CONCEPTS OF PROGRAMMING

MODULE 4
MPCS 50101

THE UNIVERSITY OF CHICAGO

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

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

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

`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 is 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. — Correct Answer
- [ ] A sets items can be accessed by index.
- [ ] The items in a set are ordered.

Question

In Python, `frozenset` is the same as `set` except that `forzenset`s are immutable (ie. elements from the `frozenset` cann
removed once created).

- ( ) True — Correct Answer
- ( ) False

Question 4

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

- [ ] s = {'a', 'b', 'c'} — Correct Answer
- [ ] s = set(['a', 'b', 'c']) — Correct Answer
- [ ] s = set('abc') — Correct Answer
- [ ] s = set(['a', 'b', 'c']) — Correct Answer
- [ ] s = {('a', 'b', 'c')}

LECTURE QUIZ

Question 5

Given the dictionary `x`:

```
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`:

```
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
- ○ False

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 a
bout telescopes.\n\t– Edsger W. Dijkstra"
string = string.rstrip()
string = string.split('c')
string = string[3].split()[2]
```

Correct Answers astronomy

BREAKOUT EXERCISES

MODULE 4
MPCS 50101

THE UNIVERSITY OF CHICAGO

BREAKOUT EXERCISES

Module 4: Breakout Exercises

EXAM PRACTICE

Exercise 1

In preparation for you 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

MODULE 4
MPCS 50101

THE UNIVERSITY OF CHICAGO

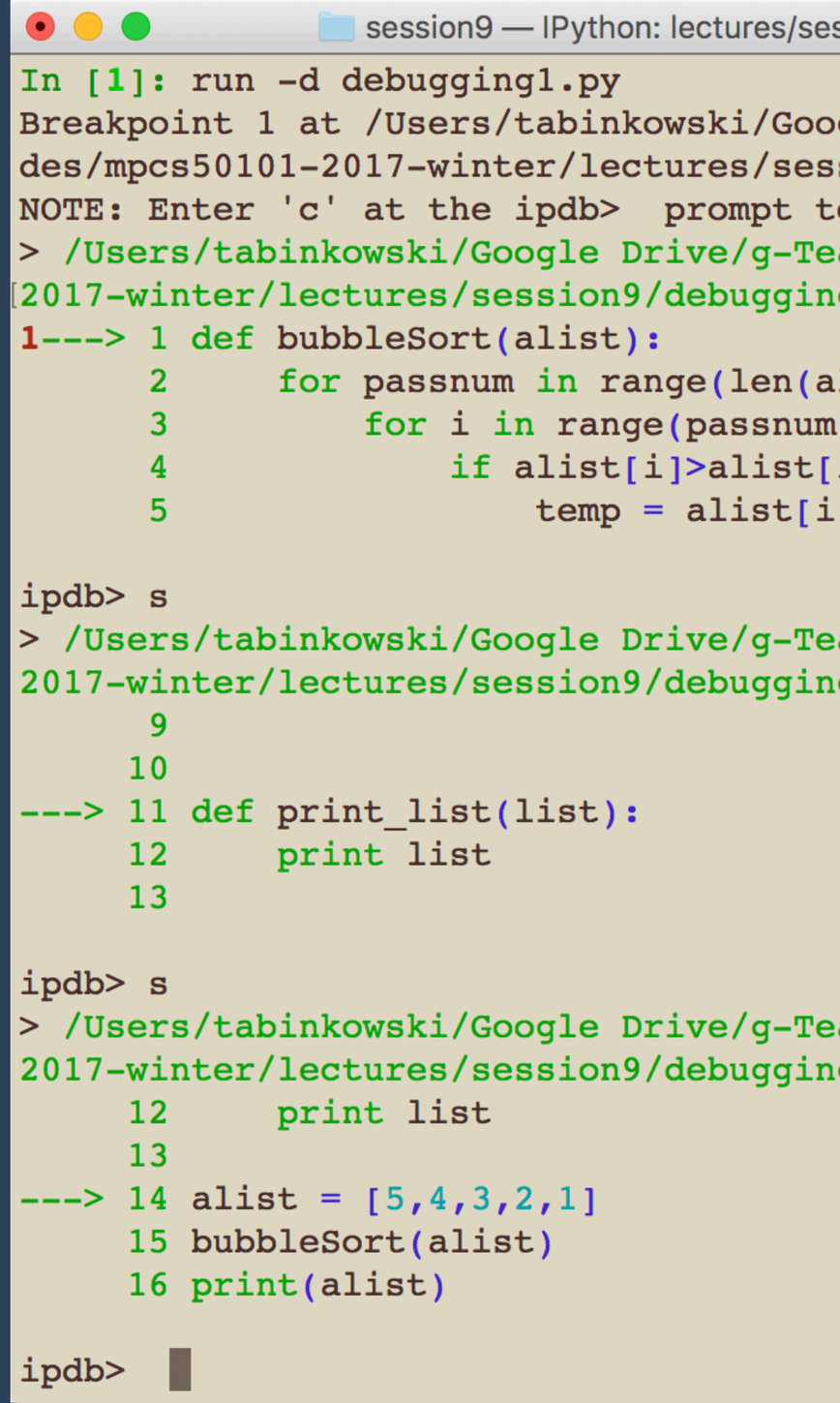
DEBUGGING

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...



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 - an interactive source code debugger for Python programs

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:

```
>>> 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)
```

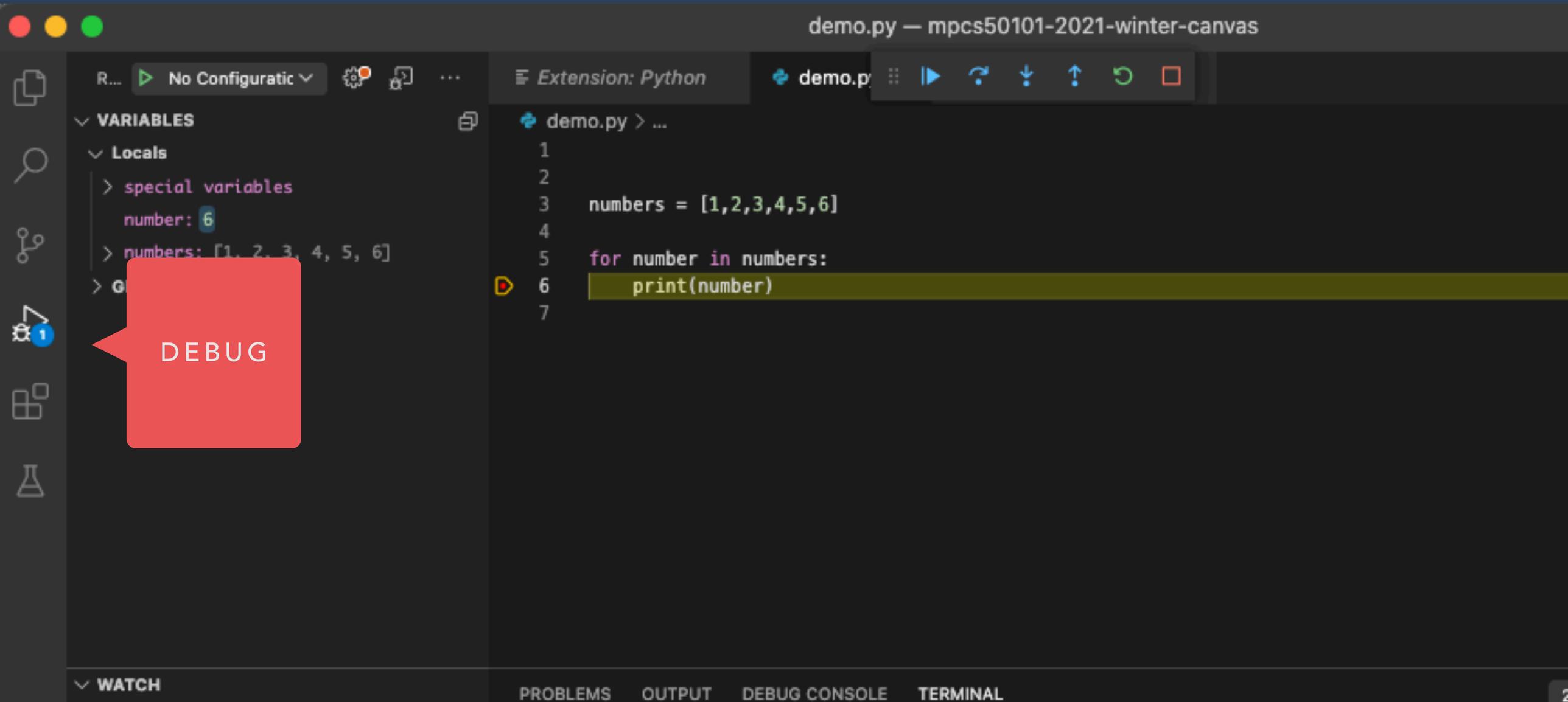
DEBUGGING WITH PDB

- Functionality is the same
- Usability is very different

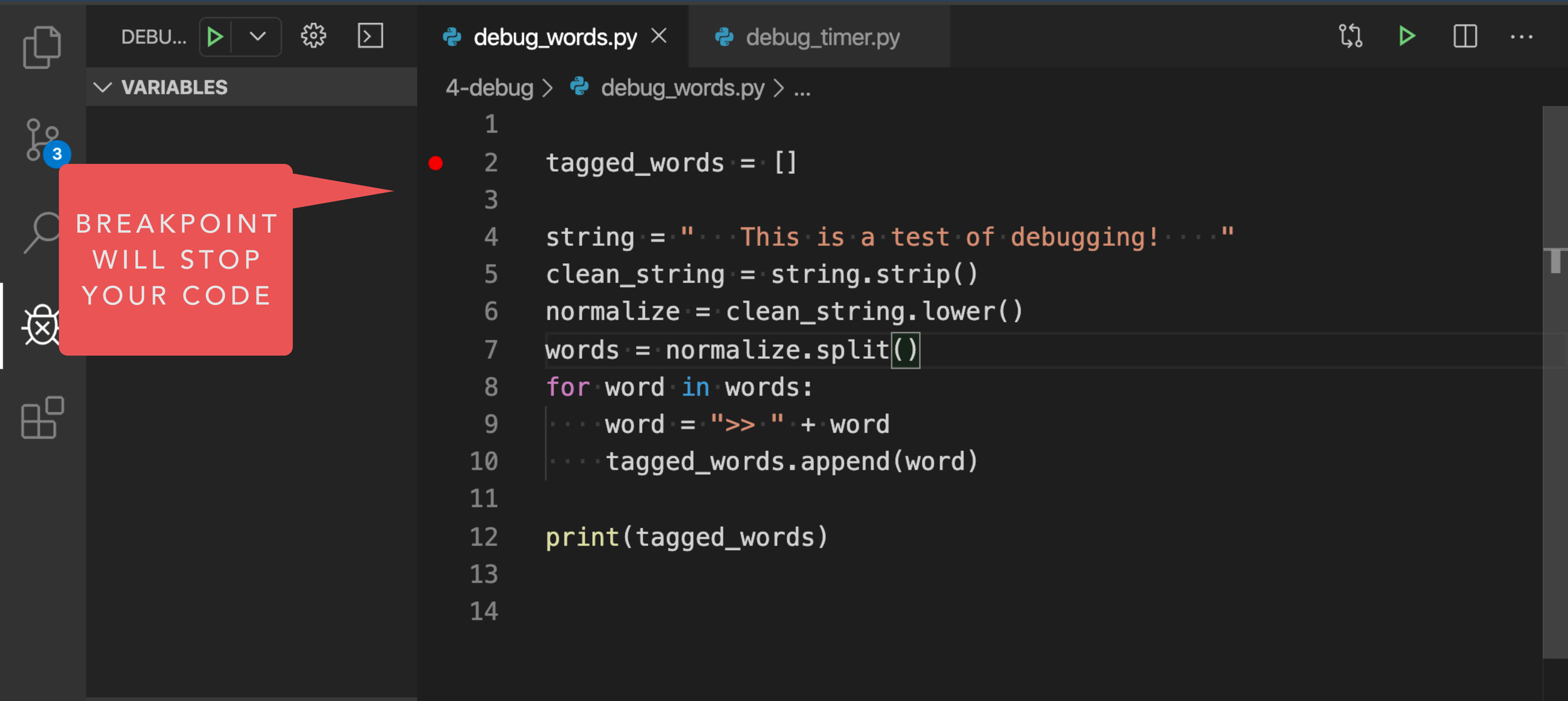
```
[483 % python debug_pdb.py
> /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs50101-
2017-winter/lectures/session9/debug_pdb.py(6)iter_hello()
-> for n in range(i):
(Pdb) l
  1
  2     import pdb
  3
  4     def iter_hello(i):
  5         pdb.set_trace()
  6  ->     for n in range(i):
  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 VSCODE

