CONCEPTS OF PROGRAMMING

MPCS 50101
MODULE 1

THE UNIVERSITY OF CHICAGO

WELCOME TO PROGRAMMING

MODULE 1
MPCS 50101

THE UNIVERSITY OF CHICAGO

PROGRAMMING

- Programming is how a person writes instructions to tell a computer what to do
- Now matter how fancy AI/ machine learning are using… you still need to tell a computer exactly what to do

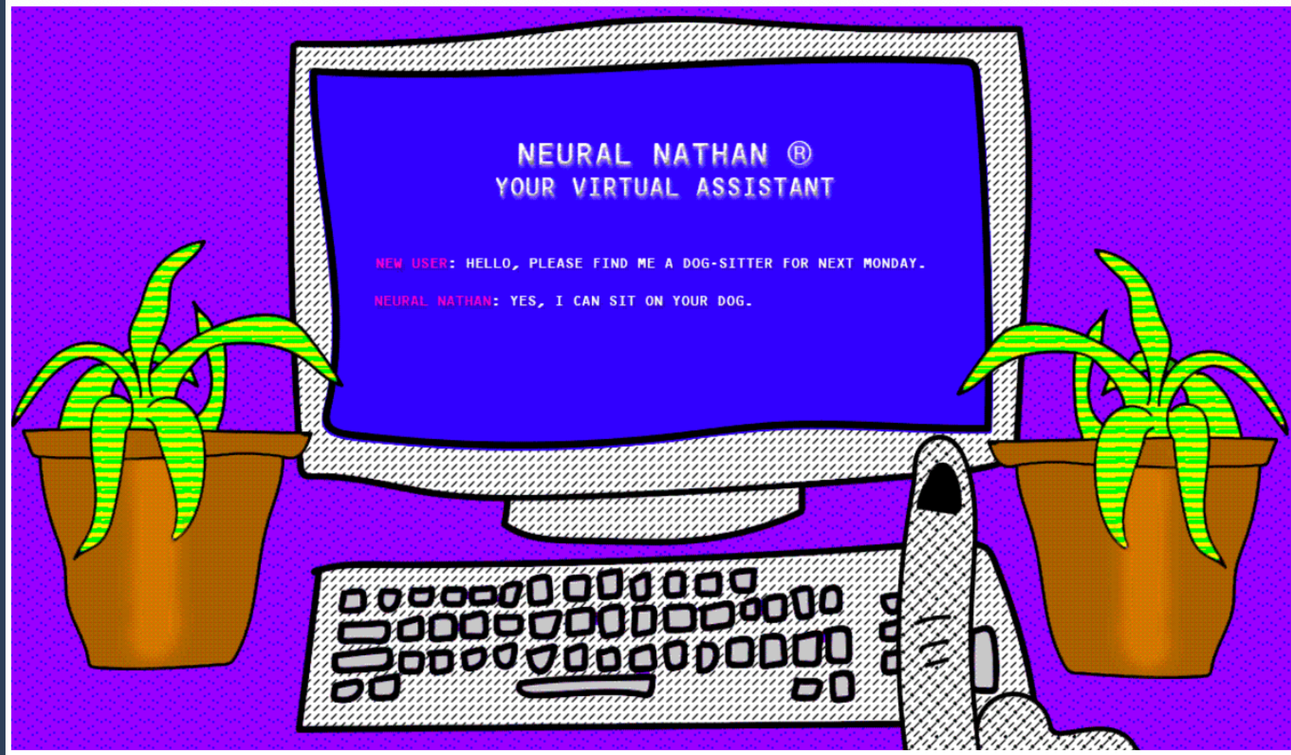
PROGRAMMING

- Sometimes the best programs are actually humans

The Humans Hiding Behind the Chatbots

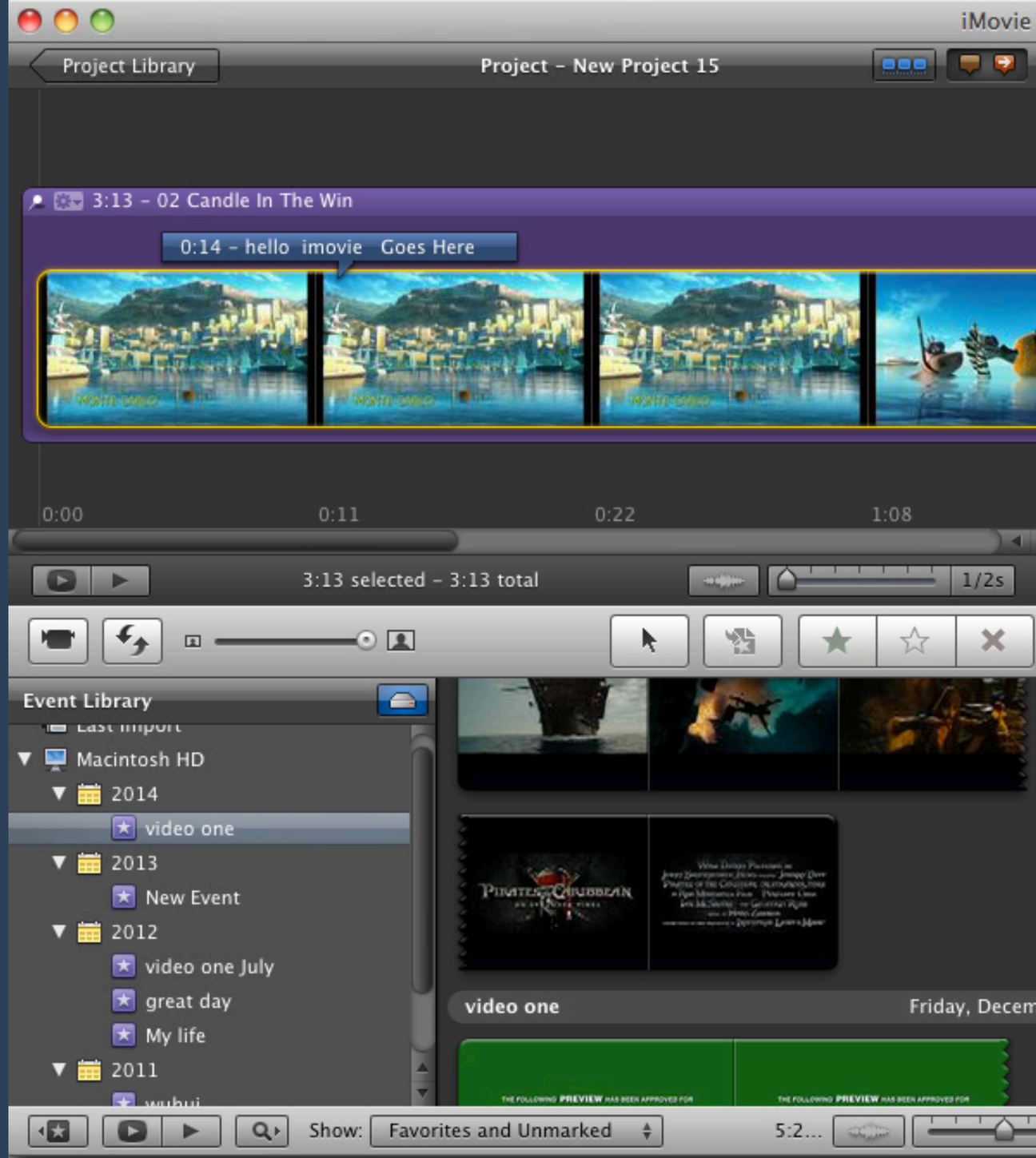
Behind the artificial intelligence personal assistants and concierges are actual people, reading e-mails and ordering Chipotle.

By **Ellen Huet**



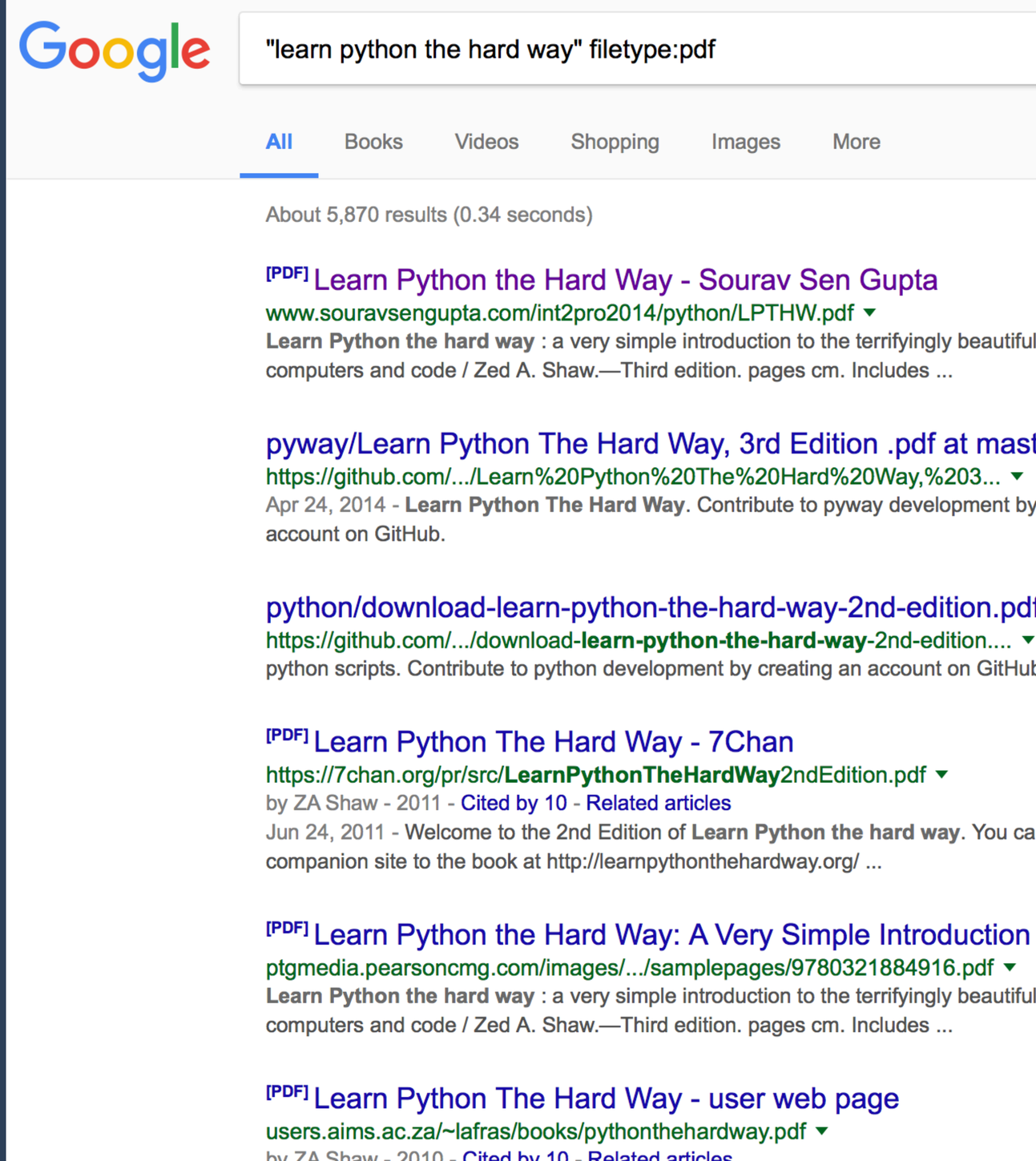
PROGRAMMING

- Ways people interact with a computer program
- User of software
  - Running instructions that have been designed by someone else



