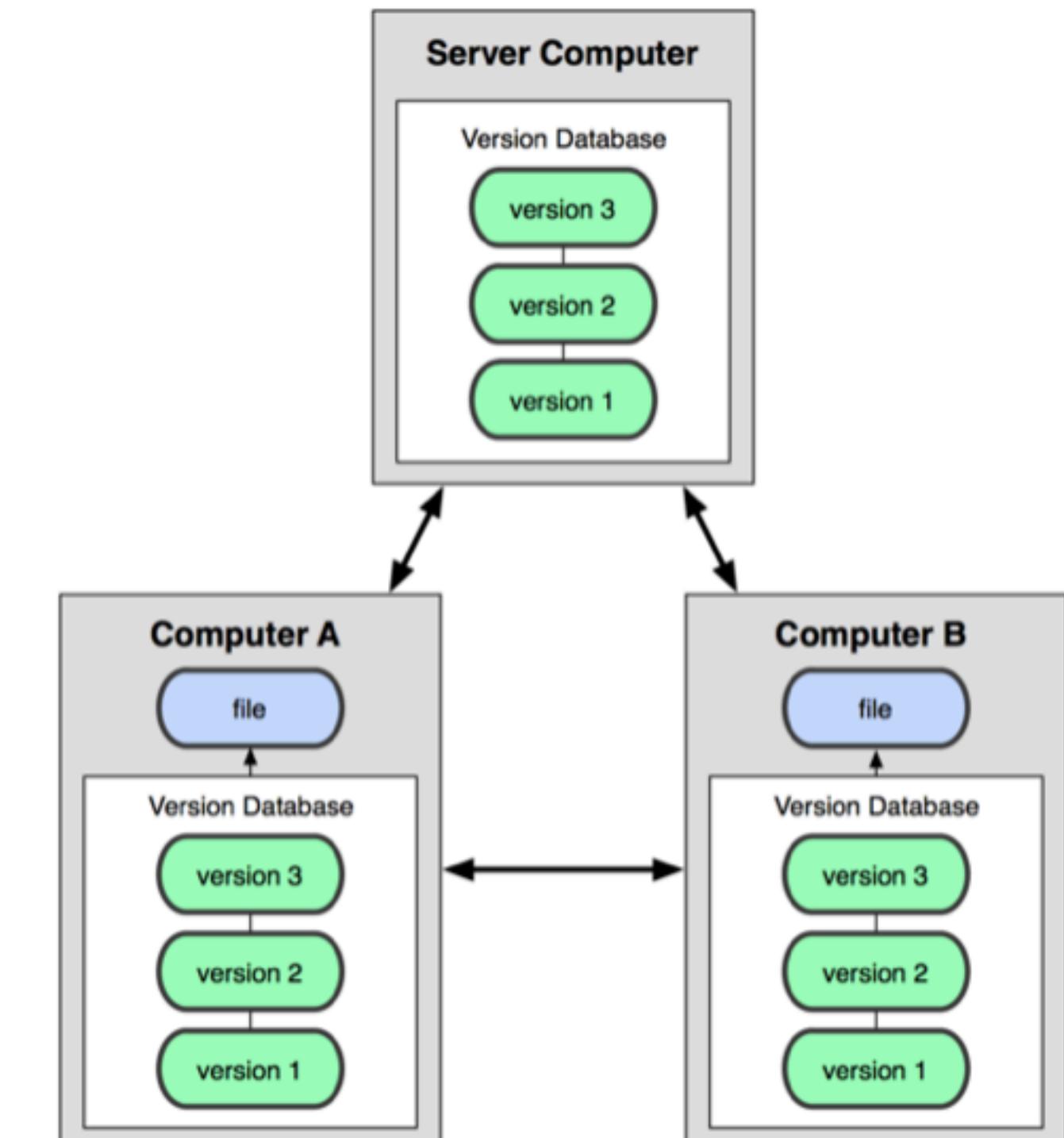
Centralized Model



(CVS, Subversion, Perforce)