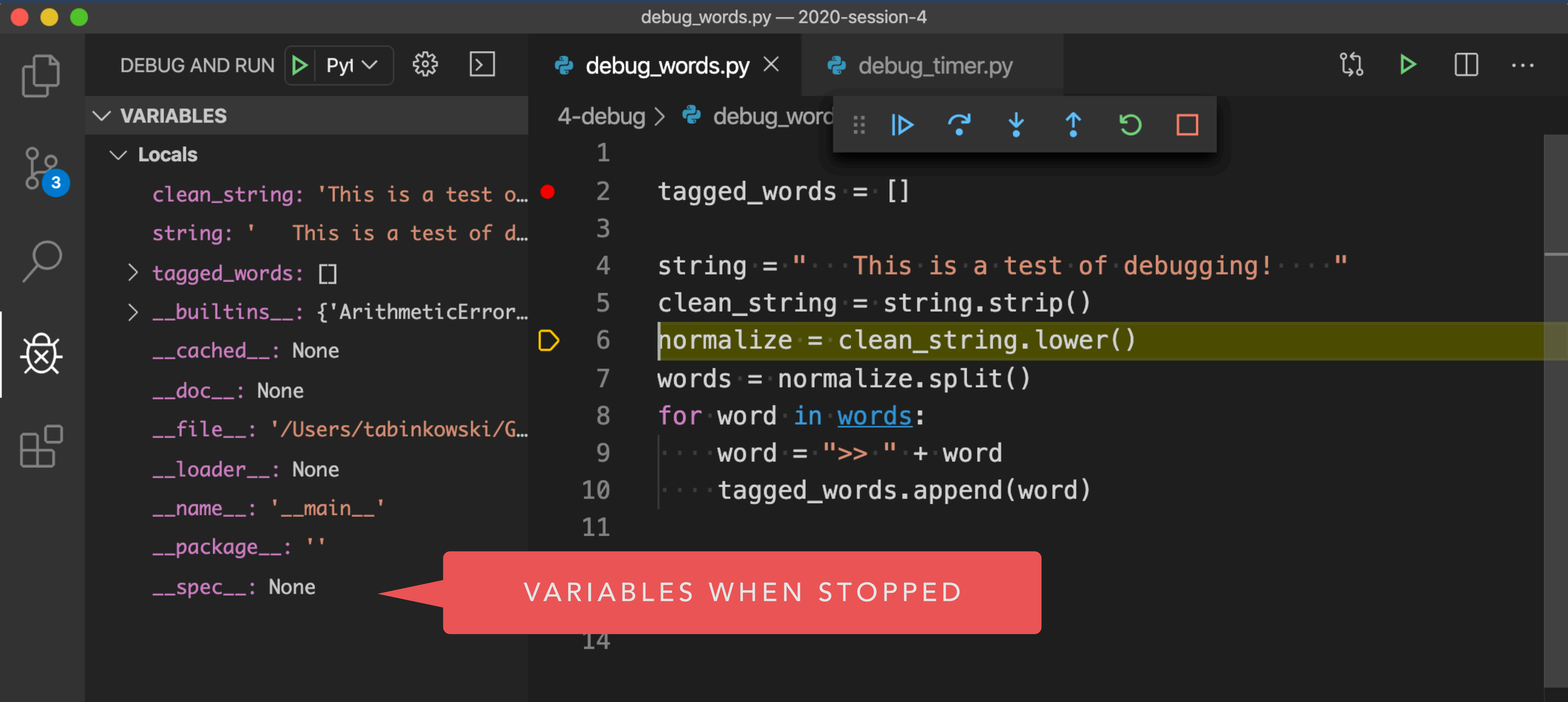
DEBUGGING


demo.py — mpcs50101-2021-winter-canvas
No Configuratic
Extension: Python
demo.py
VARIABLES
Locals
special variables
number: 6
numbers: [1, 2, 3, 4, 5, 6]
demo.py > ...
1
2
3 numbers = [1,2,3,4,5,6]
4
5 for number in numbers:
6 print(number)
7
DEBUG
WATCH
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

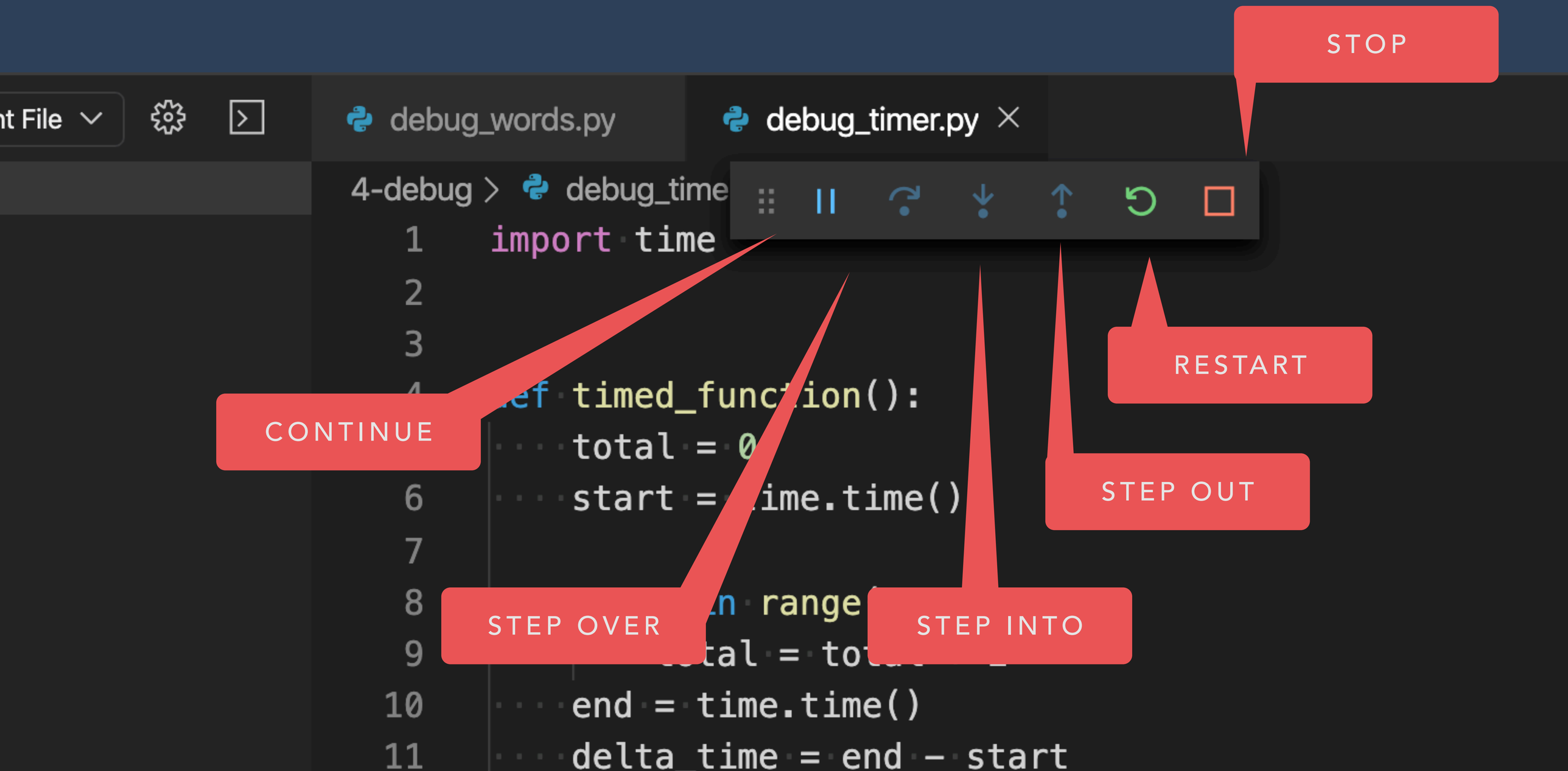
DEBUGGING


DEBU...
debug_words.py
debug_timer.py
VARIABLES
4-debug > debug_words.py > ...
BREAKPOINT WILL STOP YOUR 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

DEBUGGING

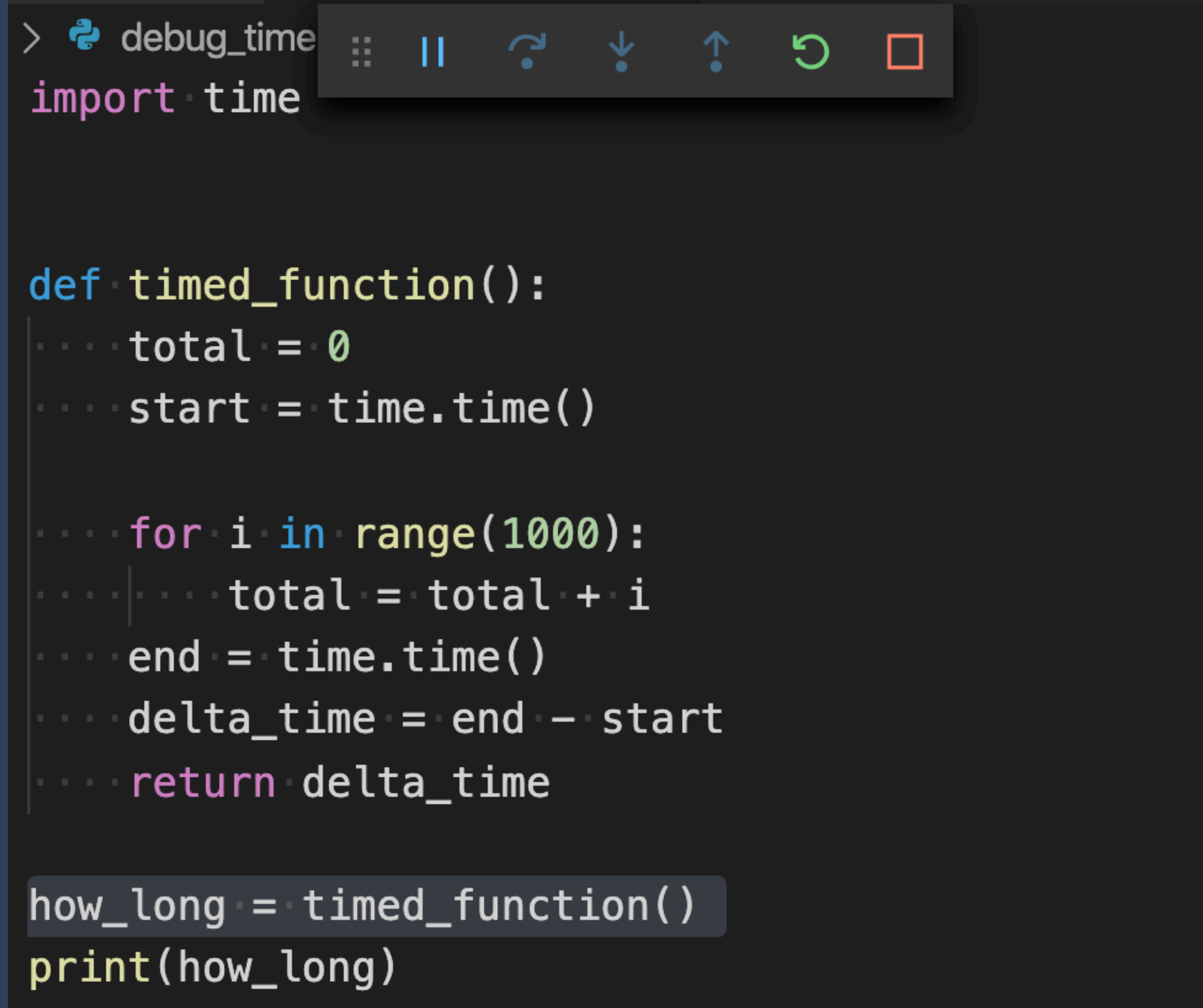

debug_words.py — 2020-session-4
DEBUG AND RUN
Pyt
debug_words.py
debug_timer.py
4-debug
debug_word
VARIABLES
Locals
clean_string: 'This is a test o…
string: ' This is a test of d…
tagged_words: []
__builtins__: {'ArithmeticError…
__cached__: None
__doc__: None
__file__: '/Users/tabinkowski/G…
__loader__: None
__name__: '__main__'
__package__: ''
__spec__: None
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
14
VARIABLES WHEN STOPPED