PROGRAMMING

- Ways people interact with a computer program
- Power user
  - Google queries, Excel macros



PROGRAMMING

- Ways people interact with a computer program
- Programmer
  - Write your own code

```
import random

def drawBoard(board):
    # This function prints out the board that it was passed.

    # "board" is a list of 10 strings representing the board (ignore index 0)
    print('   |   |')
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
    print('   |   |')

def inputPlayerLetter():
    # Lets the player type which letter they want to be.
    # Returns a list with the player's letter as the first item, and the compute
    letter = ''
    while not (letter == 'X' or letter == 'O'):
        print('Do you want to be X or O?')
        letter = input().upper()

    # the first element in the list is the player's letter, the second is the comp
    if letter == 'X':
        return ['X', 'O']
    else:
        return ['O', 'X']
```

PROGRAMMING LANGUAGES

PROGRAMMING LANGUAGES

- A language that allows a programmer execute instructions on a computer
- The instructions are called the source code

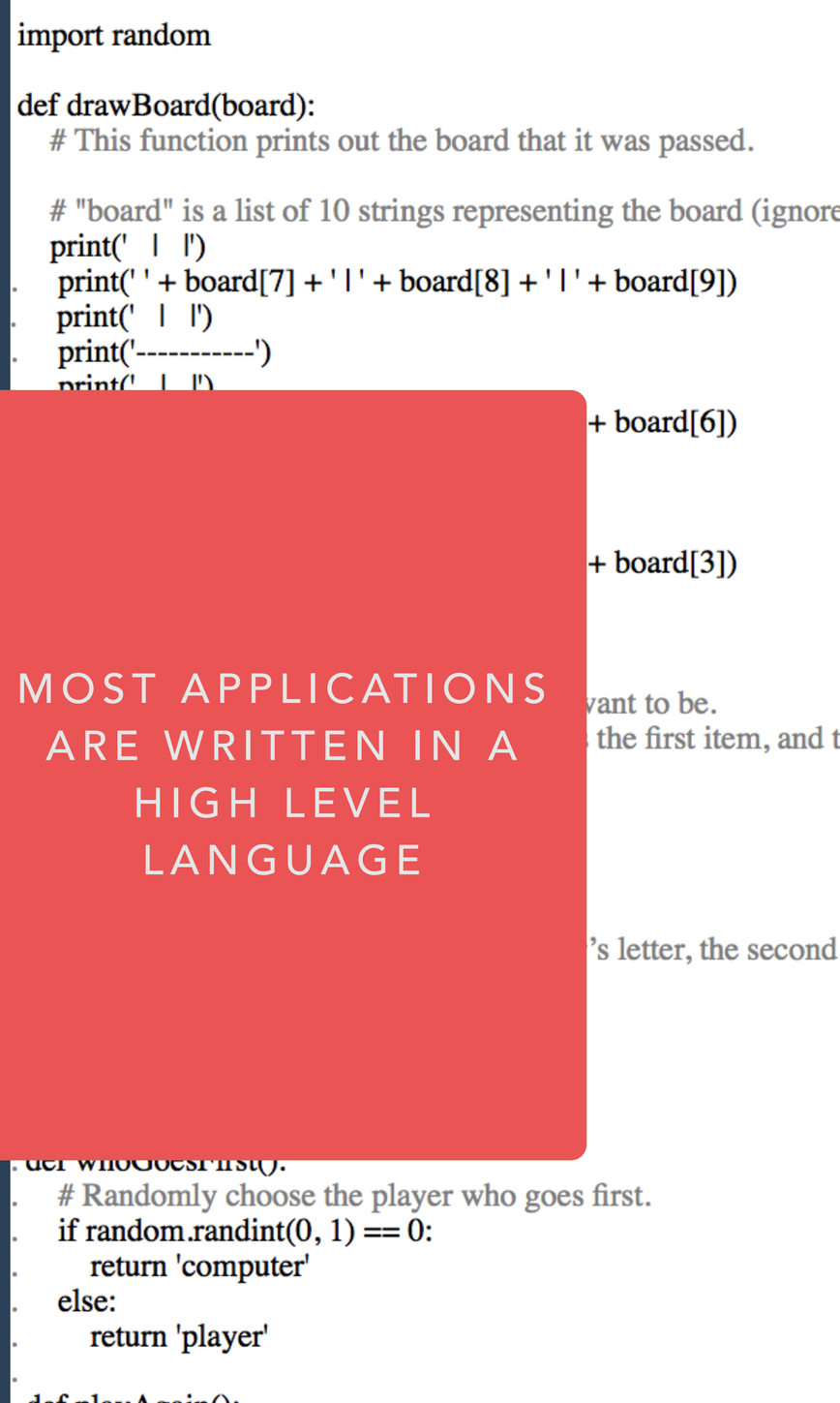
PROGRAMMING LANGUAGES

- Two different kinds of languages
  - Low-level
    - Machine, assembly code
  - High-level
    - Text languages with syntax, rules, etc.

PROGRAMMING LANGUAGES

- High-level languages
  - Simpler languages that are converted to low-level languages
  - Examples: C, C++, Perl, Java, Swift, Go
  - Advantages
    - Easier to write, shorter, faster development time, portable

MOST APPLICATIONS ARE WRITTEN IN A HIGH LEVEL LANGUAGE



PROGRAMMING LANGUAGES

- High-level language vs. low-level language

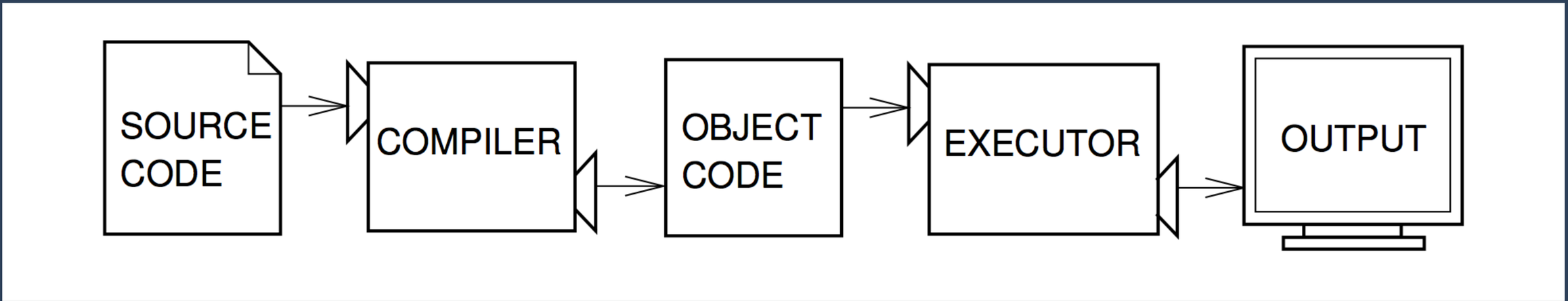
S.NO	HIGH LEVEL LANGUAGE	LOW LEVEL LANGUAGE
1.	It is programmer friendly language.	It is a machine friendly language.
2.	High level language is less memory efficient.	Low level language is high memory efficient.
3.	It is easy to understand.	It is tough to understand.
4.	It is simple to debug.	It is complex to debug comparatively.
5.	It is simple to maintain.	It is complex to maintain comparatively.
6.	It is portable.	It is non-portable.
7.	It can run on any platform.	It is machine-dependent.
8.	It needs compiler or interpreter for translation.	It needs assembler for translation.
9.	It is used widely for programming.	It is not commonly used now-a-days in programming.

https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/

PROGRAMMING LANGUAGES

- Python, JavaScript
  - Interpreted every time it runs
- C, C++
  - Compiled into an executable file
- Assembly language
  - Assembled into machine code
- Machine code
  - Run by the CPU

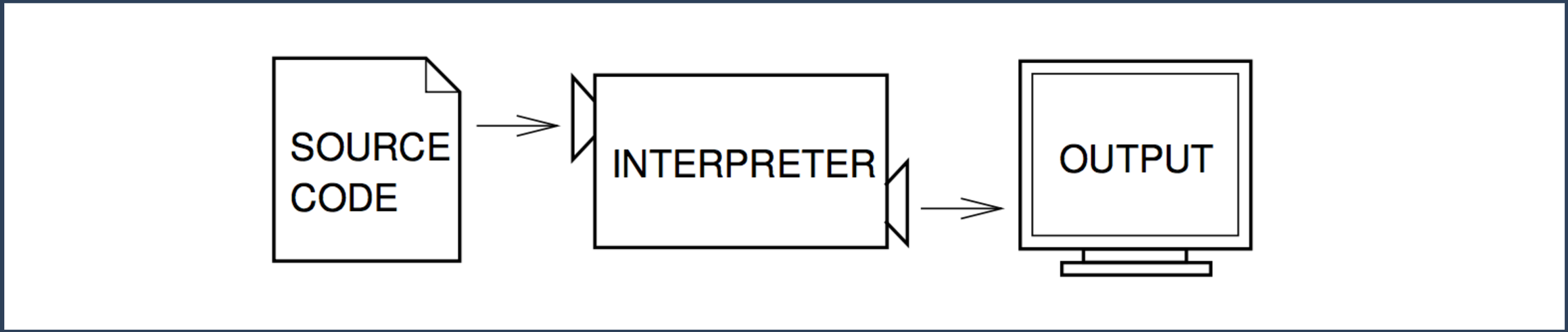
PROGRAMMING LANGUAGES



- Translate a high-level language program using a **compiler**
- Done a single time to create an executable program

HIGH-LEVEL LANGUAGES MUST BE CONVERTED TO CODE THE MACHINE CAN READ

PROGRAMMING LANGUAGES



- Translating a high-level language program using an **interpreter**
- File is read line by line and executed