GIT,
MERCURIAL

Distributed Model

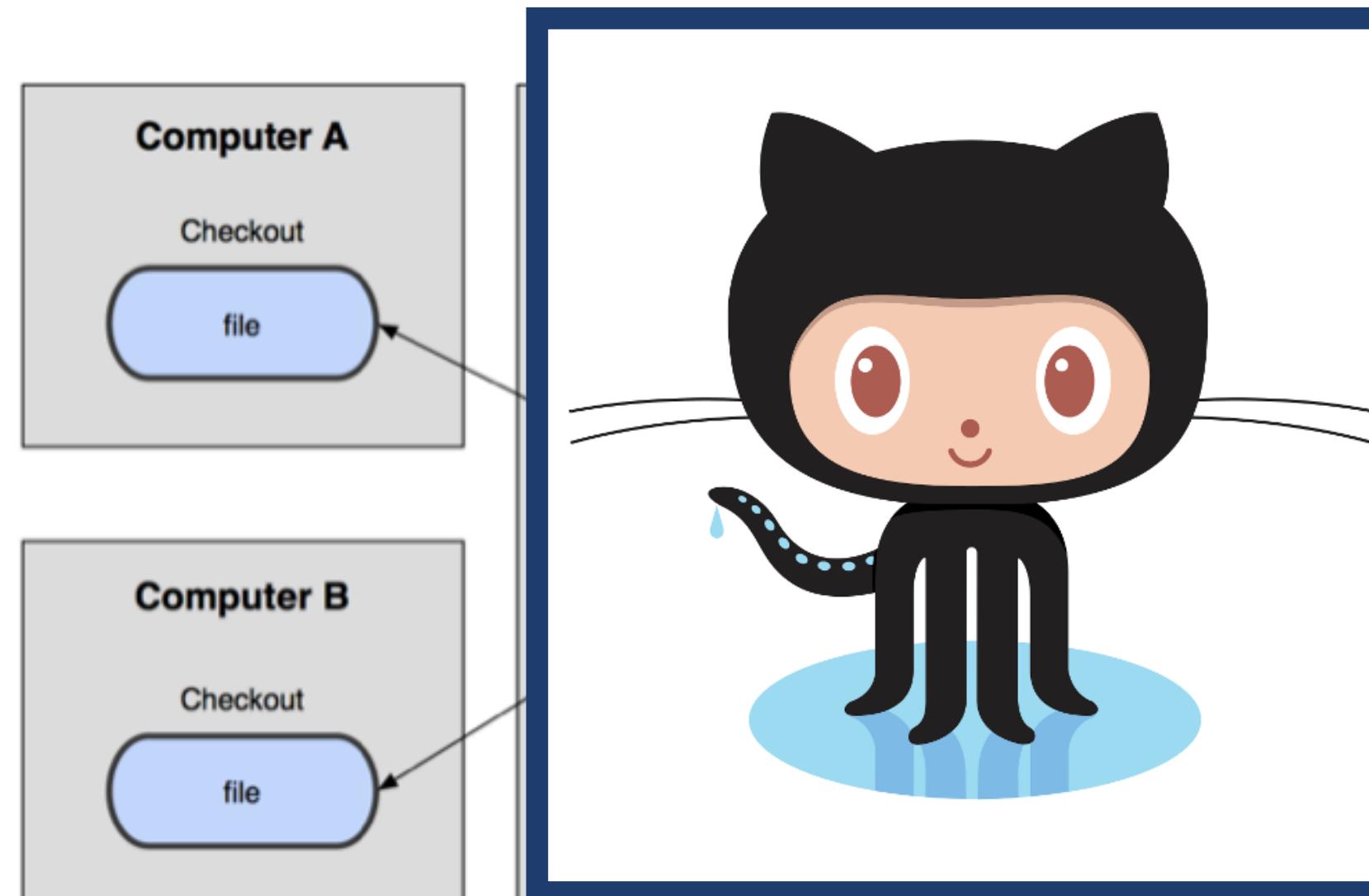


GIT

- In practice the way people use Github makes git more like centralized model
- Does not really matter

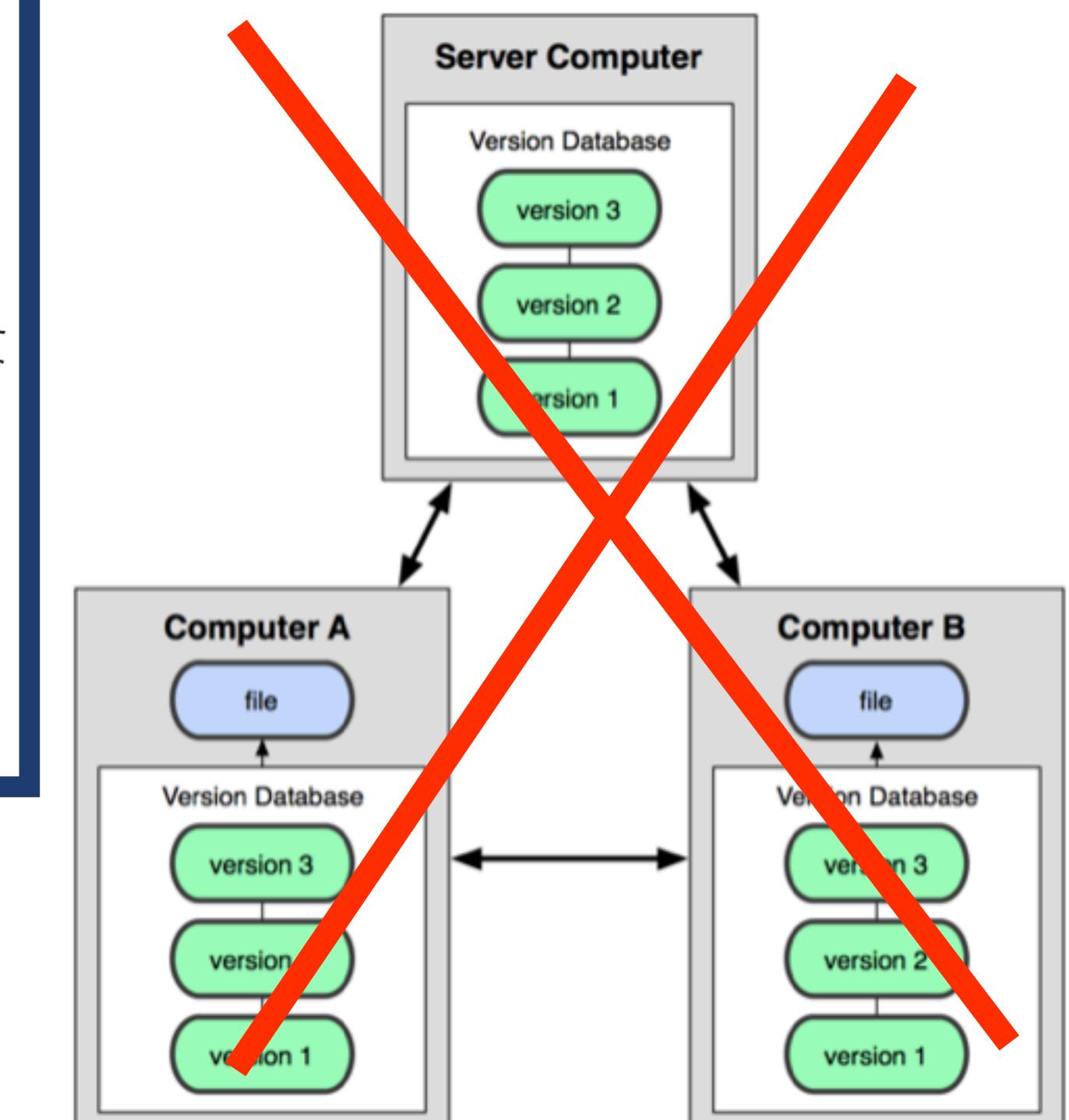
GIT,
MERCURIAL

Centralized Model



(CVS, Subversion, Perforce)