DEBUGGING

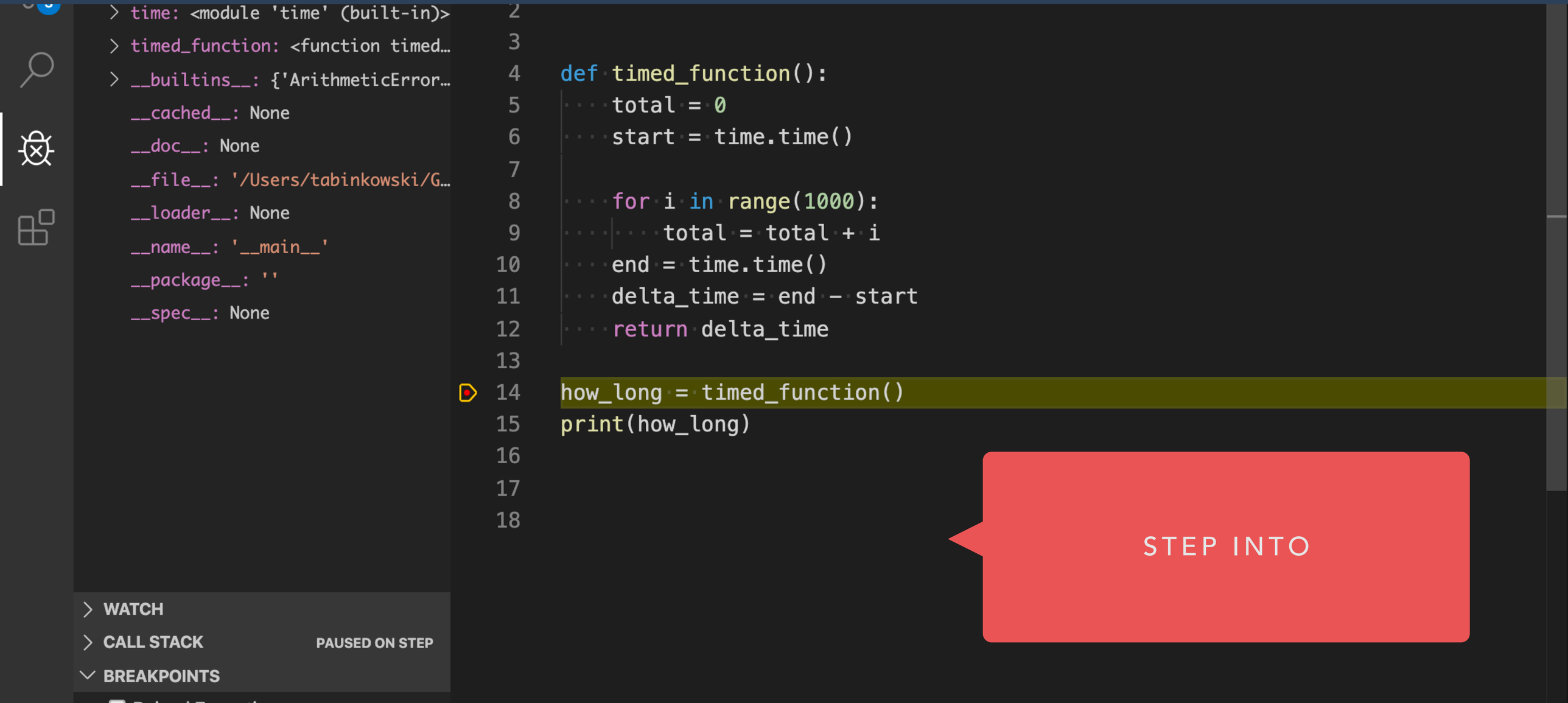

STOP
debug_words.py
debug_timer.py
4-debug > debug_time
1 import time
RESTART
CONTINUE
STEP OUT
STEP OVER
STEP INTO
10 end = time.time()
11 delta_time = end - start

DEBUGGING

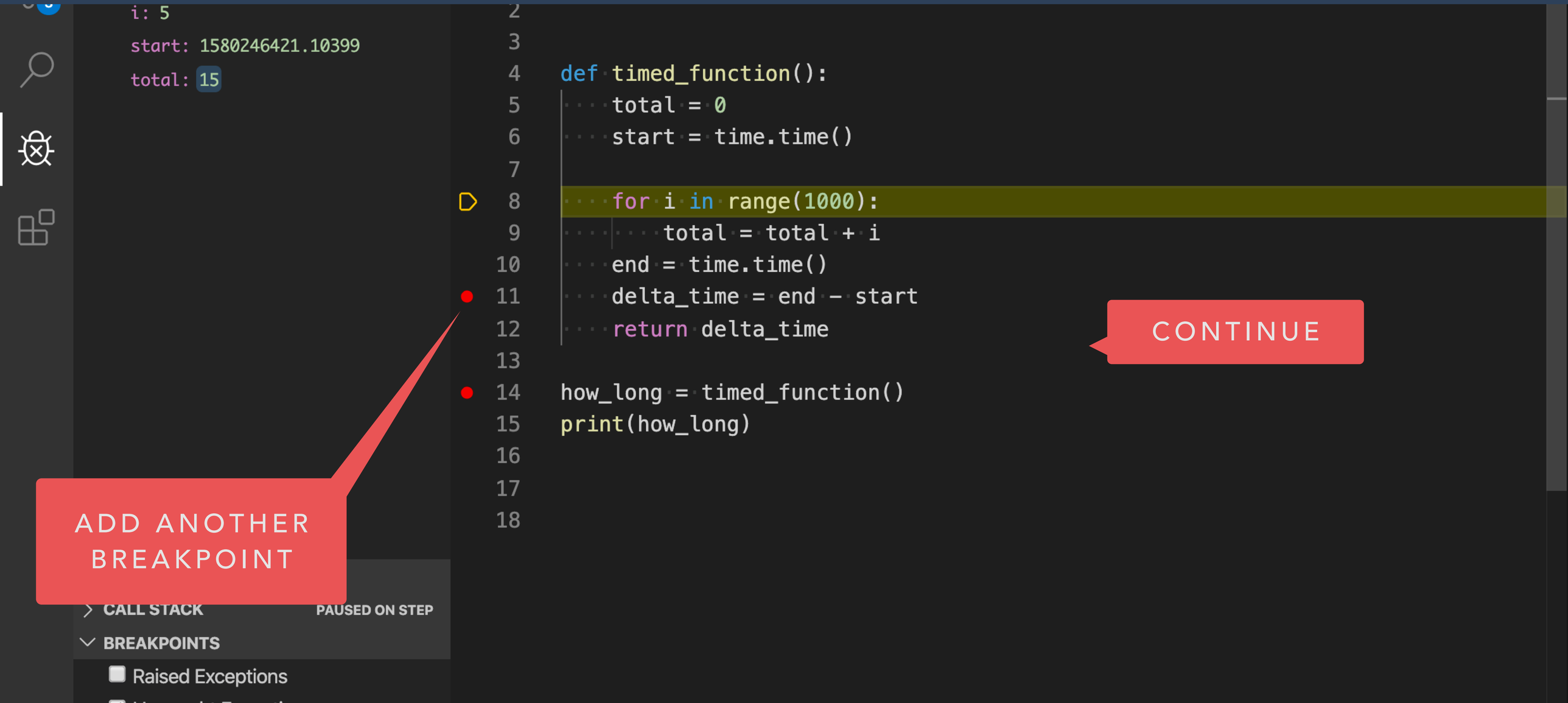
- Continue
- Step Over - next line
- Step Into - go into function
- Step Out - exit function
- Restart
- End



DEBUGGING


time: <module 'time' (built-in)>
timed_function: <function timed…
__builtins__: {'ArithmeticError…
__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
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)
CONTINUE
ADD ANOTHER BREAKPOINT
CALL STACK
PAUSED ON STEP
BREAKPOINTS
Raised Exceptions