HIGH-LEVEL LANGUAGES MUST BE CONVERTED TO CODE THE MACHINE CAN READ

PROGRAMMING LANGUAGES

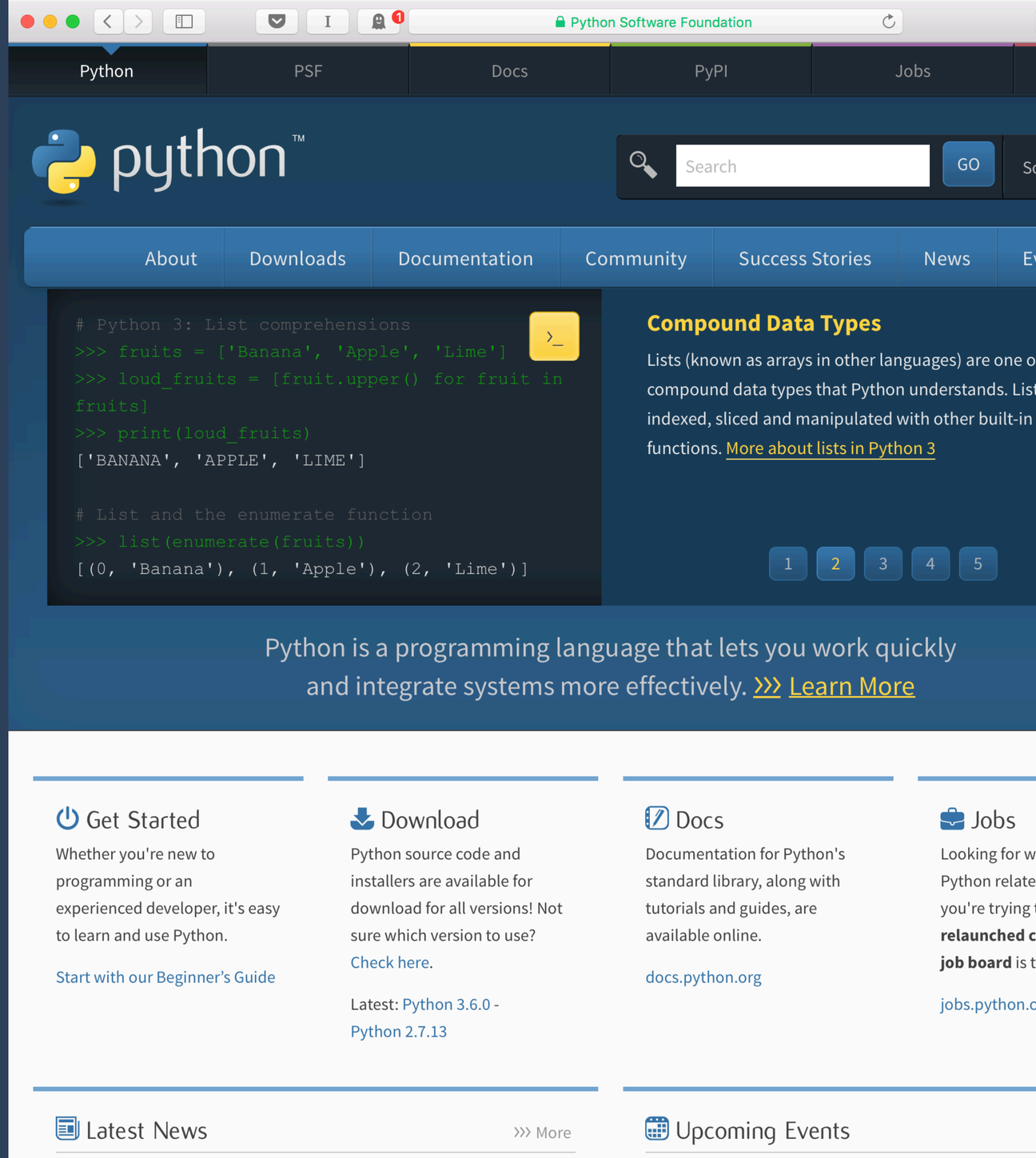
- Interpreted
  - Source code run line-by-line
  - Faster development cycle
  - Slower than compiled
  - Examples: Perl, Python, Ruby, Swift
- Compiled
  - Source code converted to executable
  - Better performance
  - Slower development time
  - Examples: C, C++, Go, Swift

PROGRAMMING LANGUAGES

- Languages, politics, religion
- Why choose one over another
  - Targeted platform (iOS)
  - Development time
  - Performance
  - Familiarity

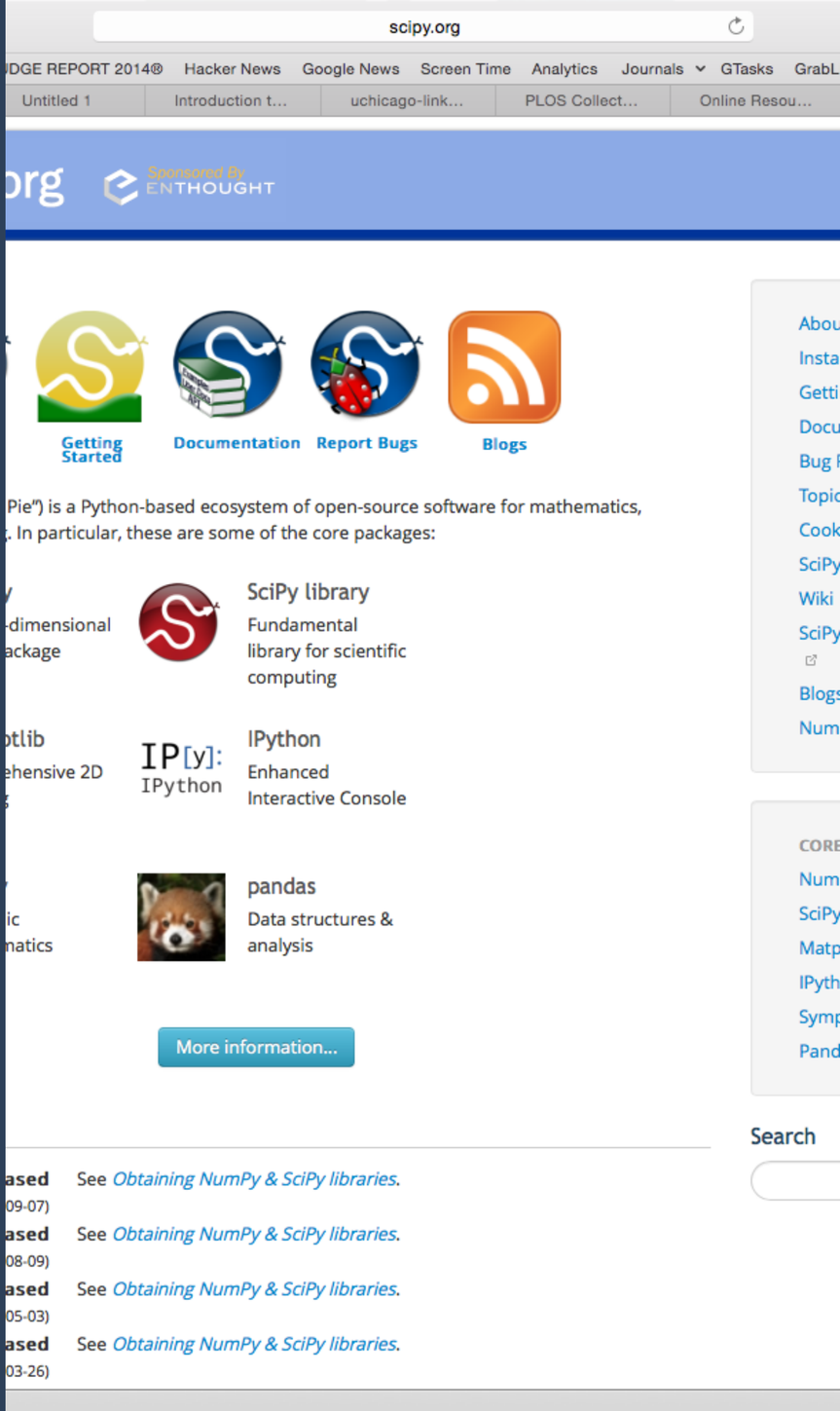
PROGRAMMING LANGUAGES

- Python
  - Interpreted language
  - Runs everywhere
  - Widely used across many fields (and in the program)
  - Strong ecosystem for third-party code and tools



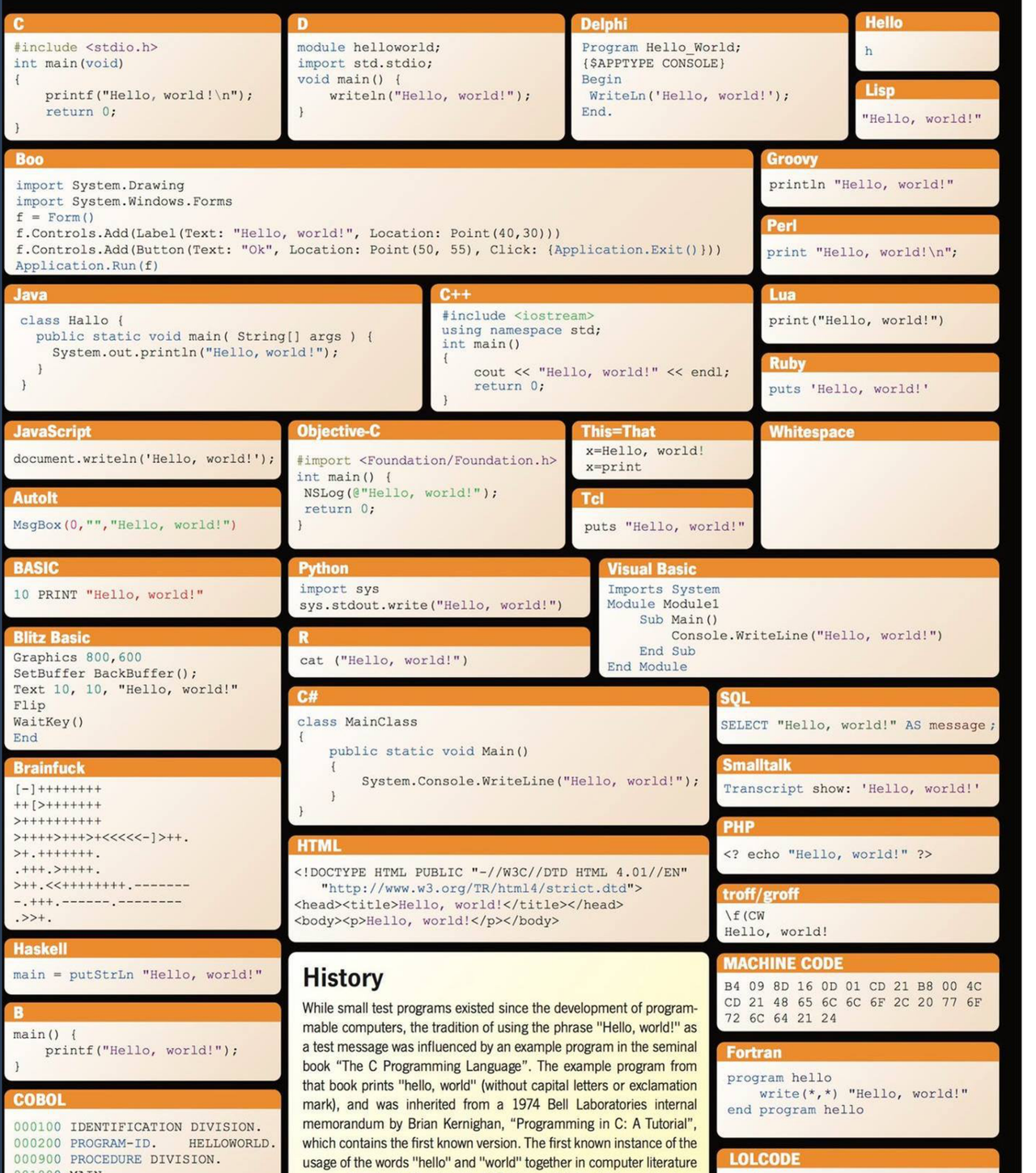
COURSE TECHNOLOGIES

- Python
  - Support for SciPy scientific programming
  - Extensibility for C modules
    - Optimize when you need it
  - Native web application language



PROGRAMMING LANGUAGES

- Once you know one language, learning a new language is straightforward
  - "Hello world" in different languages
- You will spend more time thinking about how to solve the problem
  - Are you fighting the problem or the language?