Distributed Model



GIT

- Use Git from the command line or Github Desktop
 - Same thing behind the scenes
 - Start with Desktop (please)

The screenshot shows the GitHub Desktop application interface. At the top, it displays the current branch as "esc-pr" and the pull request number "#3972" with a green checkmark. To the right, there's a "Fetch origin" button with a refresh icon and the text "Last fetched 3 minutes ago". The main area shows a pull request titled "Add event handler to dropdown component". The commit message includes "Co-Authored-By: Markus Olsson <niik@users.noreply.github.com>". The code diff shows changes made to the file "app/src/ui/t.../dropdown.tsx". The changes are color-coded: lines 145-151 are in a yellow box at the top, lines 148-151 are in a green box in the middle, and lines 148-154 are in a blue box at the bottom. The code itself includes imports like "React.Component<" and "React.KeyboardEvent<HTMLElement>". Lines 145, 146, and 147 show standard code, while lines 148 through 154 introduce a private method "isOpen" and its implementation.

```
@@ -145,6 +145,10 @@ export class React.Component<
    this.state = { clientRect
  }
}

146 146   }
147 147   }

148 + private get isOpen() {
149 +   return this.props.dropdown
150 + }
151 +
152 private dropdownIcon(state: boolean {
153   // @TODO: Remake triangle
154   // right now it's scaled
155 }

@@ -249,6 +253,13 @@ export class React.Component<
  }
  }
  }

249 253   }
250 254   }
251 255

256 + private onFoldoutKeyDown = React.KeyboardEvent<HTMLElement>
257 + if (!event.defaultPrevent
258   event.key === 'Escape') {
259     event.preventDefault()
260     this.props.onDropdownSt
ard')
```

GIT

```
+++ b/SplashScreen.xcodeproj/project.pbxproj
@@ -17,6 +17,14 @@
    37A8E0E51419A8BC00BF37A6 /* MainWindow_iPhone.xib in Resources */ = {isa = PBXBuildFile;
    37A8E0E91419A8BC00BF37A6 /* SplashScreenAppDelegate_iPad.m in Sources */ = {isa = PBXBuildFile;
    37A8E0EC1419A8BC00BF37A6 /* MainWindow_iPad.xib in Resources */ = {isa = PBXBuildFile; fileRef = 37A8E1081419C0D700BF37A6, fileUUID = 37A8E1081419C0D700BF37A6};
+   37A8E1081419C0D700BF37A6 /* Default-Landscape-ipad.png in Resources */ = {isa = PBXBuildFile; fileRef = 37A8E10A1419C0E000BF37A6, fileUUID = 37A8E10A1419C0E000BF37A6};
+   37A8E10A1419C0E000BF37A6 /* Default-Portrait-ipad.png in Resources */ = {isa = PBXBuildFile; fileRef = 37A8E1101419C70300BF37A6, fileUUID = 37A8E1101419C70300BF37A6};
+   37A8E1101419C70300BF37A6 /* Default.png in Resources */ = {isa = PBXBuildFile; fileRef = 37A8E1121419C70700BF37A6, fileUUID = 37A8E1121419C70700BF37A6};
+   37A8E1121419C70700BF37A6 /* Default@2x.png in Resources */ = {isa = PBXBuildFile; fileRef = 37A8E1201419C96100BF37A6, fileUUID = 37A8E1201419C96100BF37A6};
+   37A8E1201419C96100BF37A6 /* UCSplashScreen.m in Sources */ = {isa = PBXBuildFile; fileRef = 37A8E13D1419E58400BF37A6, fileUUID = 37A8E13D1419E58400BF37A6};
+   37A8E13D1419E58400BF37A6 /* Home_iPhone.m in Sources */ = {isa = PBXBuildFile; fileRef = 37A8E13E1419E58400BF37A6, fileUUID = 37A8E13E1419E58400BF37A6};
+   37A8E13E1419E58400BF37A6 /* Home_iPhone.xib in Resources */ = {isa = PBXBuildFile; fileRef = 37A8E15F141A65F000BF37A6, fileUUID = 37A8E15F141A65F000BF37A6};
+   37A8E15F141A65F000BF37A6 /* Home_iPad.m in Sources */ = {isa = PBXBuildFile; fileRef = 37A8E15F141A65F000BF37A6, fileUUID = 37A8E15F141A65F000BF37A6};
/* End PBXBuildFile section */

/* Begin PBXFileReference section */
@@ -24,18 +32,30 @@
    37A8E0CD1419A8BC00BF37A6 /* UIKit.framework */ = {isa = PBXFileReference; lastKnownFileType = framework.framework, path = /System/Library/Frameworks/UIKit.framework};
    37A8E0CF1419A8BC00BF37A6 /* Foundation.framework */ = {isa = PBXFileReference; lastKnownFileType = framework.framework, path = /System/Library/Frameworks/Foundation.framework};
    37A8E0D11419A8BC00BF37A6 /* CoreGraphics.framework */ = {isa = PBXFileReference; lastKnownFileType = framework.framework, path = /System/Library/Frameworks/CoreGraphics.framework};
-   37A8E0D51419A8BC00BF37A6 /* SplashScreen-Info.plist */ = {isa = PBXFileReference; path = SplashScreen-Info.plist, lastKnownFileType = text.plist};
+   37A8E0D51419A8BC00BF37A6 /* SplashScreen-Info.plist */ = {isa = PBXFileReference; lastKnownFileType = text.plist, path = SplashScreen-Info.plist};
    37A8E0D71419A8BC00BF37A6 /* en */ = {isa = PBXFileReference; lastKnownFileType = text.plist, path = en};
-   37A8E0D91419A8BC00BF37A6 /* SplashScreen-Prefix.pch */ = {isa = PBXFileReference; path = SplashScreen-Prefix.pch, lastKnownFileType = precompiledHeader.file};
+   37A8E0D91419A8BC00BF37A6 /* SplashScreen-Prefix.pch */ = {isa = PBXFileReference; lastKnownFileType = precompiledHeader.file, path = SplashScreen-Prefix.pch};
    37A8E0DA1419A8BC00BF37A6 /* main.m */ = {isa = PBXFileReference; lastKnownFileType = sourcecode.c.objc, path = main.m};
-   37A8E0DC1419A8BC00BF37A6 /* SplashScreenAppDelegate.h */ = {isa = PBXFileReference; path = SplashScreenAppDelegate.h, lastKnownFileType = headerfile};