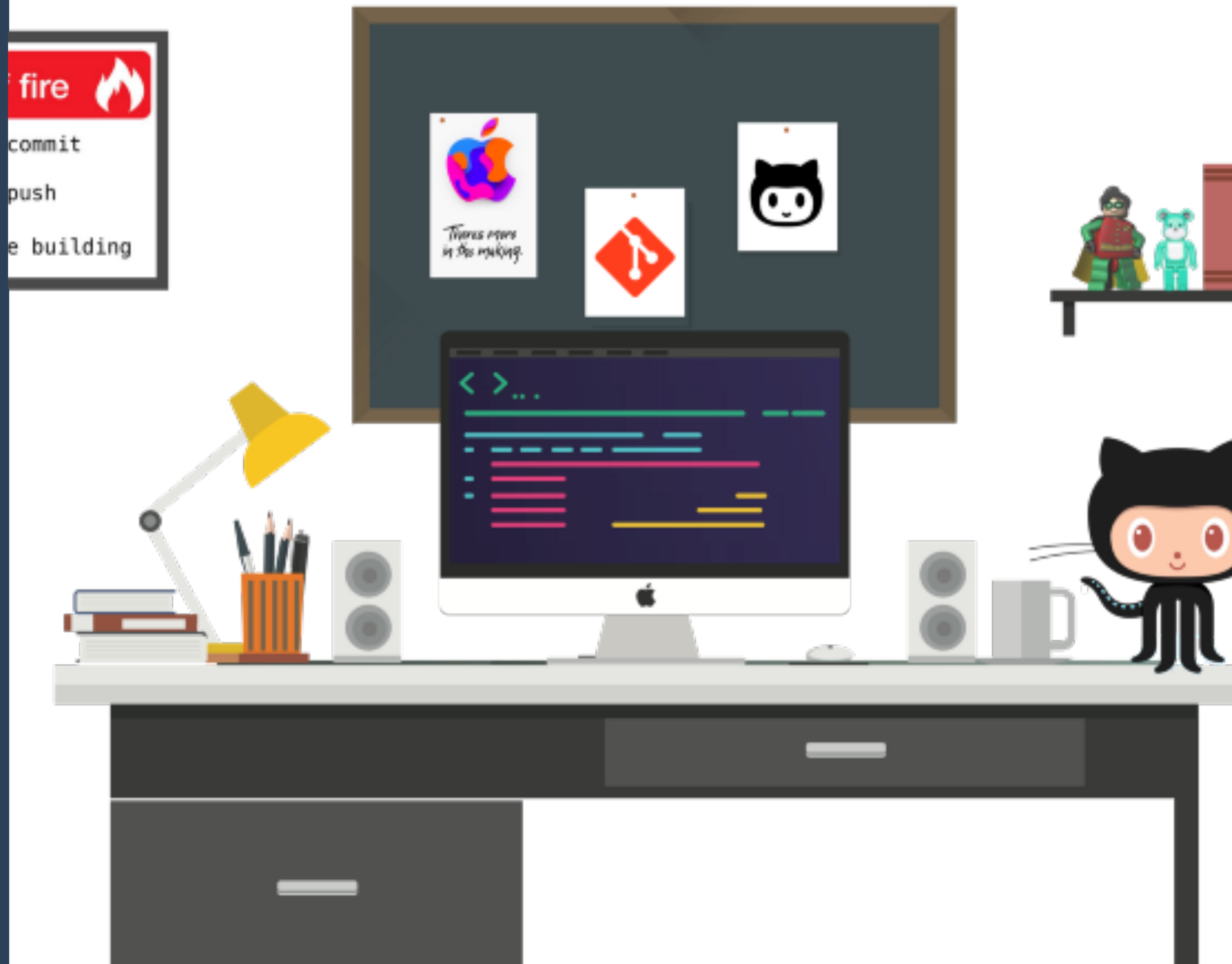
VERSION CONTROL

MODULE 3
MPCS 50101

THE UNIVERSITY OF CHICAGO

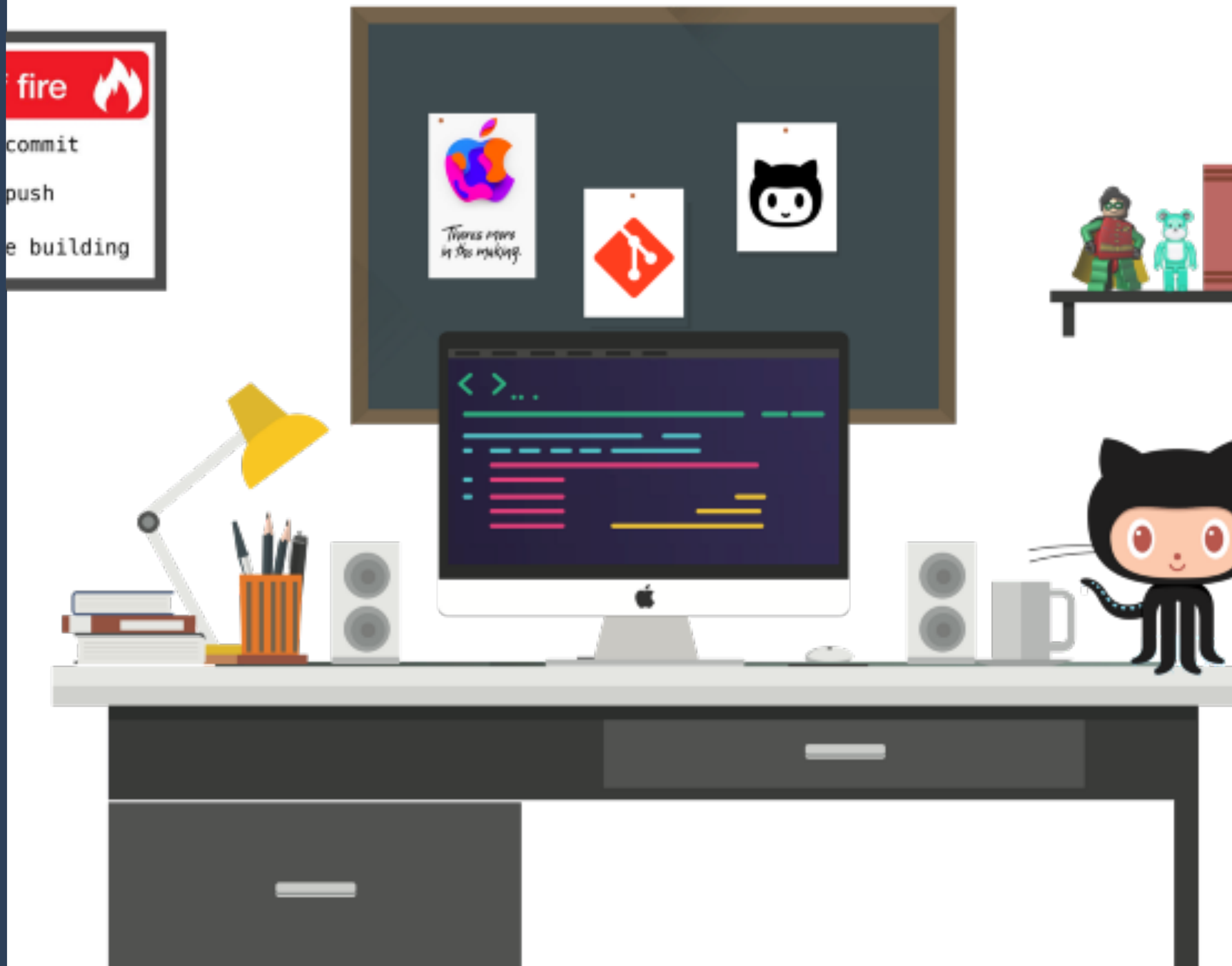
VERSION CONTROL

- 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

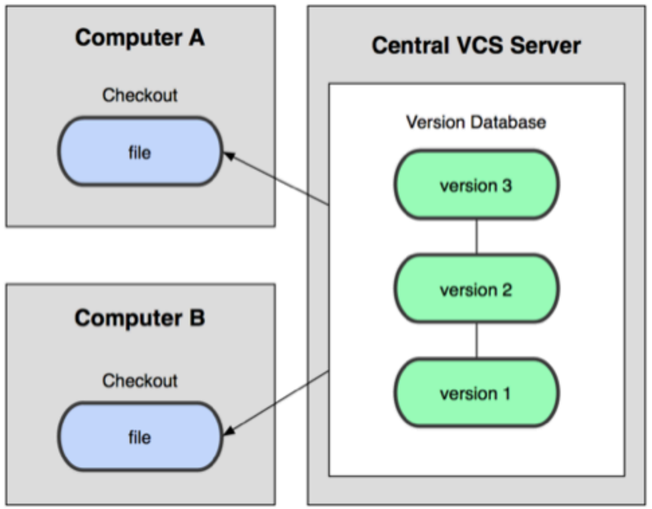
GIT

- Every git repository contains the entire history of the project

SUBVERSION

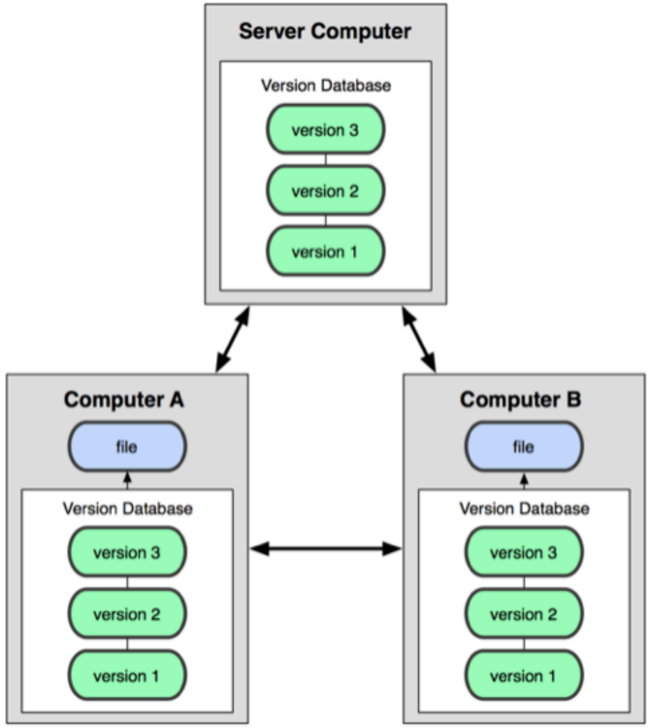
GIT, MERCURIAL

Centralized Model



(CVS, Subversion, Perforce)

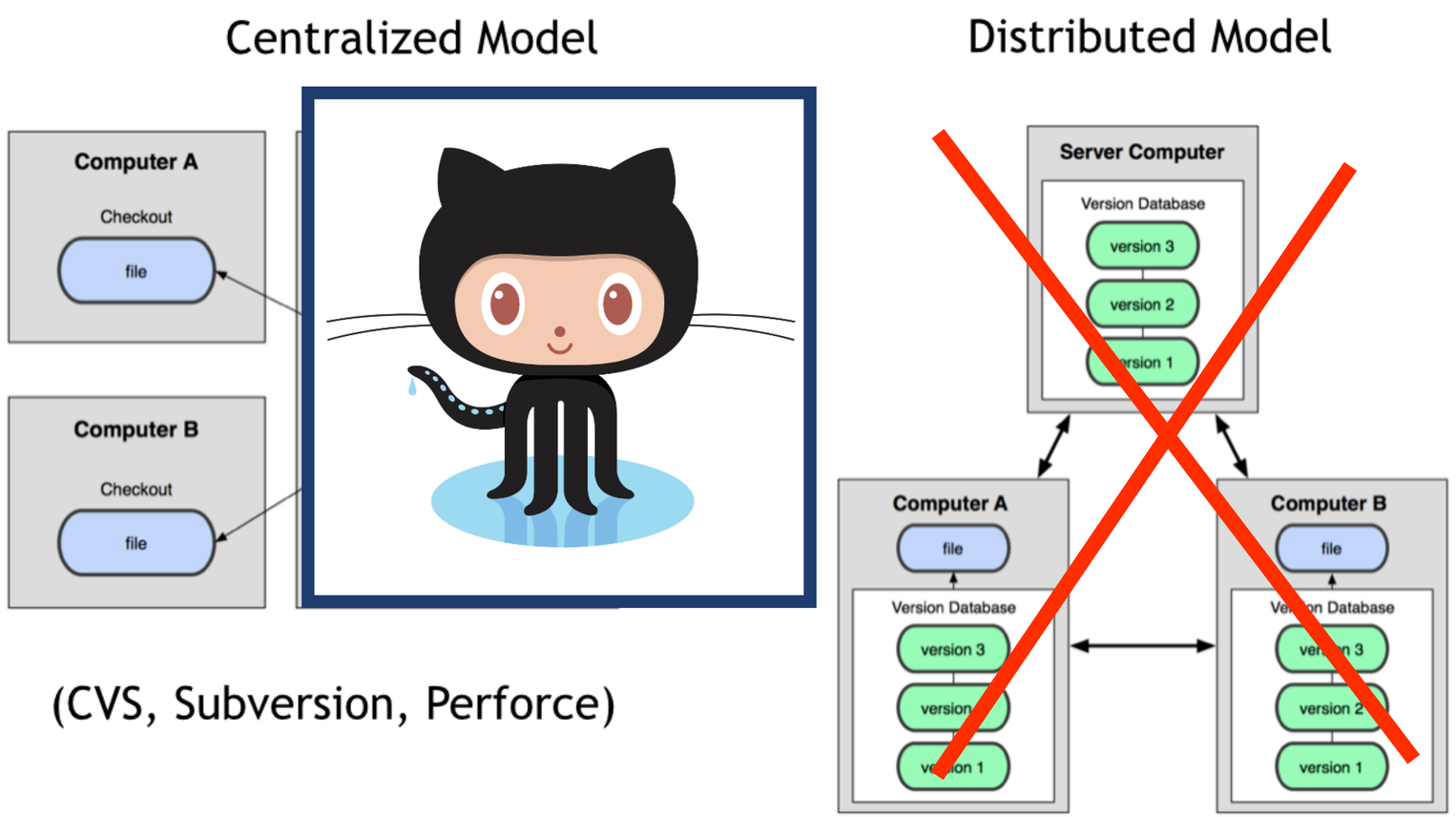
Distributed Model



GIT

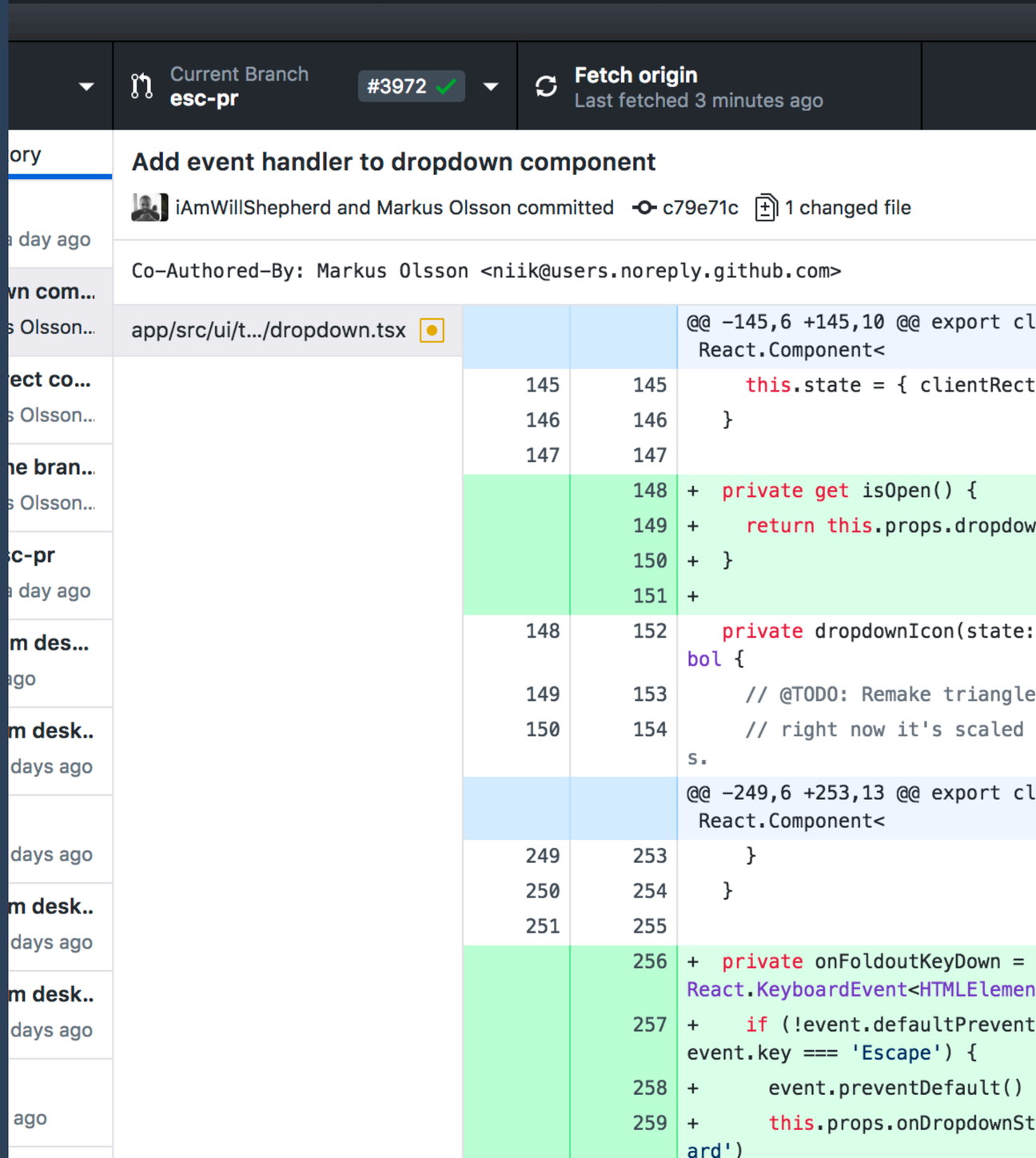
- In practice the way people use Github makes git more like centralized model
- Does not really matter