C
```
#include <stdio.h>
int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

D
```
module helloworld;
import std.stdio;
void main() {
    writeln("Hello, world!");
}
```

Delphi
```
Program Hello_World;
{$APPTYPE CONSOLE}
Begin
 WriteLn('Hello, world!');
End.
```

Hello
```
h
```

Lisp
```
"Hello, world!"
```

Boo
```
import System.Drawing
import System.Windows.Forms
f = Form()
f.Controls.Add(Label(Text: "Hello, world!", Location: Point(40,30)))
f.Controls.Add(Button(Text: "Ok", Location: Point(50, 55), Click: {Application.Exit()}))
Application.Run(f)
```

Groovy
```
println "Hello, world!"
```

Perl
```
print "Hello, world!\n";
```

Java
```
class Hallo {
  public static void main( String[] args ) {
    System.out.println("Hello, world!");
  }
}
```

C++
```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, world!" << endl;
    return 0;
}
```

Lua
```
print("Hello, world!")
```

Ruby
```
puts 'Hello, world!'
```

JavaScript
```
document.writeln('Hello, world!');
```

Objective-C
```
#import <Foundation/Foundation.h>
int main() {
 NSLog(@"Hello, world!");
 return 0;
}
```

This=That
```
x=Hello, world!
x=print
```

Whitespace

Autolt
```
MsgBox(0,"","Hello, world!")
```

Tcl
```
puts "Hello, world!"
```

BASIC
```
10 PRINT "Hello, world!"
```

Python
```
import sys
sys.stdout.write("Hello, world!")
```

Visual Basic
```
Imports System
Module Module1
    Sub Main()
        Console.WriteLine("Hello, world!")
    End Sub
End Module
```

Blitz Basic
```
Graphics 800,600
SetBuffer BackBuffer()
Text 10, 10, "Hello, world!"
Flip
WaitKey()
End
```

R
```
cat ("Hello, world!")
```

C#
```
class MainClass
{
    public static void Main()
    {
        System.Console.WriteLine("Hello, world!");
    }
}
```

SQL
```
SELECT "Hello, world!" AS message ;
```

Brainfuck
```
[-]++++++++
++[>+++++++
>++++++++++
>++++>+++>+<<<<<-]>++.
>+.+++++++.
.+++.>++++.
>++.<<++++++++.-------
-.+++.------.--------
.>>+.
```

Smalltalk
```
Transcript show: 'Hello, world!'
```

PHP
```
<? echo "Hello, world!" ?>
```

HTML
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<head><title>Hello, world!</title></head>
<body><p>Hello, world!</p></body>
```

troff/groff
```
\f(CW
Hello, world!
```

Haskell
```
main = putStrLn "Hello, world!"
```

B
```
main() {
    printf("Hello, world!");
}
```

COBOL
```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    HELLOWORLD.
000900 PROCEDURE DIVISION.
```

History

While small test programs existed since the development of programmable computers, the tradition of using the phrase "Hello, world!" as a test message was influenced by an example program in the seminal book "The C Programming Language". The example program from that book prints "hello, world" (without capital letters or exclamation mark), and was inherited from a 1974 Bell Laboratories internal memorandum by Brian Kernighan, "Programming in C: A Tutorial", which contains the first known version. The first known instance of the usage of the words "hello" and "world" together in computer literature

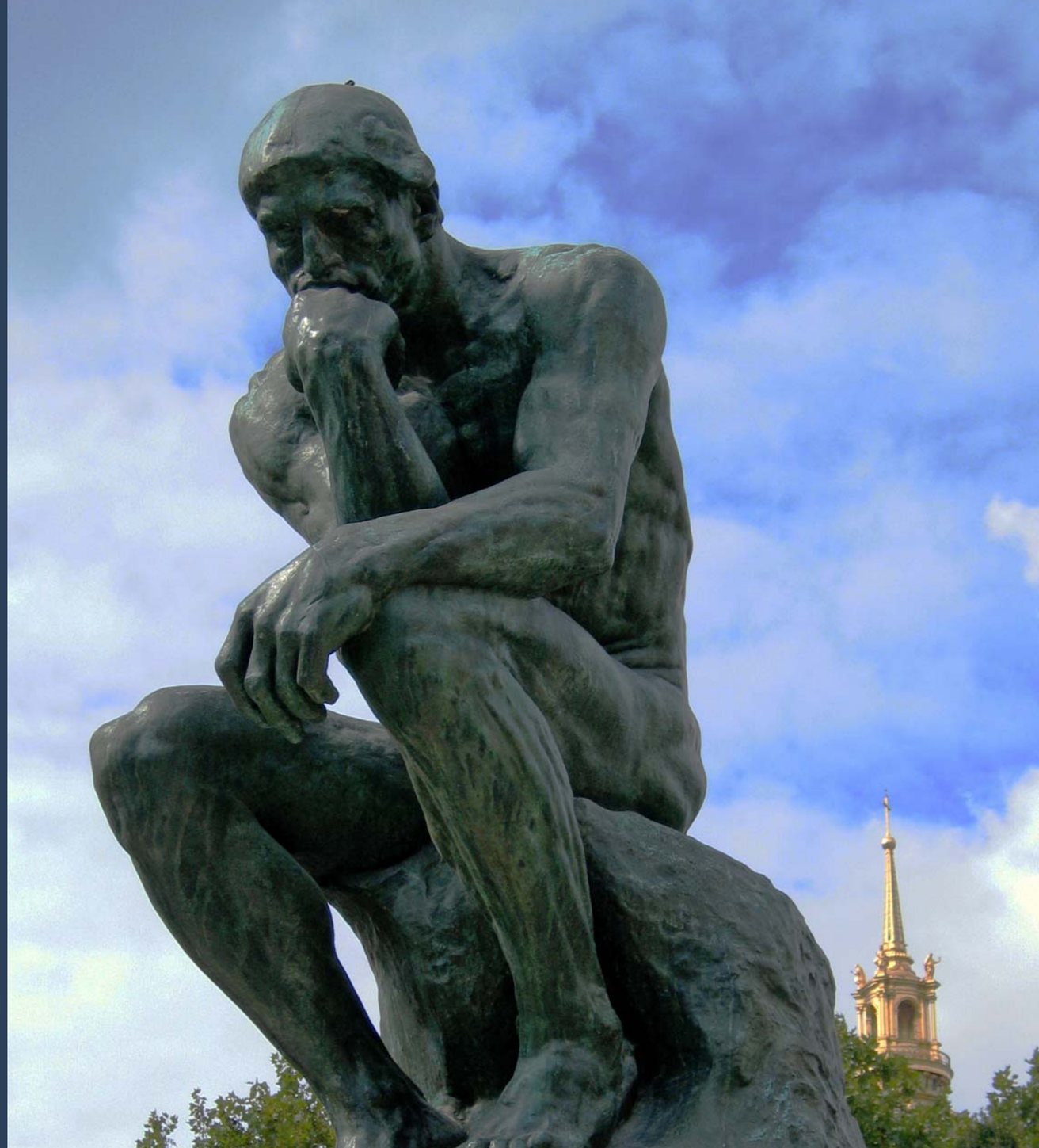
MACHINE CODE
```
B4 09 8D 16 0D 01 CD 21 B8 00 4C
CD 21 48 65 6C 6C 6F 2C 20 77 6F
72 6C 64 21 24
```

Fortran
```
program hello
    write(*,*) "Hello, world!"
end program hello
```

LOLCODE

PROGRAMMING LANGUAGES

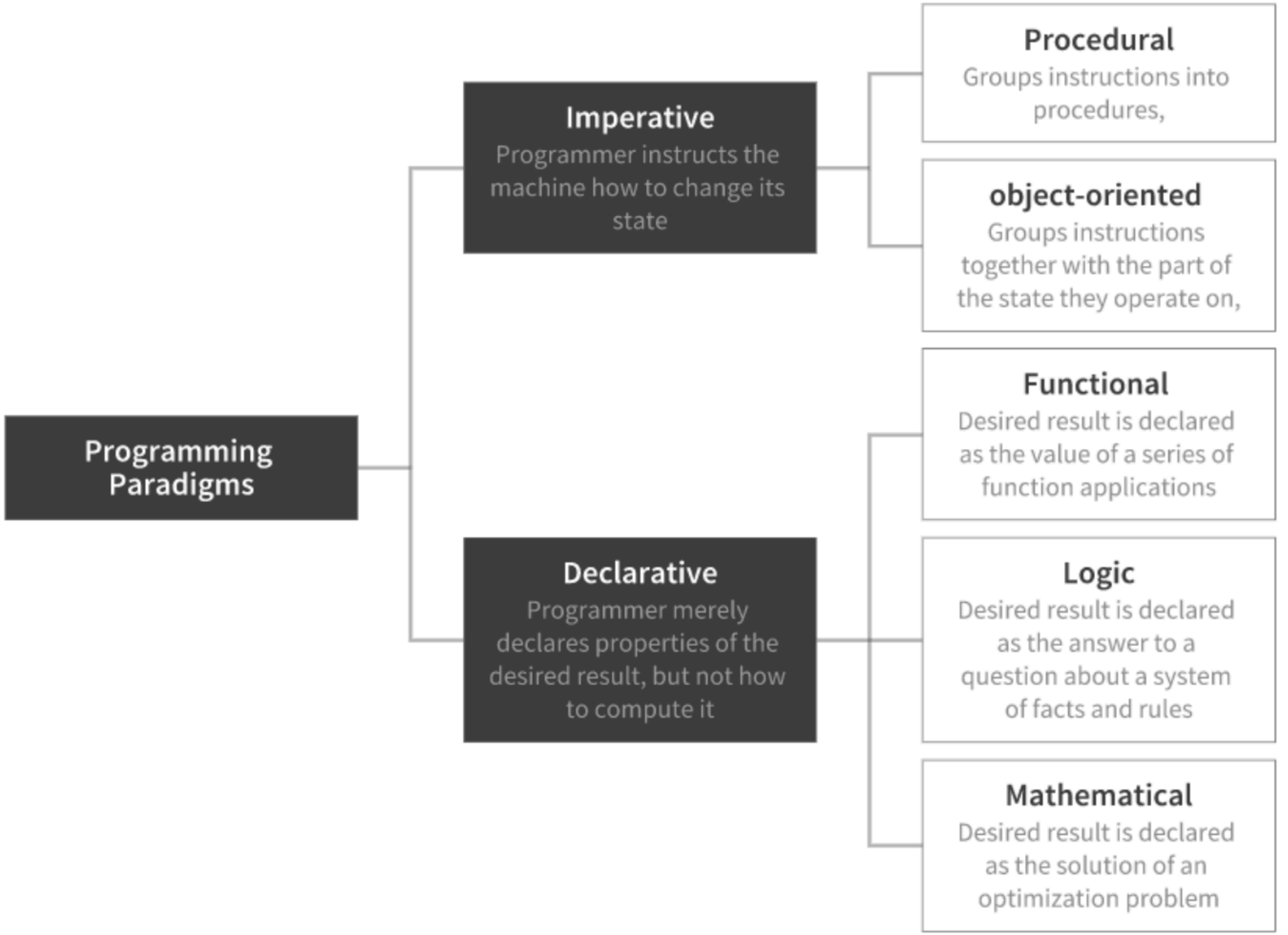
- You will spend more time thinking about how to solve the problem
  - Are you fighting the problem or the language?