```

GIT

- git init
- git add my_file
- git commit
- git status

```
git-practice — bash — 62x22
...n/mpcs50101-2019-autumn-code — bash ... | ...n-sessions/session4/git-practice — bash | /usr/local/bin — bash

tabinkowski:session4/git-practice
540 % git init
Initialized empty Git repository in /Users/tabinkowski/Goo
Drive/g-Teaching/uchicago.codes/uchicago.codes-courses/mpc
01/mpcs50101-2019-autumn/mpcs50101-2019-autumn-sessions/se
n4/git-practice/.git/
tabinkowski:session4/git-practice
541 % touch file1.txt
tabinkowski:session4/git-practice
542 % git add file1.txt
tabinkowski:session4/git-practice
543 % git commit -m "Add file1 to repository"
[master (root-commit) 114efe0] Add file1 to repository
  1 file changed, 0 insertions(+), 0 deletions(-)
   create mode 100644 file1.txt
tabinkowski:session4/git-practice (master)
544 % git status
On branch master
nothing to commit, working tree clean
tabinkowski:session4/git-practice (master)
545 %
```

GIT

- git log
- git status
- git checkout

```
git-practice — bash — 68x22
n pcs50101-2019-autumn-code — bash ... ...utumn-sessions/session4/git-practice — bash /usr/local/bin — bash

abinkowski:session4/git-practice (master)
  git log
commit 114efe08742be3a50d4e47a33f3b4286fe733c7e (HEAD -> master)
Author: tabinks <abinkowski@uchicago.edu>
Date:   Wed Oct 9 21:08:30 2019 -0500

  add file1 to repository
abinkowski:session4/git-practice (master)
```

GIT

- git status
- git add
- git commit

```
git-practice -- bash -- 68x22
n pcs50101-2019-autumn-code -- bash ... ...utumn-sessions/session4/git-practice -- bash /usr/local/bin -- bash

Changes added to commit (use "git add" and/or "git commit -a"
akowski:session4/git-practice (master)
  git status
* branch master
  nothing to commit, working directory clean
  changes not staged for commit:
    (use "git add <file>..." to update what will be committed)
    (use "git checkout -- <file>..." to discard changes in working directory)

modified:   file1.txt

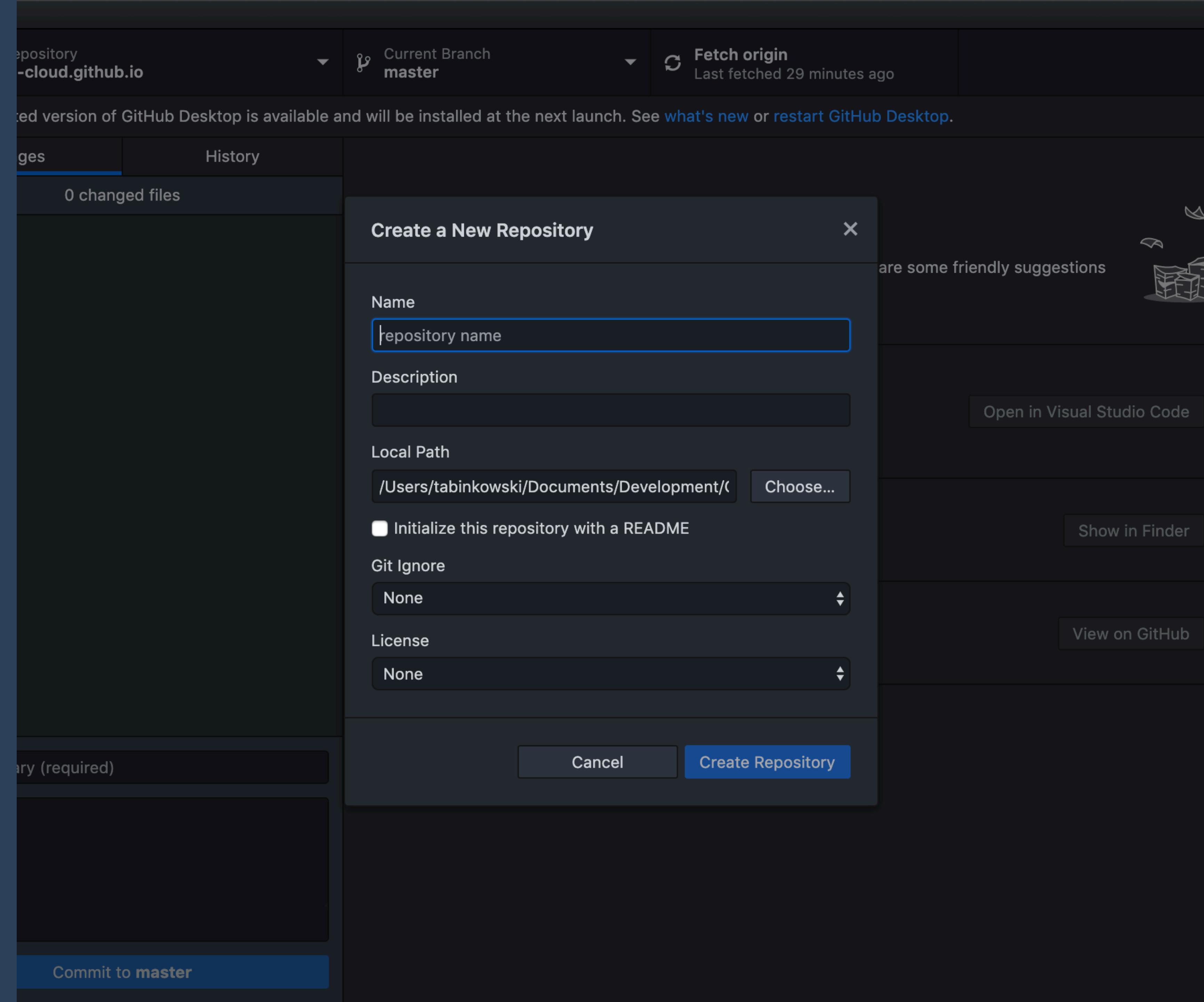
Changes added to commit (use "git add" and/or "git commit -a"
akowski:session4/git-practice (master)
  git add file1.txt
akowski:session4/git-practice (master)
560 % git commit -m "Edit file1."
[master 305ee28] Edit file1.
  1 file changed, 1 insertion(+)

akowski:session4/git-practice (master)
```