GIT, MERCURIAL



GIT

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



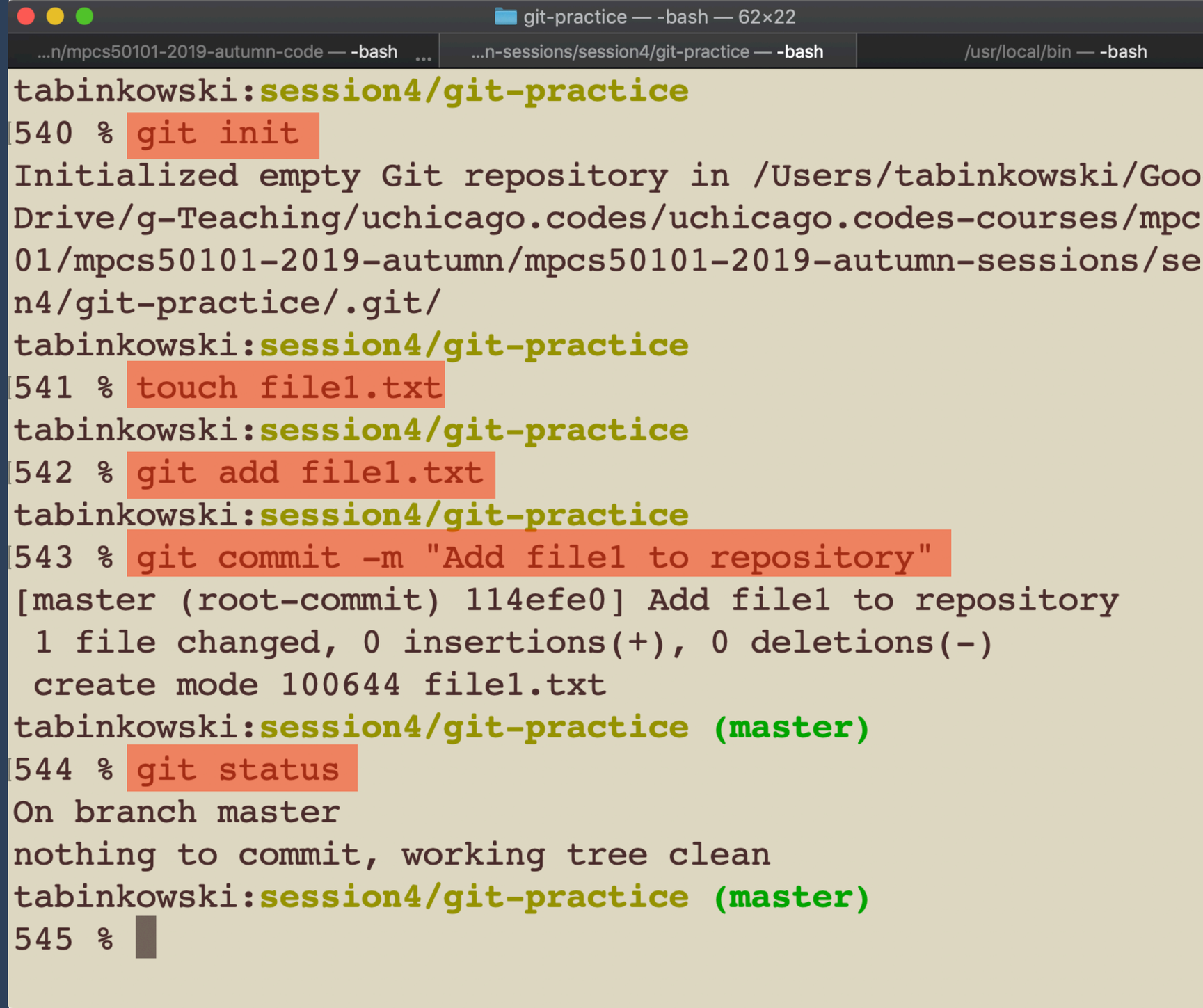
GIT

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

 /* Begin PBXFileReference section */
@@ -24,18 +32,30 @@
                37A8E0CD1419A8BC00BF37A6 /* UIKit.framework */ = {isa = PBXFileReference; lastKnownFileTy
                37A8E0CF1419A8BC00BF37A6 /* Foundation.framework */ = {isa = PBXFileReference; lastKnownF
                37A8E0D11419A8BC00BF37A6 /* CoreGraphics.framework */ = {isa = PBXFileReference; lastKnow
-               37A8E0D51419A8BC00BF37A6 /* SplashScreen-Info.plist */ = {isa = PBXFileReference; path =
+               37A8E0D51419A8BC00BF37A6 /* SplashScreen-Info.plist */ = {isa = PBXFileReference; lastKno