PROGRAMMING PARADIGMS

PROGRAMMING PARADIGMS

- Programming paradigms classifies languages and styles based on their features and uses
- In some cases, it just the style in which you program



PROGRAMMING PARADIGMS

- In some cases, it just the style in which you program

```
# Function to compute 5x+2
def f(x):
    return 5*x+2

f(3)
17
```

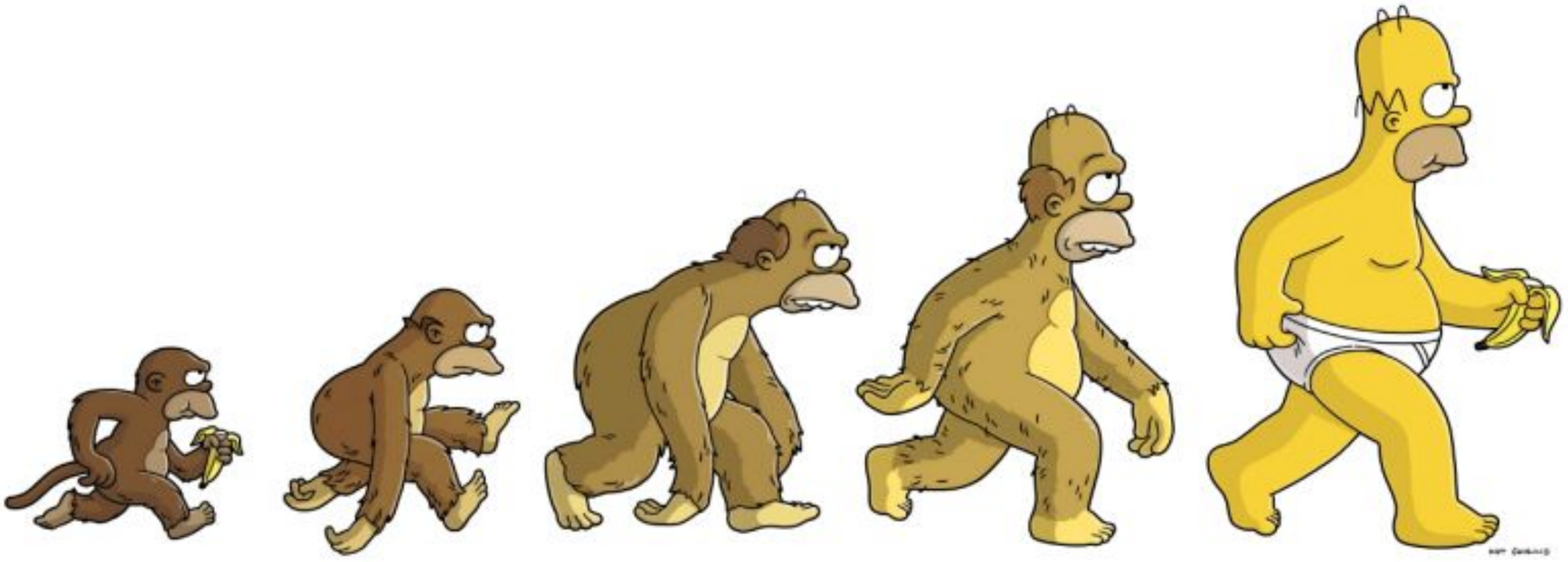
PROCEDURAL

FUNCTIONAL

```
# Function to compute 5x+2 using Lambda function

>>> lambda x : 5*x+2
<function <lambda> at 0x101767ae8>
```

```
>>> g = lambda x : 5*x+2
>>> g(3)
17
```

PROGRAMMING PARADIGMS

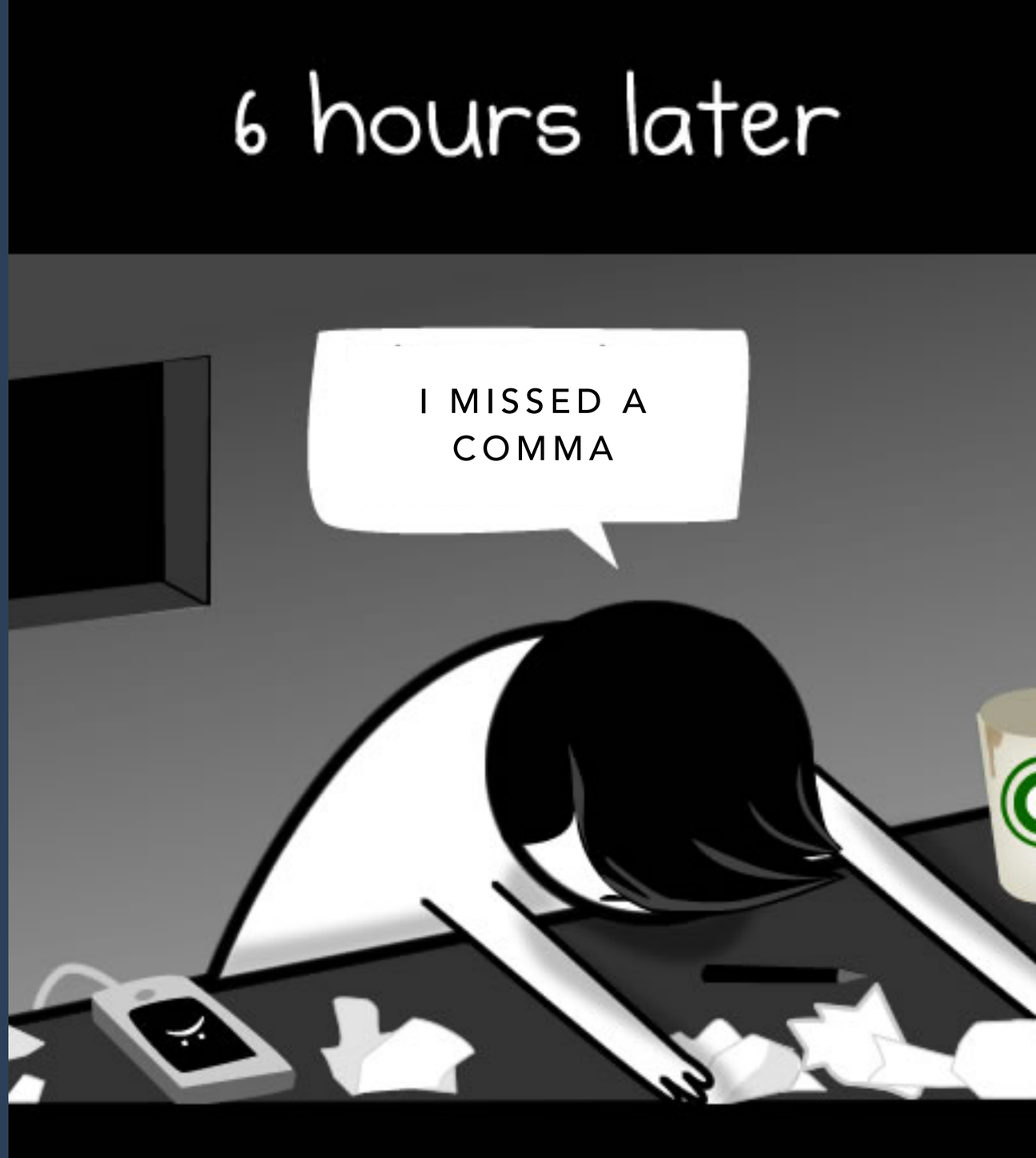


DEBUGGING

(EVERYTHING THAT MAY DRIVE YOU MAD WHILE PROGRAMMING)

DEBUGGING

- Programming requires absolute precision and attention to detail



DEBUGGING

- Programming requires absolute precision and attention to detail

DEBUGGING

- Term "bug" coined by **Grace Hopper** in 1946
- A moth was discovered to be the cause of a malfunction in the Mark II naval computer

DEBUGGING

- Types of errors in a program
  - Syntax
  - Runtime
  - Semantic

DEBUGGING

- Syntax errors
  - Error is related to the syntax used and/or structure of the program
  - First error found is reported and quits
  - Highlights problem with "^"
- Interpreters are not forgiving

```
>>> 1st_name = "Ada"
  File "<stdin>", line 1
    1st_name = "Ada"
             ^
SyntaxError: invalid syntax
```

DEBUGGING

- Runtime errors
  - Errors that only occur while program is running
  - Refers to line of error
  - Describes error

```
>>> x = 100

>>> y = 0

>>> x/y

Traceback (most recent call
last):
  File "<stdin>", line 1, in
<module>

ZeroDivisionError: integer
division or modulo by zero
```

DEBUGGING

- Semantic (logical) errors
  - Program runs but does not do what you expected
  - The instructions you gave it did not solve the problem

```
# Evaluate if the value
# of the variable x is equal
# to 2

x = 2
if x == 3:
  print("x equals 2")
```