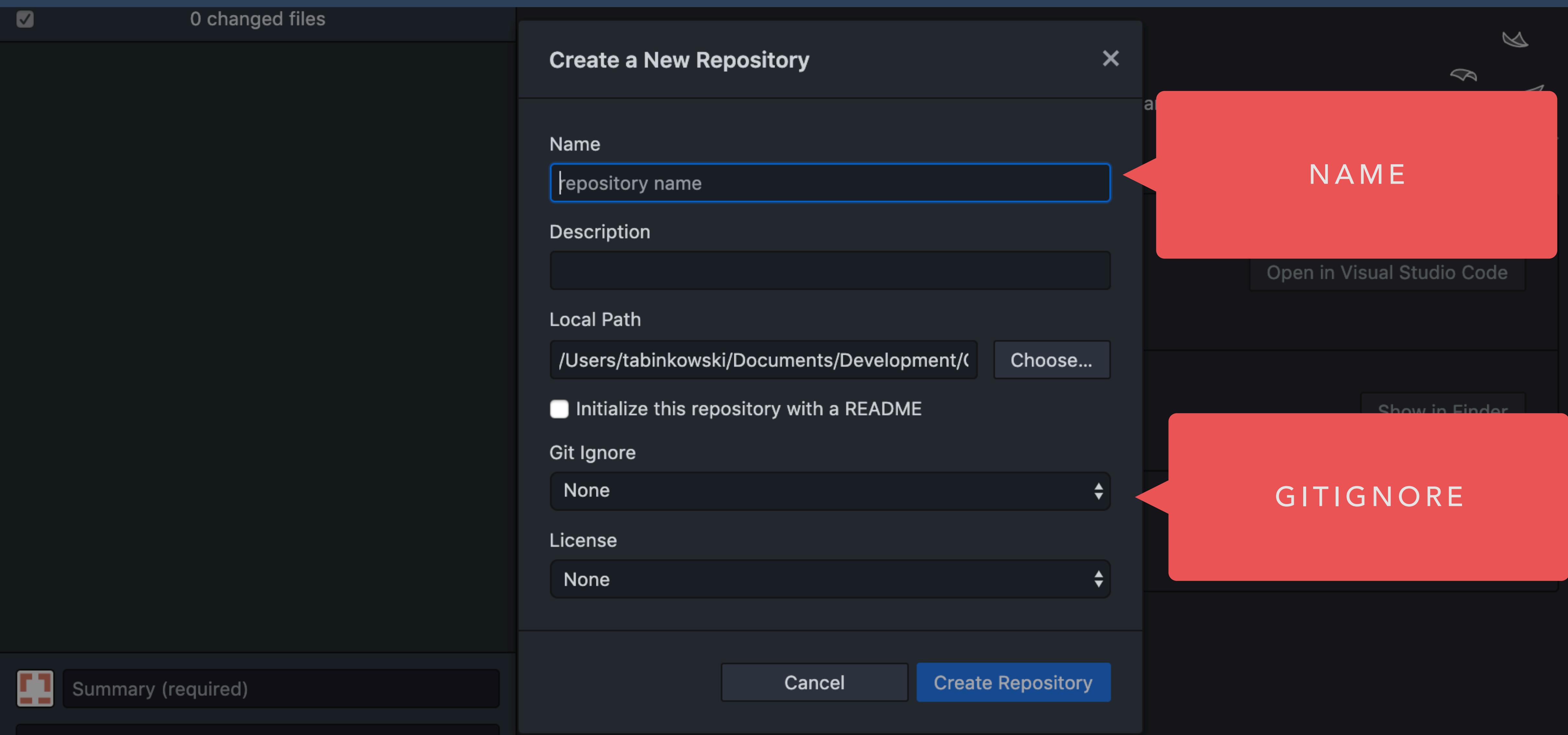
**GITHUB
DESKTOP**

GITHUB DESKTOP

- Application that performs the most common git commands
- Visual change tracker



GITHUB DESKTOP



GITHUB DESKTOP

PUBLISH
(CAN USE WITHOUT
GITHUB)

Current Repository **practice-repository**

Current Branch **master**

An updated version of GitHub Desktop is available and will be installed at the next launch. See [what's new](#) or [restart GitHub Desktop](#).