                37A8E0D71419A8BC00BF37A6 /* en */ = {isa = PBXFileReference; lastKnownFileType = text.pli
-               37A8E0D91419A8BC00BF37A6 /* SplashScreen-Prefix.pch */ = {isa = PBXFileReference; path =
+               37A8E0D91419A8BC00BF37A6 /* SplashScreen-Prefix.pch */ = {isa = PBXFileReference; lastKno
                37A8E0DA1419A8BC00BF37A6 /* main.m */ = {isa = PBXFileReference; lastKnownFileType = sour
-               37A8E0DC1419A8BC00BF37A6 /* SplashScreenAppDelegate.h */ = {isa = PBXFileReference; path
```

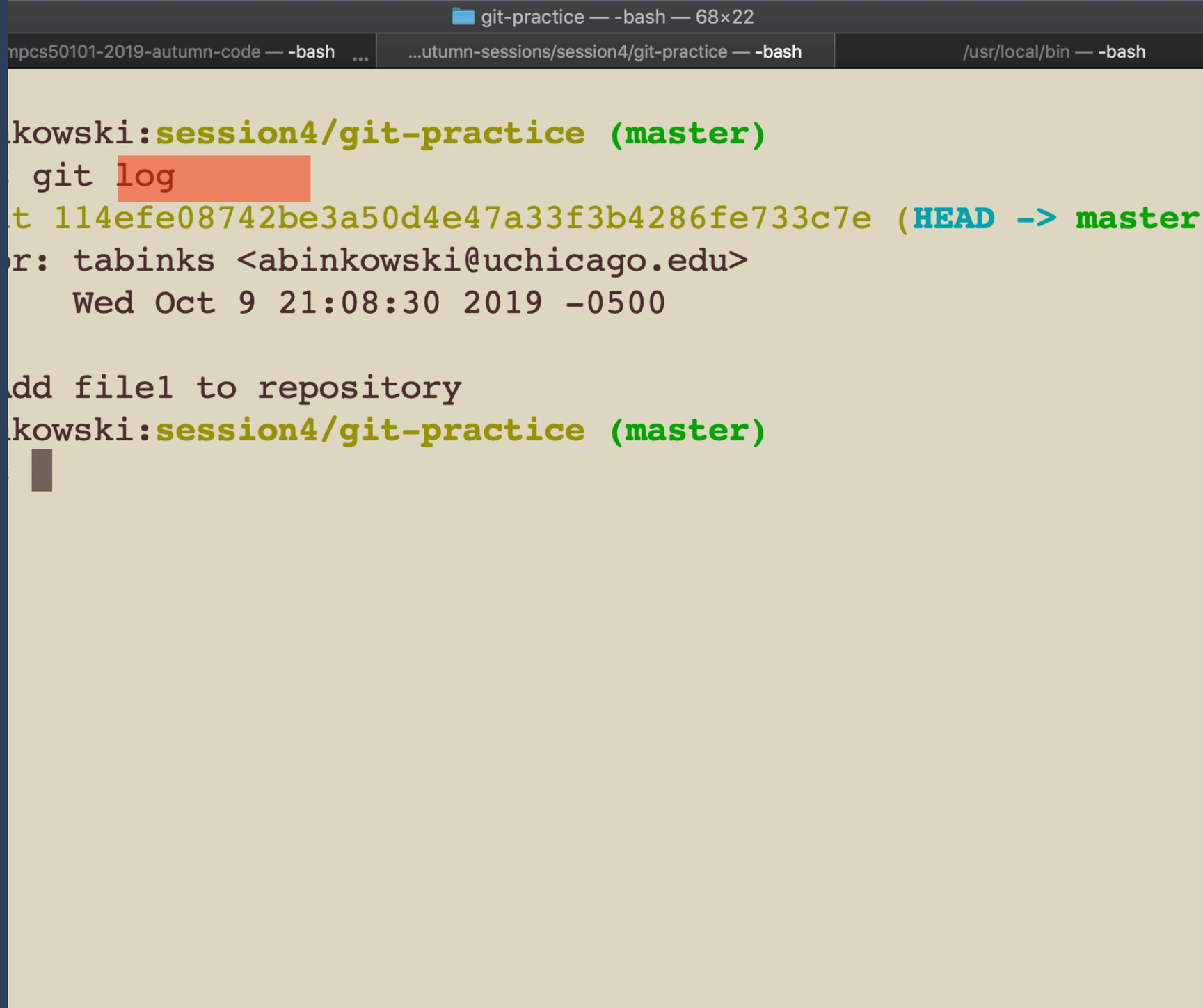
GIT

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



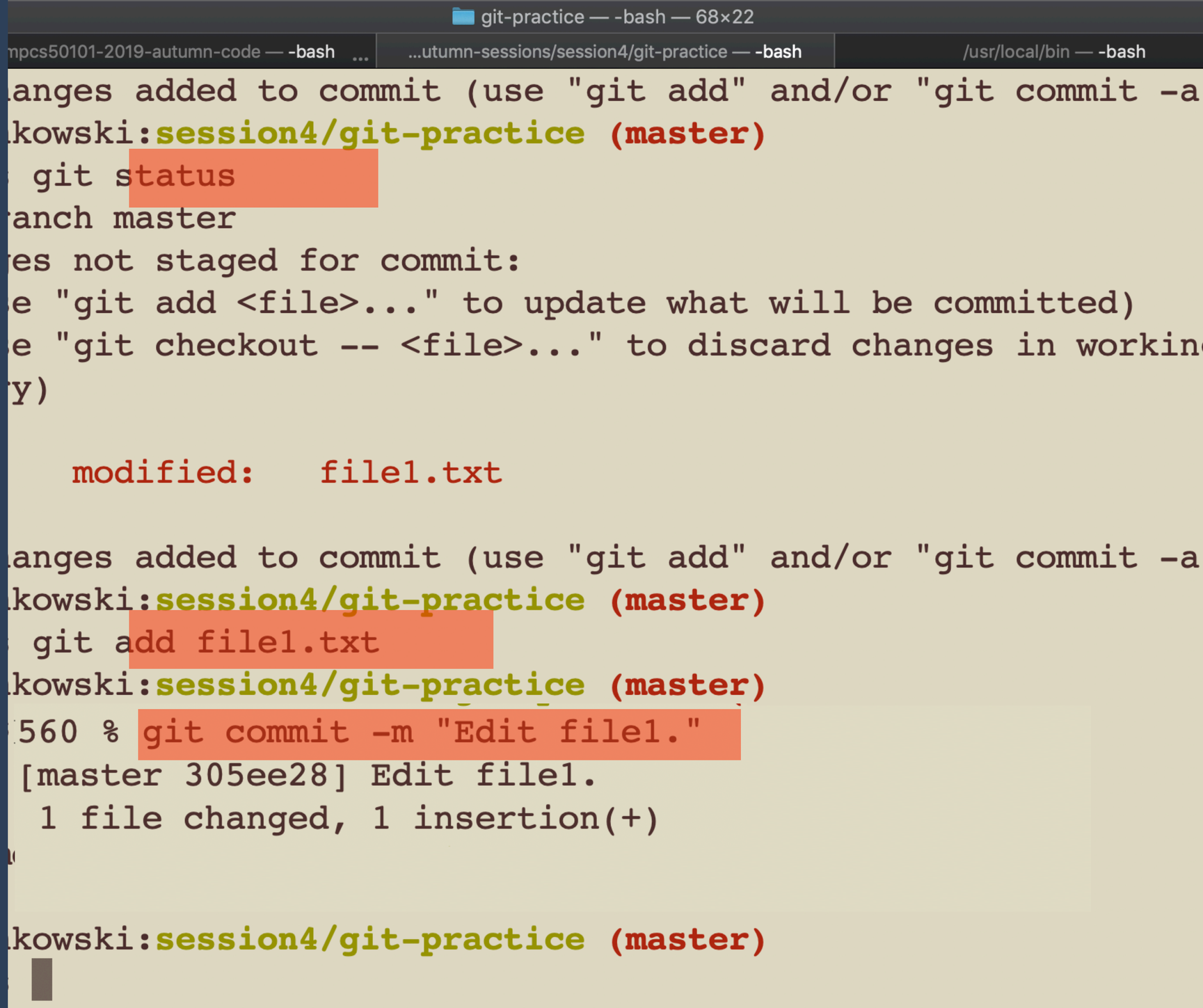
GIT

- git log
- git status
- git checkout



GIT

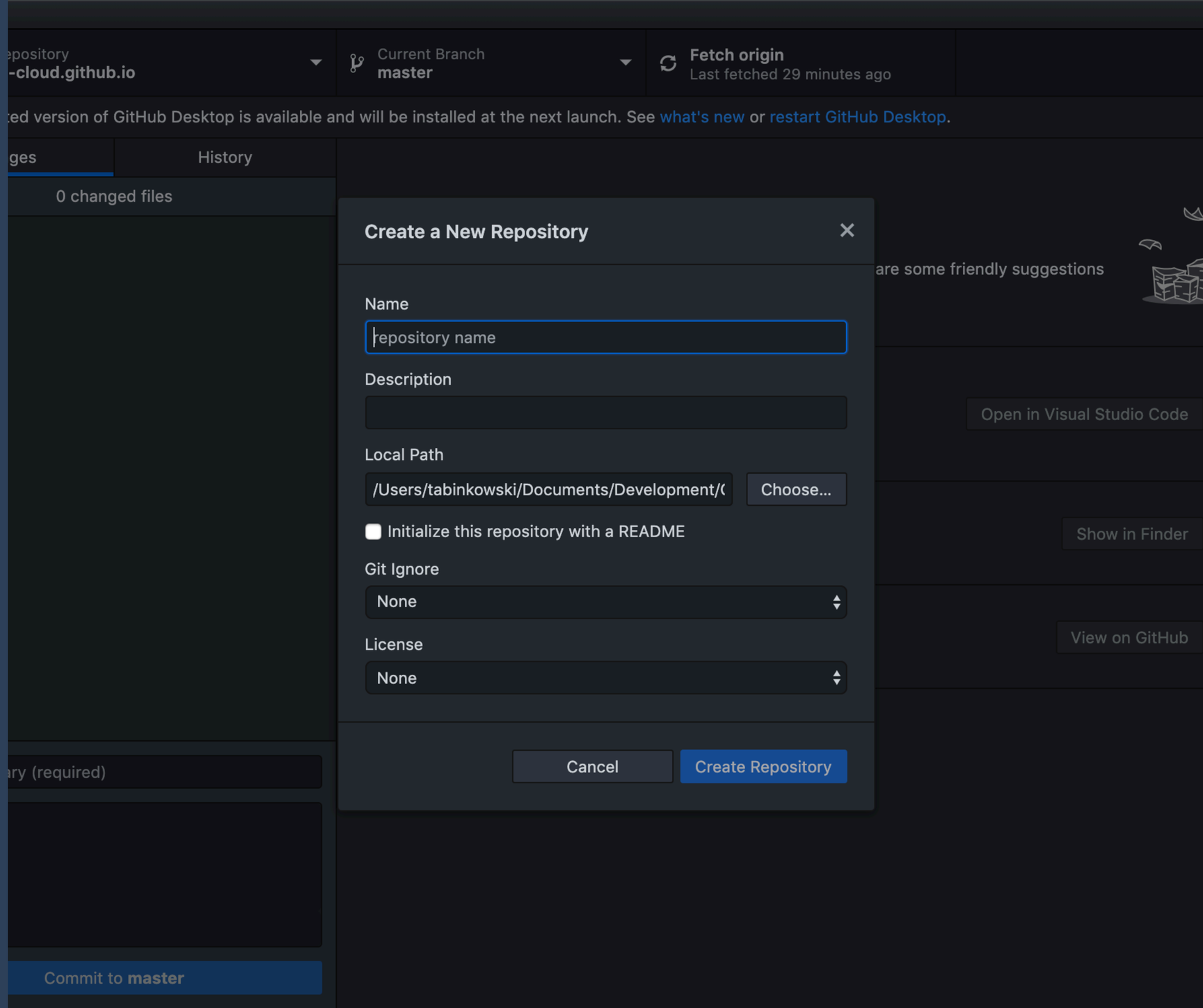
- git status
- git add
- git commit