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 Kernighan

SETTING UP A DEVELOPMENT ENVIRONMENT

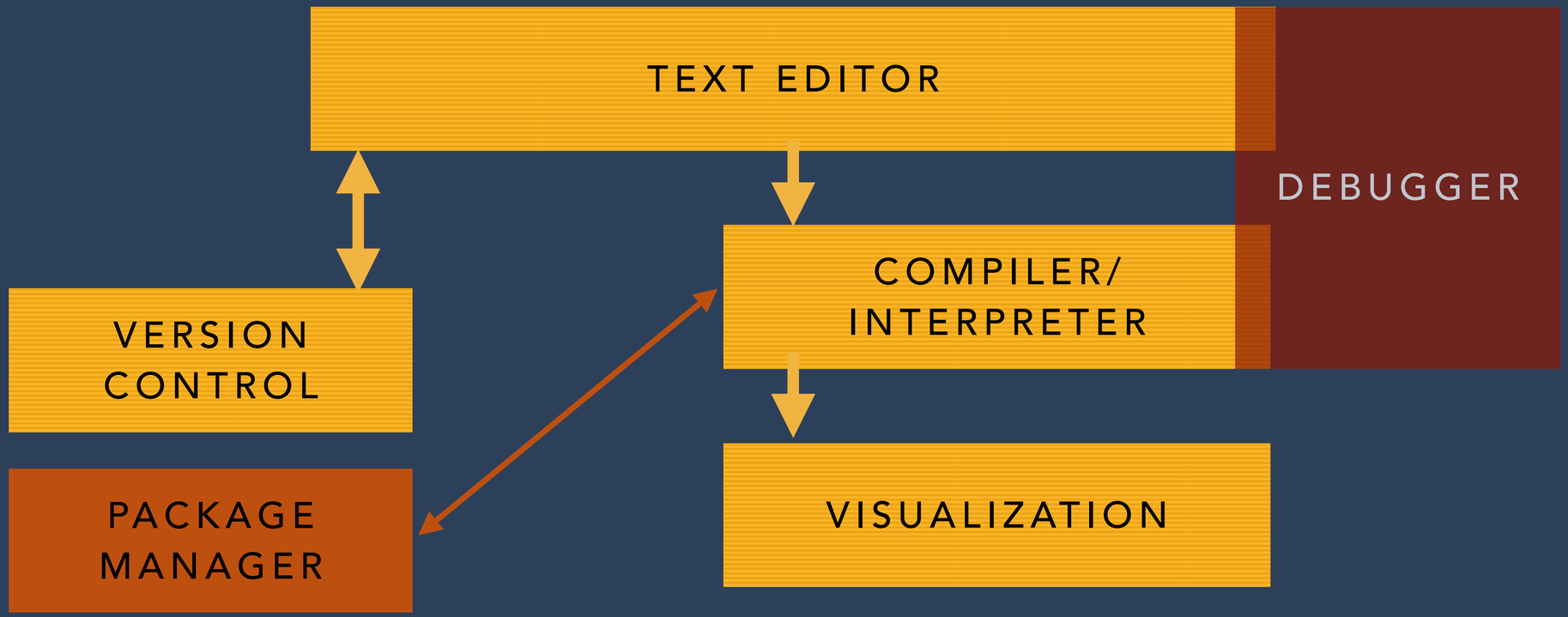
MODULE 1
MPCS 50101

THE UNIVERSITY OF CHICAGO

DEVELOPMENT ENVIRONMENT

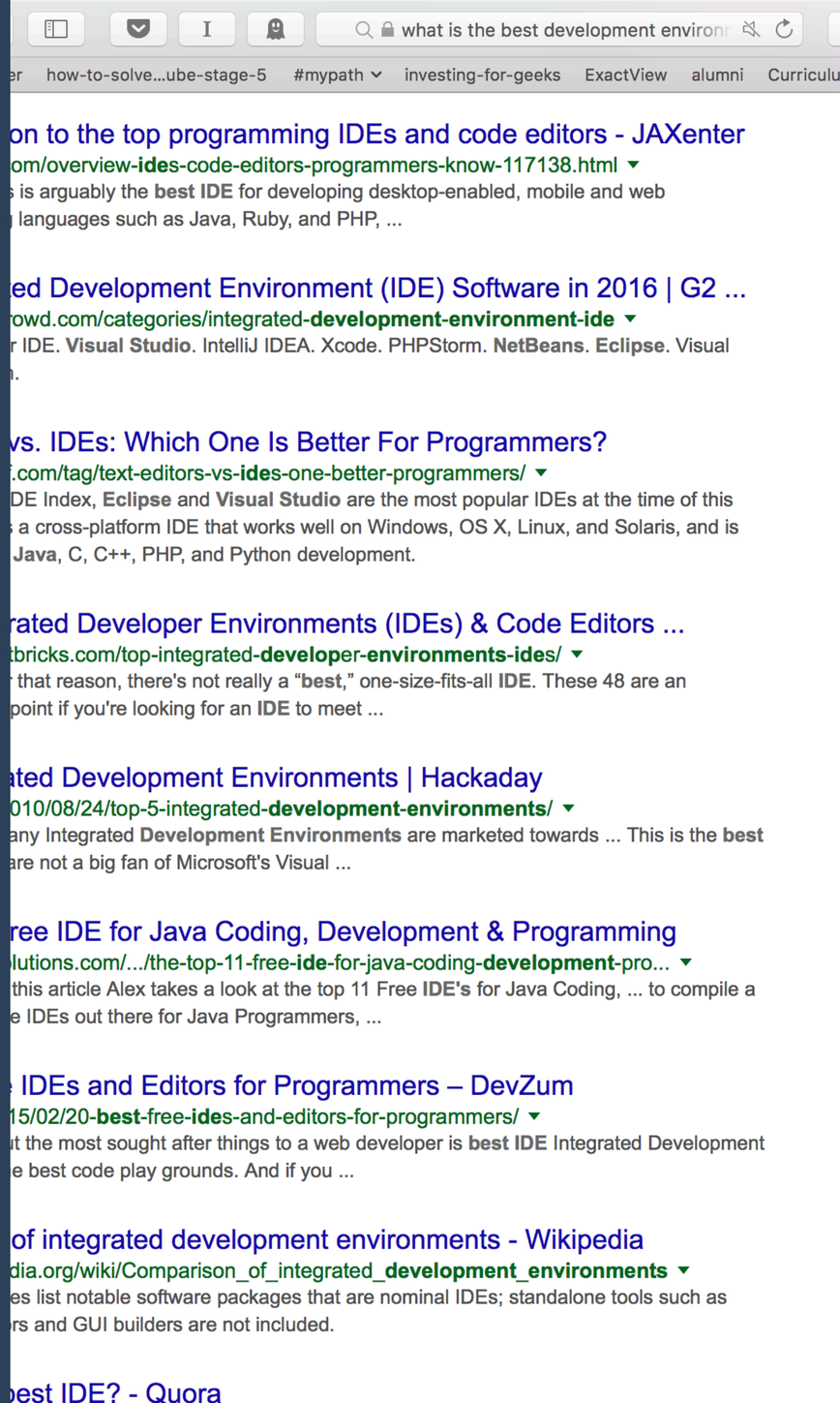
- **Development environment** - The tools you will use to develop software
- Some are:
  - Part of your operating system (eg. bash, tar, zip, git)
  - Universal(eg. text editor)
  - Specialized to a platform (eg. iOS, Android)
  - Specialized to a task (eg. Photoshop, Unity)

DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT

- This course will use universal tools that will be useful throughout the program
  - Exposed to many different tools along the way
- Development environment will be come very personal to your own preferences
  - There is no "best" (although there is no shortage of opinions)



DEVELOPMENT ENVIRONMENT

What are the best programming text editors? - Slant
https://www.slant.co/topics/12/~programming-**text-editors**
A **text editor** is a program that is used for the purpose of editing plain text files. In the context of this question, a programming **text editor** is used for writing code ...
Vim · Spacemacs · Atom · Visual Studio Code

10+ Best Text Editors For Programming 2016 - Livecoding.tv Blog
blog.livecoding.tv/10-**best-text-editors**-programming-2016/
May 23, 2016 - ... Editors can be hard to choose. There are plenty of them in the market. In this article, we list the **best text editors** in 2016 for you. Find the best!

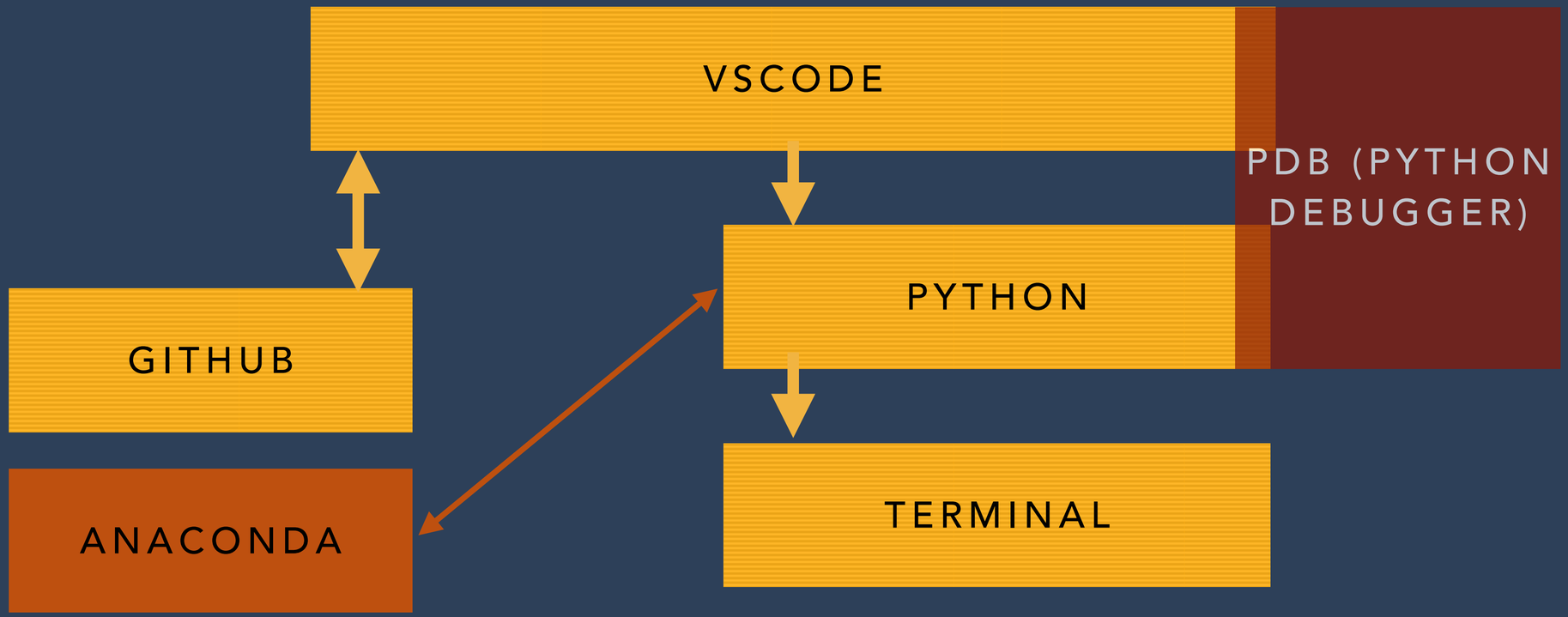
18 great text editors for web designers | Creative Bloq
www.creativebloq.com/.../10-great-**text-editors**-web-designers-71412411
Aug 18, 2015 - If we've missed your favourite, let us know about the app in the comments and why you think it's great. **Notepad**++ Platform: Windows. Atom. Platform: OSX 10.8 or later, Windows 7 & 8, Linux. Brackets. Platform: Windows/OS X/Linux. **Sublime Text**. Platform: Windows/OS X/Linux. **Vim**. **Emacs**. Komodo Edit. Buffer Editor.