Changes History

0 changed files

NAME

BRANCH

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Publish your repository to GitHub

This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.

Always available in the toolbar for local repositories or  

Publish repository

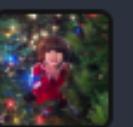
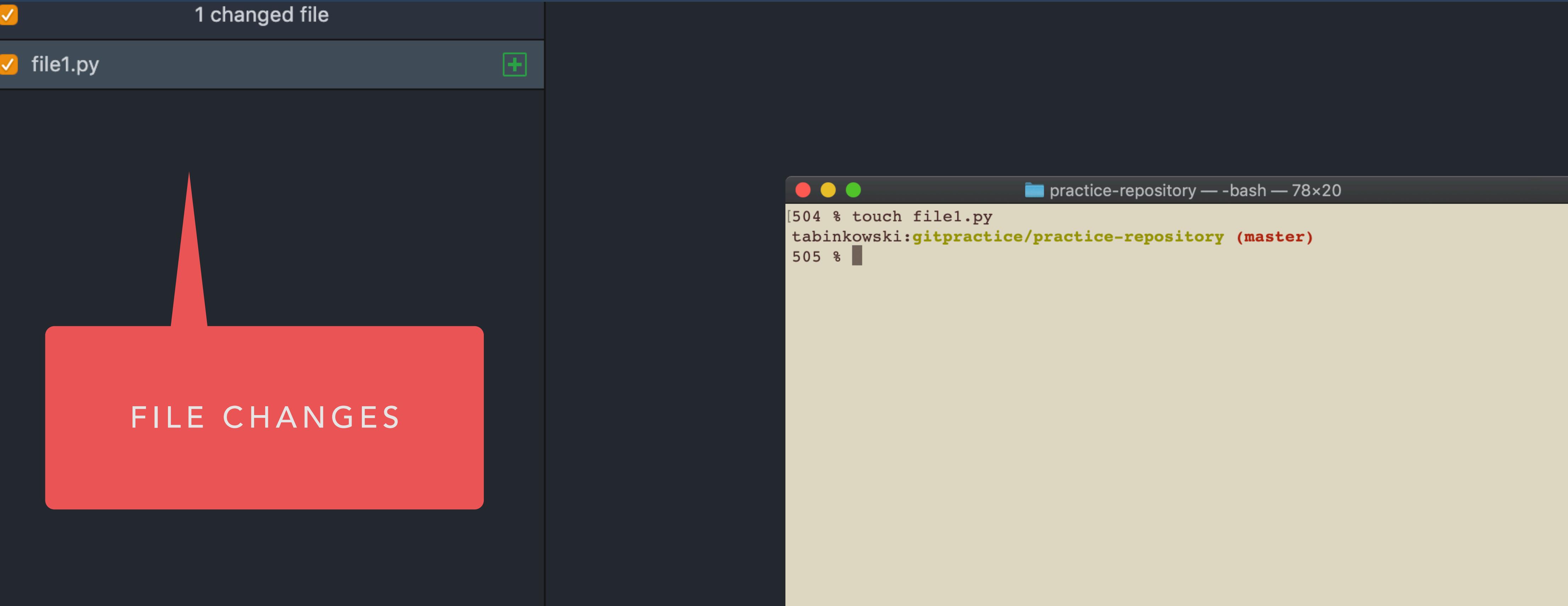
Open the repository in your external editor

Select your editor in [Preferences](#)

Repository menu or   

Open in Visual Studio Code

GITHUB DESKTOP



Create file1.py

Description

GITHUB DESKTOP

✓ 1 changed file

✓ file1.py +

SUGGESTED COMMIT MESSAGE

The file is empty

COMMIT TO
MASTER BRANCH
(DEFAULT)

Create file1.py

Description

GITHUB DESKTOP

The screenshot shows the GitHub Desktop application interface. On the left, there's a sidebar with a dark background containing four items:

- No Branches to Compare** (grey icon)
- Update file1.py.** (blue icon, highlighted)
- Create file1.py** (grey icon)
- Initial commit** (grey icon)

The main area has a dark background. At the top, it displays a commit message from **tabinks** committed **a837f5b**, **2 changed files**. Below this, the file list shows **file1.py** (marked with a yellow circle icon) and **file1.py~** (marked with a green plus sign icon). To the right, a diff view shows the content of **file1.py**:

```
@@ -0,0 +1 @@
1 +This is the first line.
```

A large red callout box with the text **HISTORY.** points to the commit message in the sidebar.

GITHUB CLASSROOM

GITHUB CLASSROOM

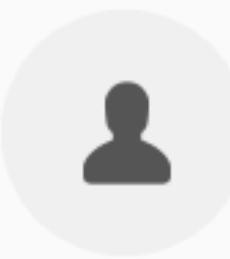


Accept the **mpcs50101-2019-autumn-assignment-1** assignment

Accepting this assignment will give you access to the **mpcs50101-2019-autumn-assignment-1-tabinks** repository in the [@uchicago-codes](#) organization on GitHub.