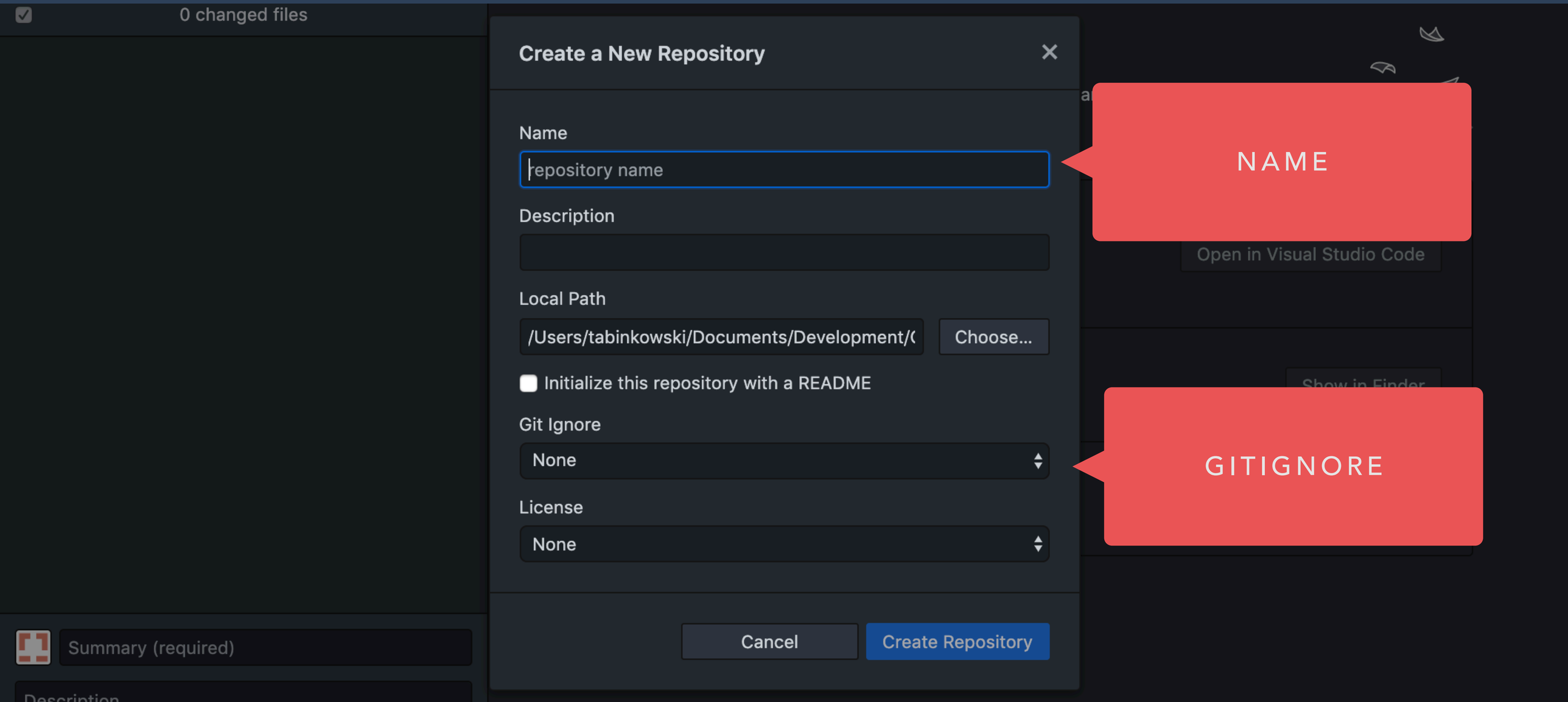
GITHUB DESKTOP

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

Publish repository
Publish this repository to GitHub

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 ⌘ P

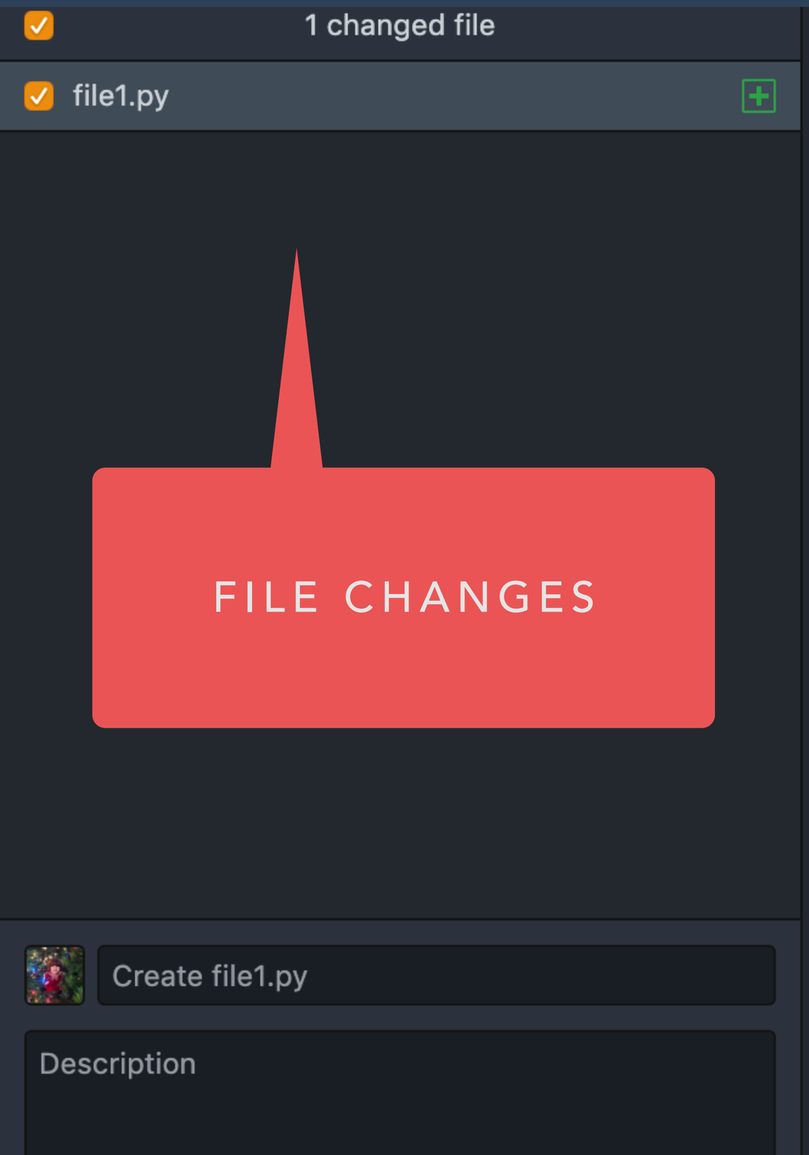
Publish repository

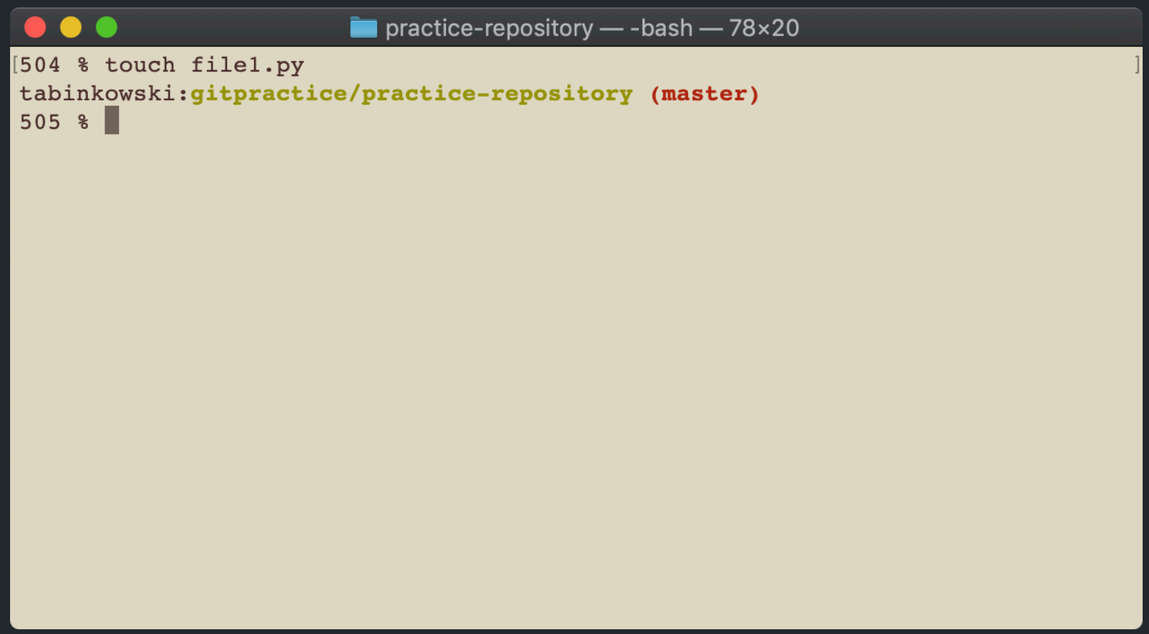
Open the repository in your external editor
Select your editor in Preferences

Repository menu or ⌘ ⇧ A

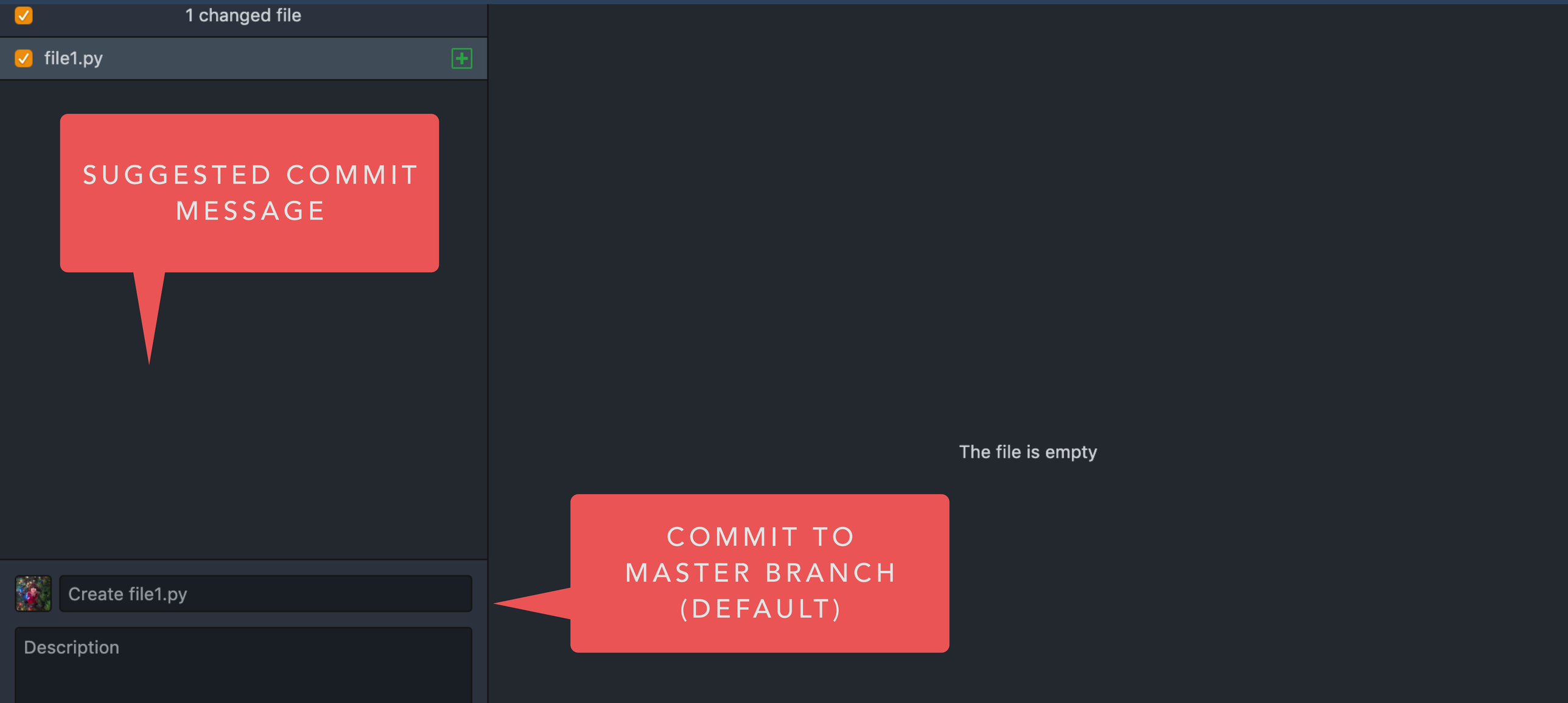
Open in Visual Studio Code

GITHUB DESKTOP

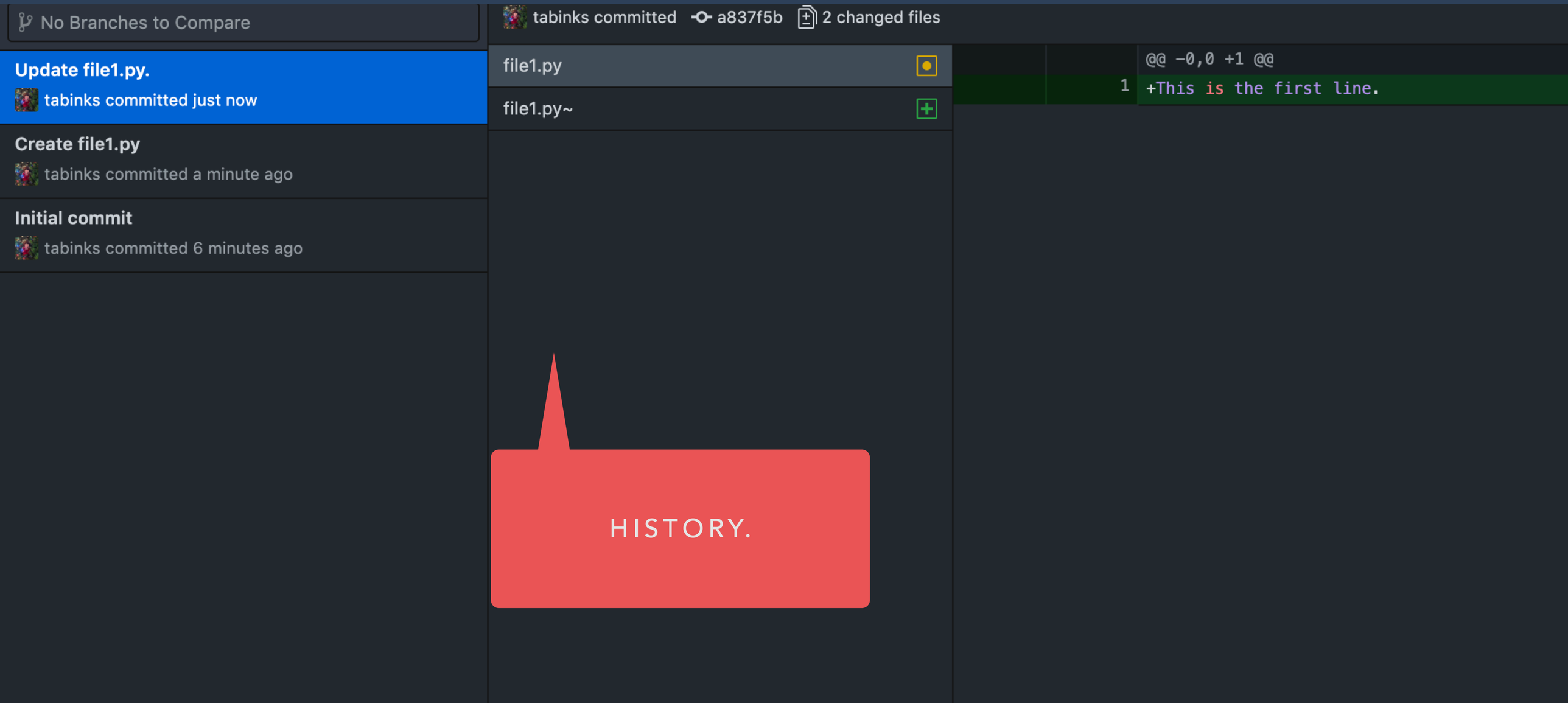




GITHUB DESKTOP



GITHUB DESKTOP



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 SSH 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

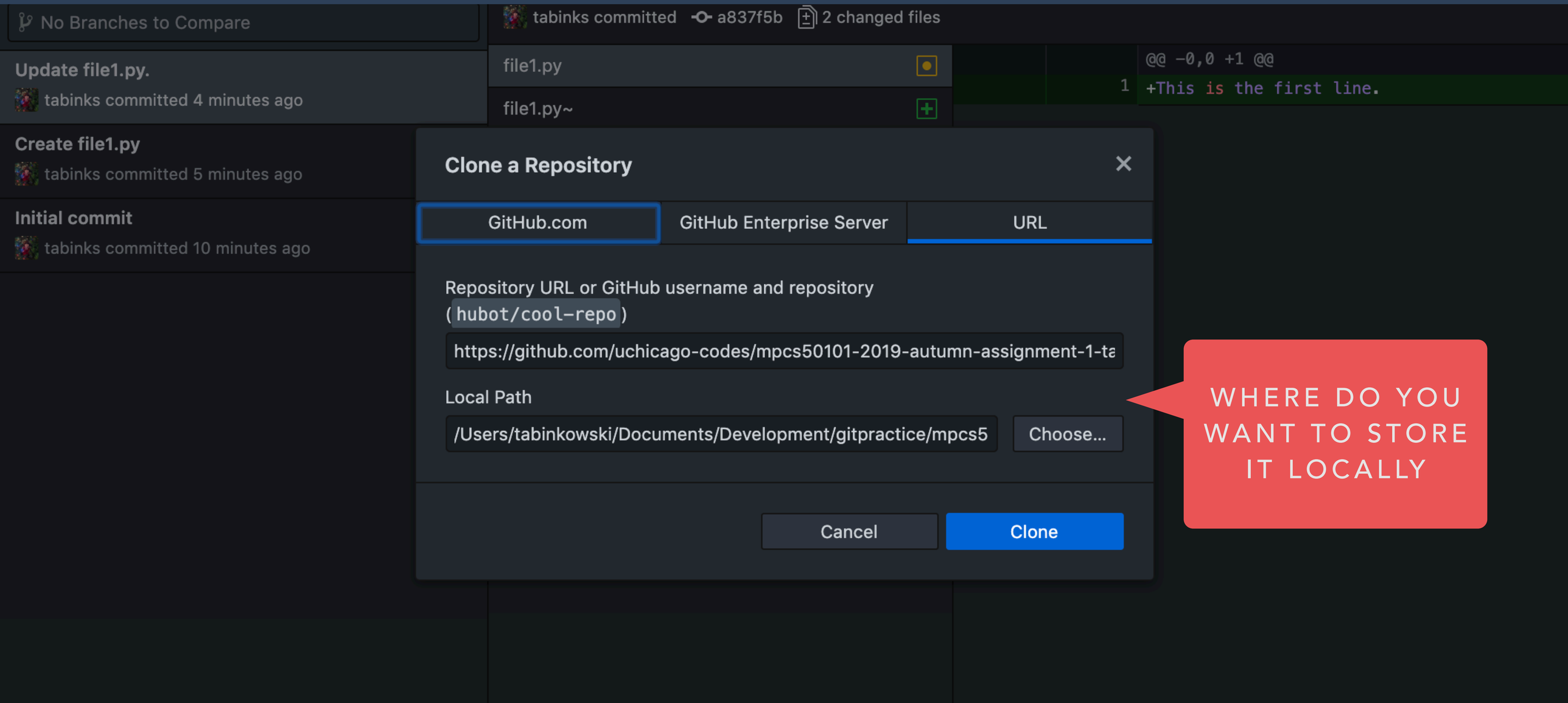
```
echo "# mpcs50101-2019-autumn-assignment-1-tabinks" >> README.md