Best Text Editor? Atom vs Sublime vs Visual Studio Code vs Vim ...
https://www.codementor.io/.../**best-text-editor**-atom-sublime-vim-visual-s...
Sep 28, 2016 - While a straightforward answer to the "what is the **best text editor** for developers?" question doesn't exist, in this post, I will share with you a side ...

Sublime Text: The text editor you'll fall in love with
https://www.sublime**text**.com/
Sublime Text is a sophisticated **text editor** for code, markup and prose. You'll love the slick user interface, extraordinary features and amazing performance.

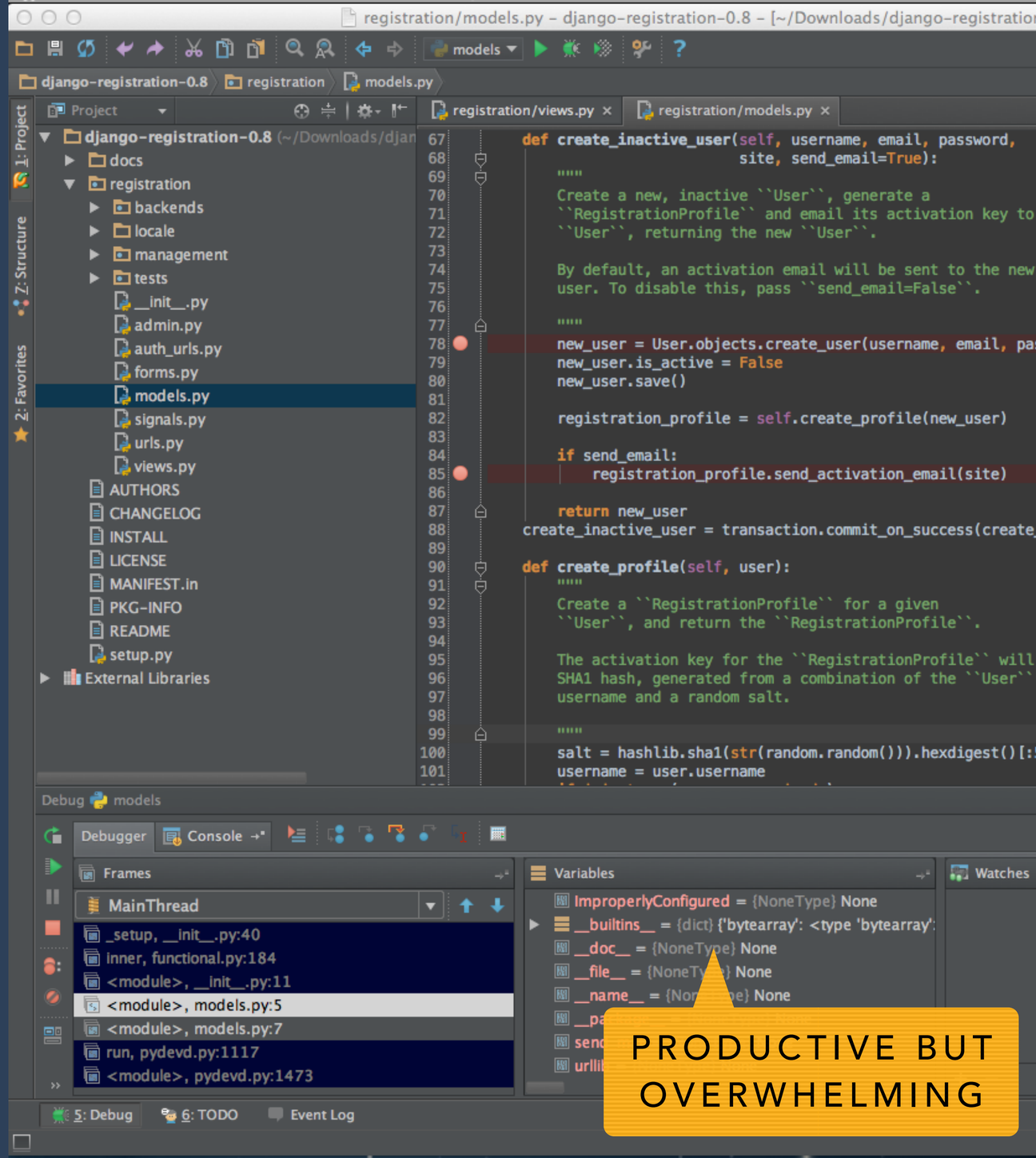
VERY PERSONAL CHOICE

DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT

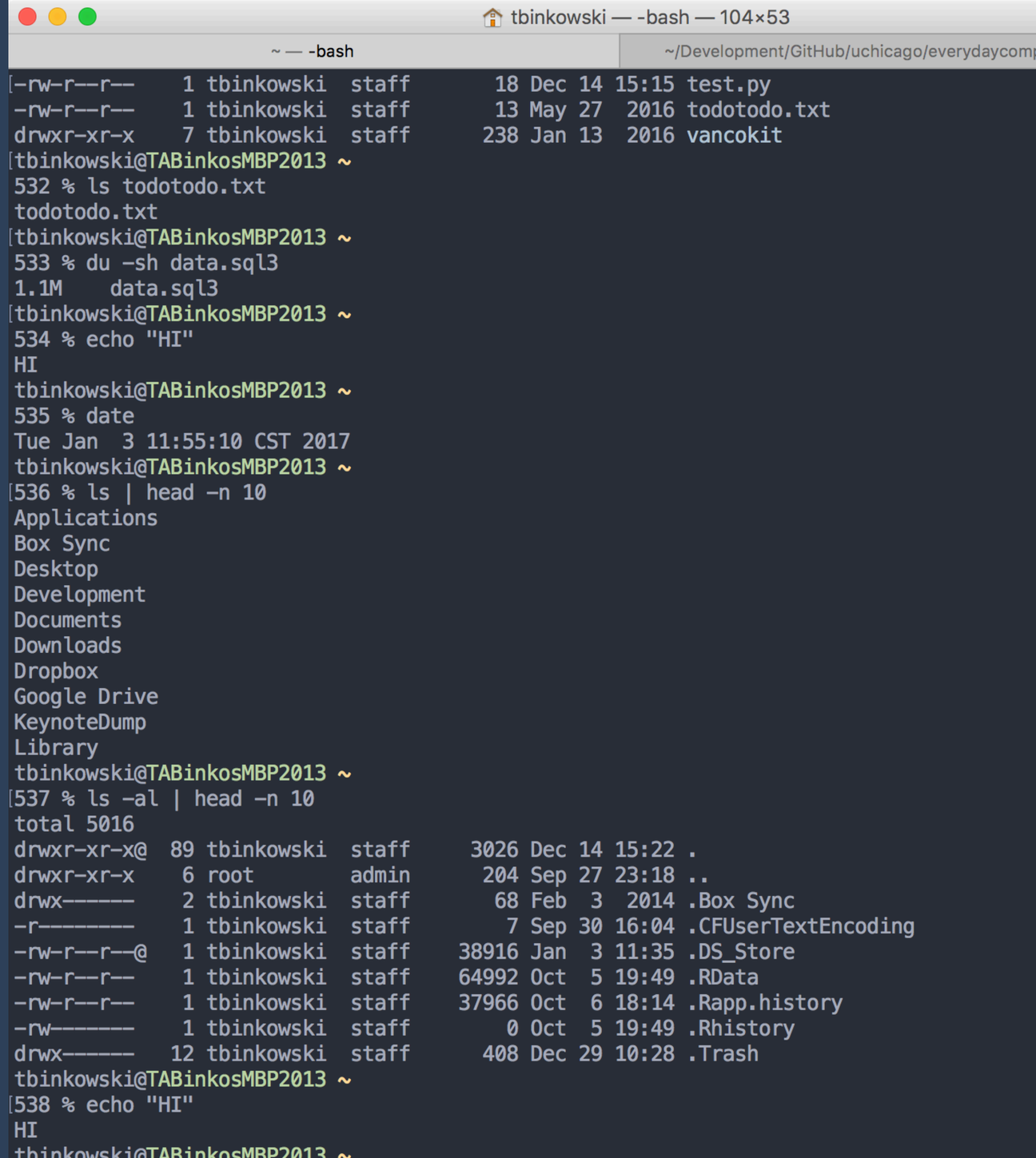
- We are going to be a universal development environment
- IDE (Integrated Development Environment)
  - Puts all development tools in a single interface
  - Specialized for languages (PyCharm) or platforms (Xcode)