Accept this assignment

GITHUB CLASSROOM



Accepted the **mpcs50101-2019-autumn-assignment-1** assignment

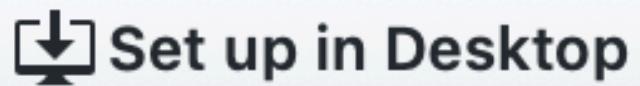
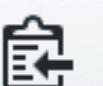
You are ready to go!

You may receive an invitation to join [@uchicago-codes](#) via email invitation on your behalf. No further action is necessary.

Your assignment has been created here: <https://github.com/uchicago-codes/mpcs50101-2019-autumn-assignment-1-tabinks>

GITHUB CLASSROOM

Quick setup — if you've done this kind of thing before

 Set up in Desktop or   <https://github.com/uchicago-codes/mpcs50101-2019-autumn-assignment-1-tabinks.git> 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# mpzs50101-2019-autumn-assignment-1-tabinks" >> README.md  
git init  
git add .  
git commit -m "Initial commit"  
git remote add origin https://github.com/uchicago-codes/mpzs50101-2019-autumn-assignment-1-tabinks.git  
git push -u origin master
```

USE THIS

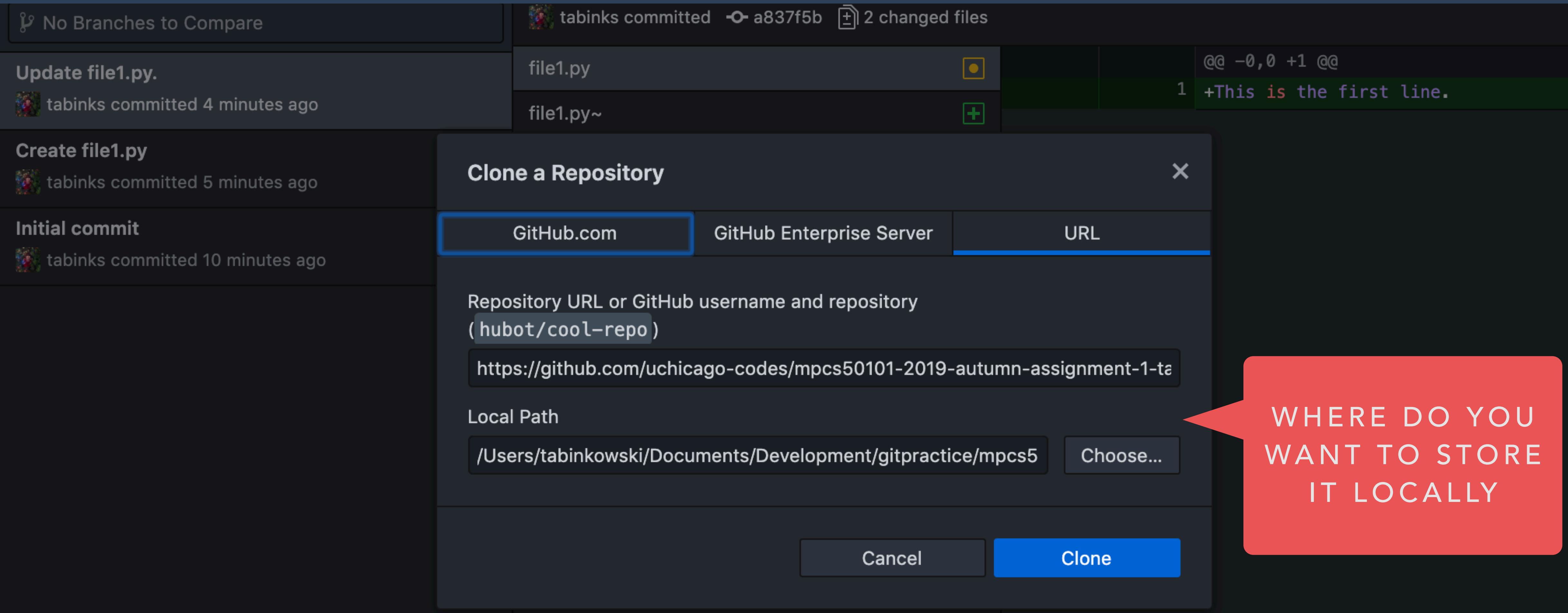


...or push an existing repository from the command line