git init
git ad
git co
git re                /github.com/uchicago-codes/mpcs50101-2019-autumn-assignment-1-tabinks.git
git pu
```

USE THIS

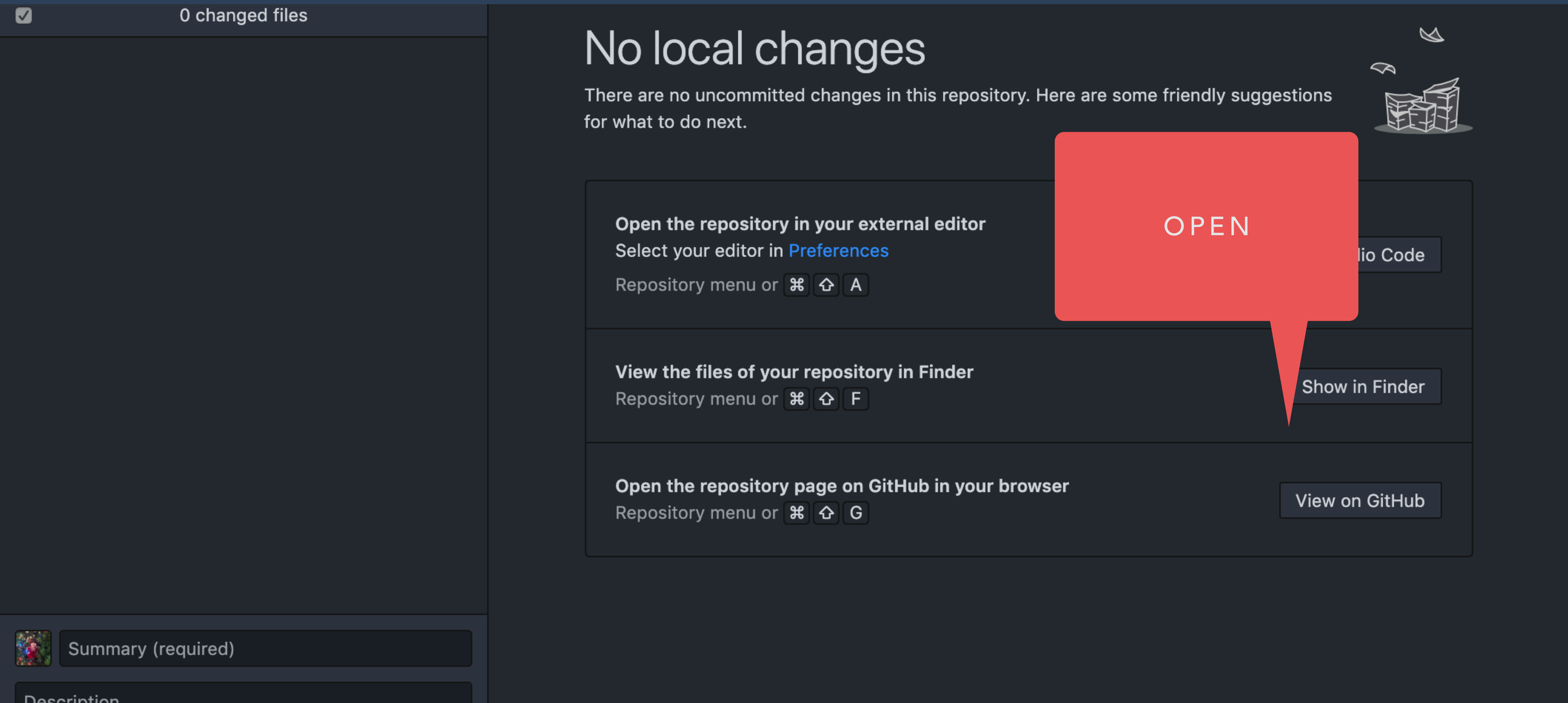
...or push an existing repository from the command line

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

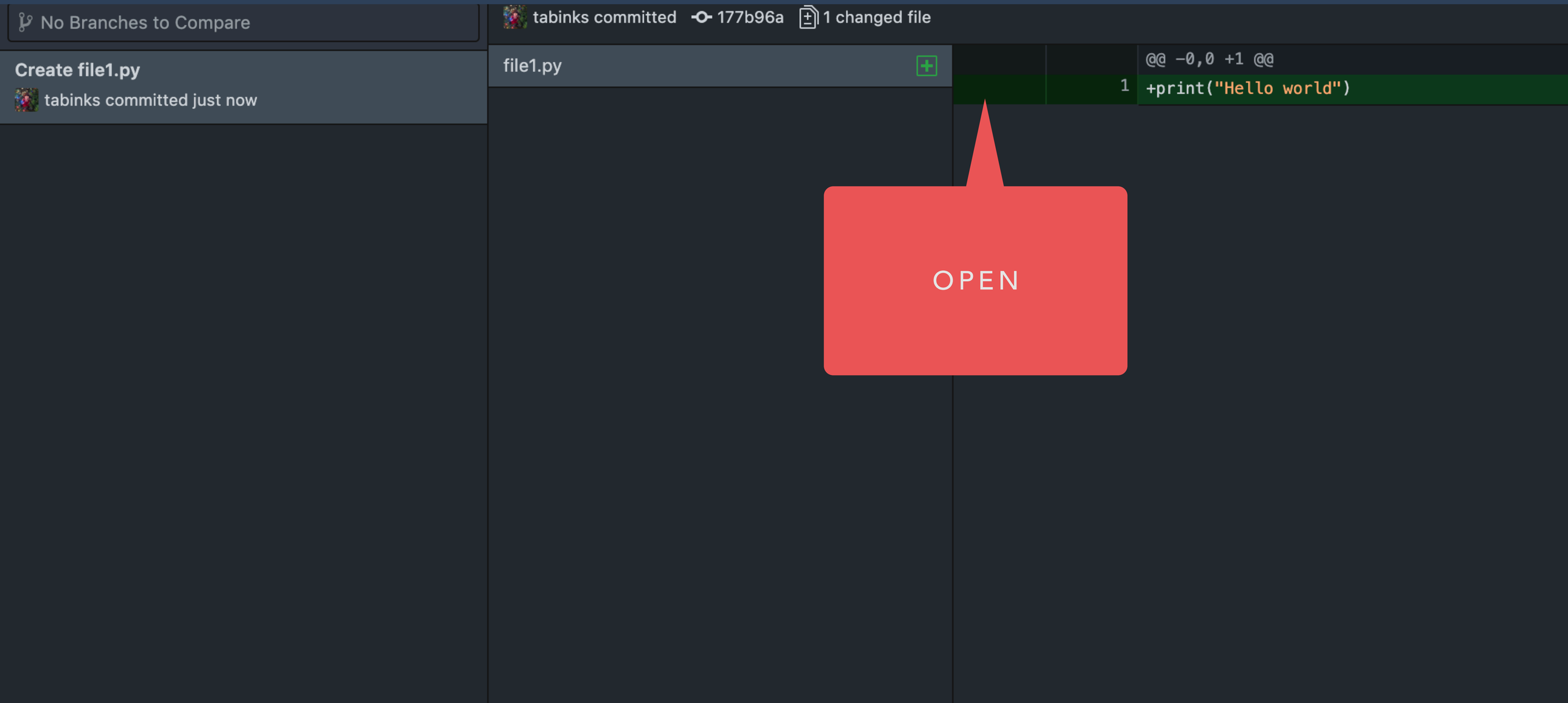
GITHUB CLASSROOM



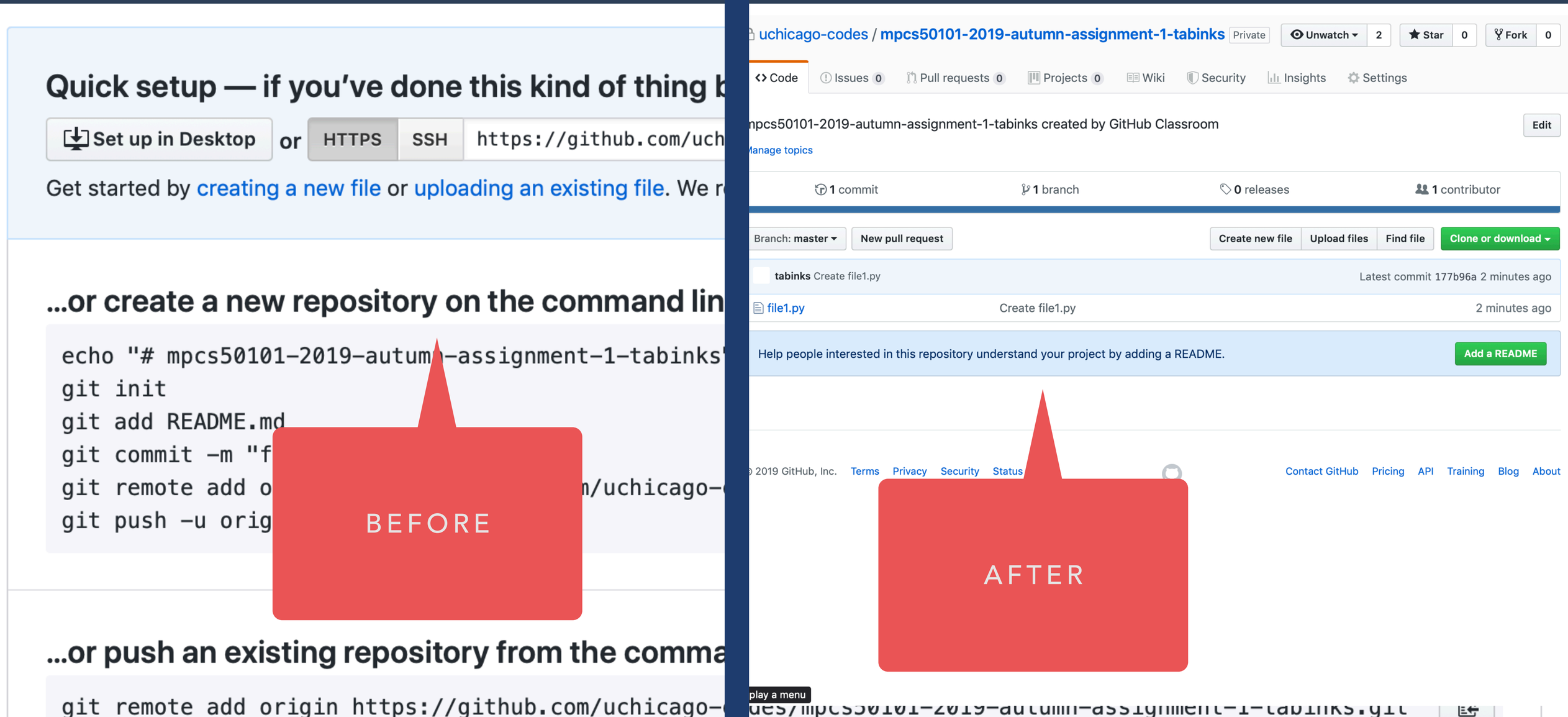
GITHUB CLASSROOM



GITHUB CLASSROOM



GITHUB CLASSROOM



ASSIGNMENT

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 roc

All the logic l actices in
module desig me play code.
Anything related me . The code
running the main uld be part of
the `__main__` mo

Keep in mind bes e.

VIDEO WALKTHROUGH
IN MODULE

Problem 2

THE END

MODULE 4
MPCS 50101

THE UNIVERSITY OF CHICAGO