```
git remote add origin https://github.com/uchicago-codes/mpzs50101-2019-autumn-assignment-1-tabinks.git
```



GITHUB CLASSROOM



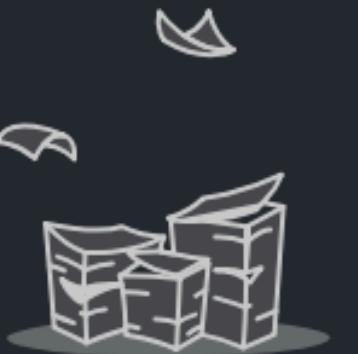
GITHUB CLASSROOM



0 changed files

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.



OPEN

Open the repository in your external editor

Select your editor in [Preferences](#)

Repository menu or ⌘ ⌘ A

lio Code

Show in Finder

View the files of your repository in Finder

Repository menu or ⌘ ⌘ F

View on GitHub

Open the repository page on GitHub in your browser

Repository menu or ⌘ ⌘ G



Summary (required)

Description

GITHUB CLASSROOM

A screenshot of a GitHub Classroom interface. At the top, it says "No Branches to Compare". Below that, a commit message "Create file1.py" is shown, followed by "tabinks committed just now". The main area shows a commit history for "file1.py": "tabinks committed 177b96a 1 changed file". The diff shows a single line added: "@@ -0,0 +1 @@ +print("Hello world")". A large red button labeled "OPEN" is overlaid in the center.

No Branches to Compare

Create file1.py

tabinks committed just now

tabinks committed 177b96a 1 changed file

file1.py

@@ -0,0 +1 @@
+print("Hello world")

OPEN

GITHUB CLASSROOM

Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH

<https://github.com/uchicago-codes/mpcs50101-2019-autumn-assignment-1-tabinks>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend using GitHub's desktop app.

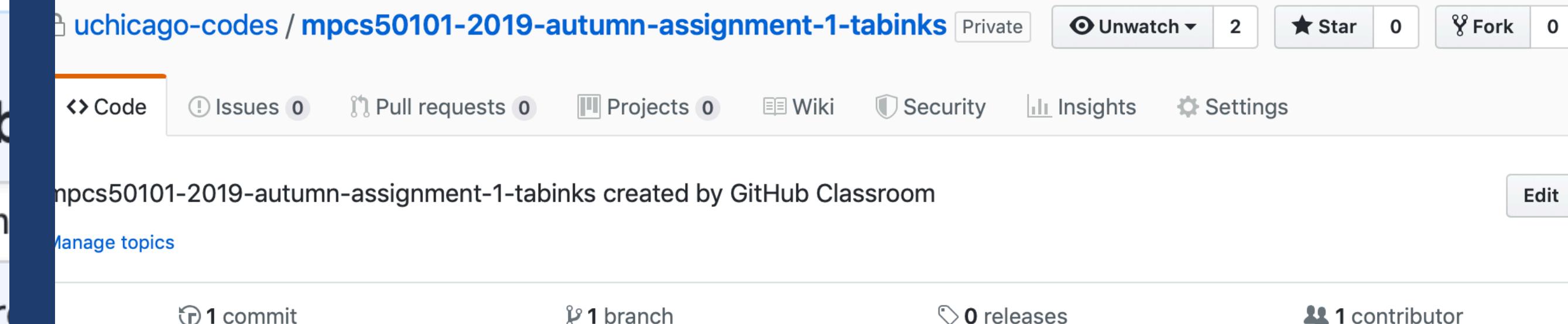
...or create a new repository on the command line

```
echo "# mpzs50101-2019-autumn-assignment-1-tabinks"
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/uchicago-codes/mpcs50101-2019-autumn-assignment-1-tabinks.git
git push -u origin master
```

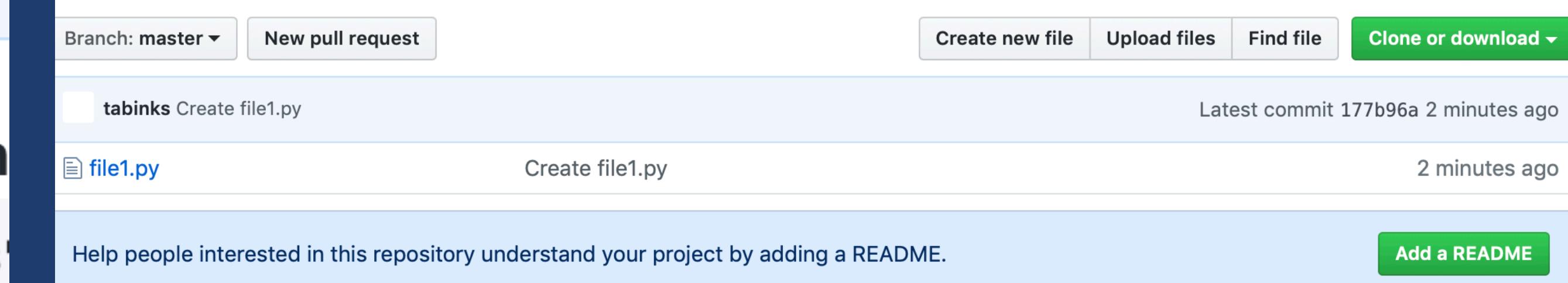
BEFORE

...or push an existing repository from the command line

```
git remote add origin https://github.com/uchicago-codes/mpcs50101-2019-autumn-assignment-1-tabinks
```



The screenshot shows a GitHub repository page for 'mpcs50101-2019-autumn-assignment-1-tabinks'. The repository is private and was created by GitHub Classroom. It has 1 commit, 1 branch, 0 releases, and 1 contributor. The commit history shows a file named 'file1.py' was created. The page includes standard GitHub navigation links like Code, Issues, Pull requests, Projects, Wiki, Security, Insights, and Settings.



The screenshot shows the same GitHub repository page after pushing changes. The commit history now shows two commits: one for 'Create file1.py' and another for 'Create file1.py'. The latest commit was made 2 minutes ago. A green button labeled 'Add a README' is visible in the top right corner.

AFTER

ASSIGNMENT

© T.A. BINKOWSKI, 2020

MODULE 4
MPCS 50101



THE UNIVERSITY OF
CHICAGO

ASSIGNMENT

- Module 4:
Assignment

For each problem, create a file name [problem1.py](#), [problem2.py](#), etc. Please make judicious use of comments and **docstrings** in your code. Use GitHub classroom to submit your work. We will only grade [master](#) branch.

Github Classroom Repository: <https://classroom.github.com/a/C81SLOqQ>

Enter the GitHub URL for your assignment to submit on Canvas.

Problem 1

Refactor you rock!

All the logic here is in the main module design. Anything related to running the main module should be part of the [__main__](#) module.

Keep in mind best practices in Python code. The code should be part of

VIDEO WALKTHROUGH IN MODULE

Problem 2

THE END

© T.A. BINKOWSKI, 2020

MODULE 4
MPCS 50101



THE UNIVERSITY OF
CHICAGO