SHELL 

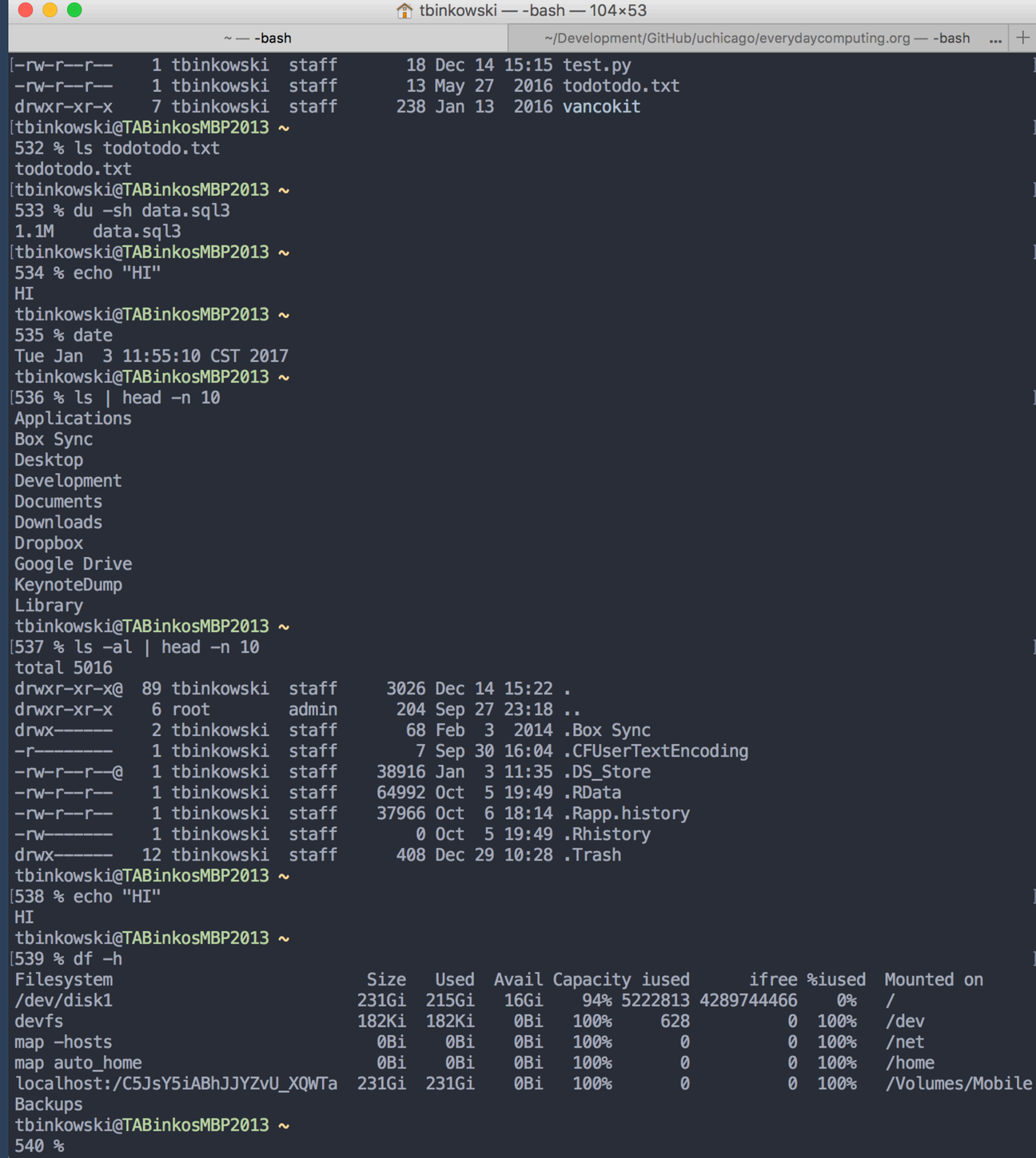
SHELL

- A Unix shell is an interpreter that provides a command line user interface
  - Text commands to execute
  - Create scripts of one or more such commands
- Shells
  - Bash,tcsh,zsh



SHELL

- A terminal window allows the user access to a text terminal and all its applications
  - Command-line interfaces (CLI)
- These may be running either on the same machine or on a different one via telnet, ssh
  - How you interface with servers, clusters, etc.



SHELL

- Essential shells commands
- We'll learn more during the course, but you should use these
- Pay attention to flags, especially "-i" on rm

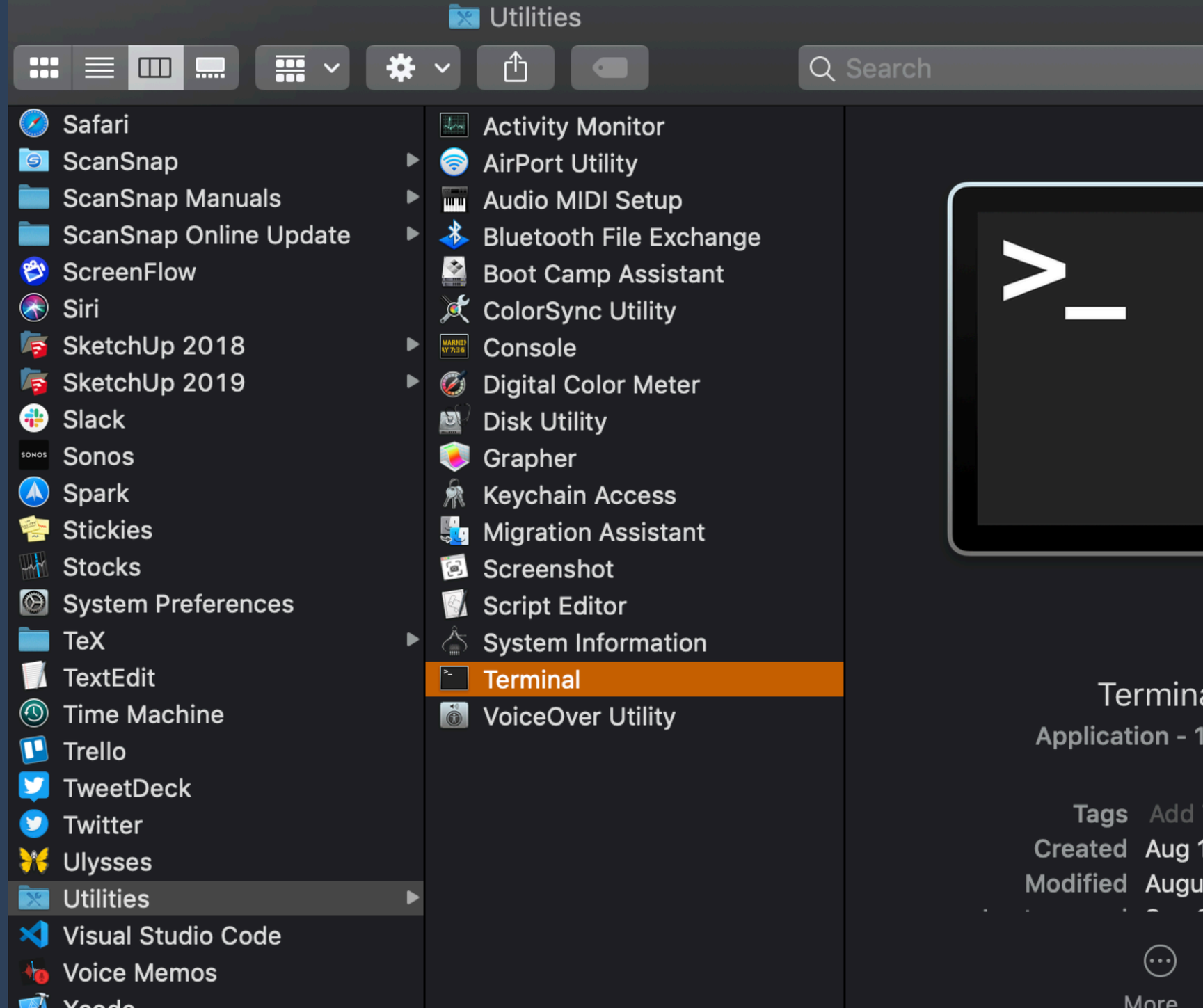
```
% ls
% cd
% mv
% mkdir
% rmdir
% man
% touch
% rm -i
% clear
% cp
% which
% whoami
```

SHELL

```
% touch "myfile.txt"

% cp myfile.txt ~/Documents/myfile.txt

% cd ~/Documents/

% pwd

% ls

% rm myfile.txt
```

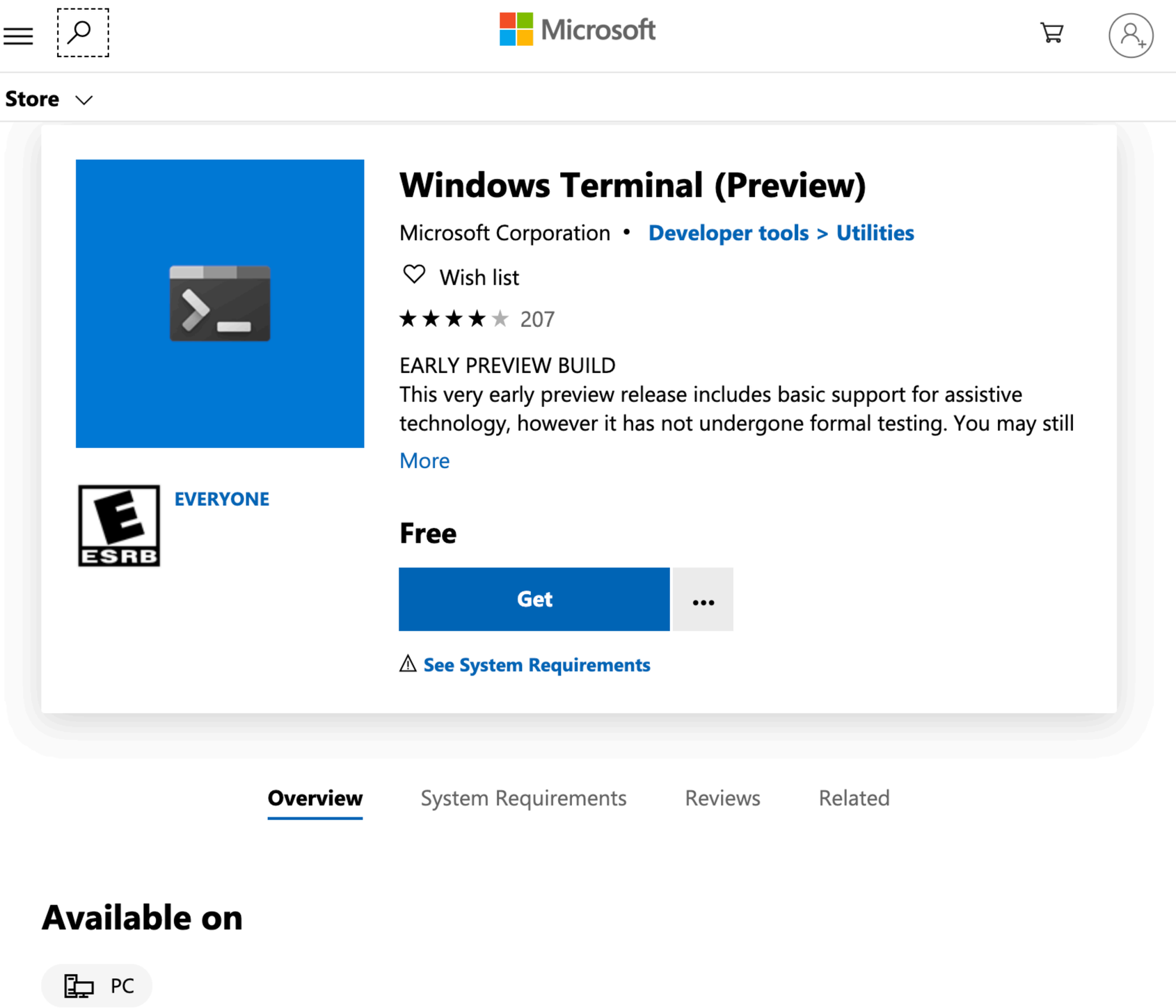
SHELL

- Where is it?



SHELL

- Powershell

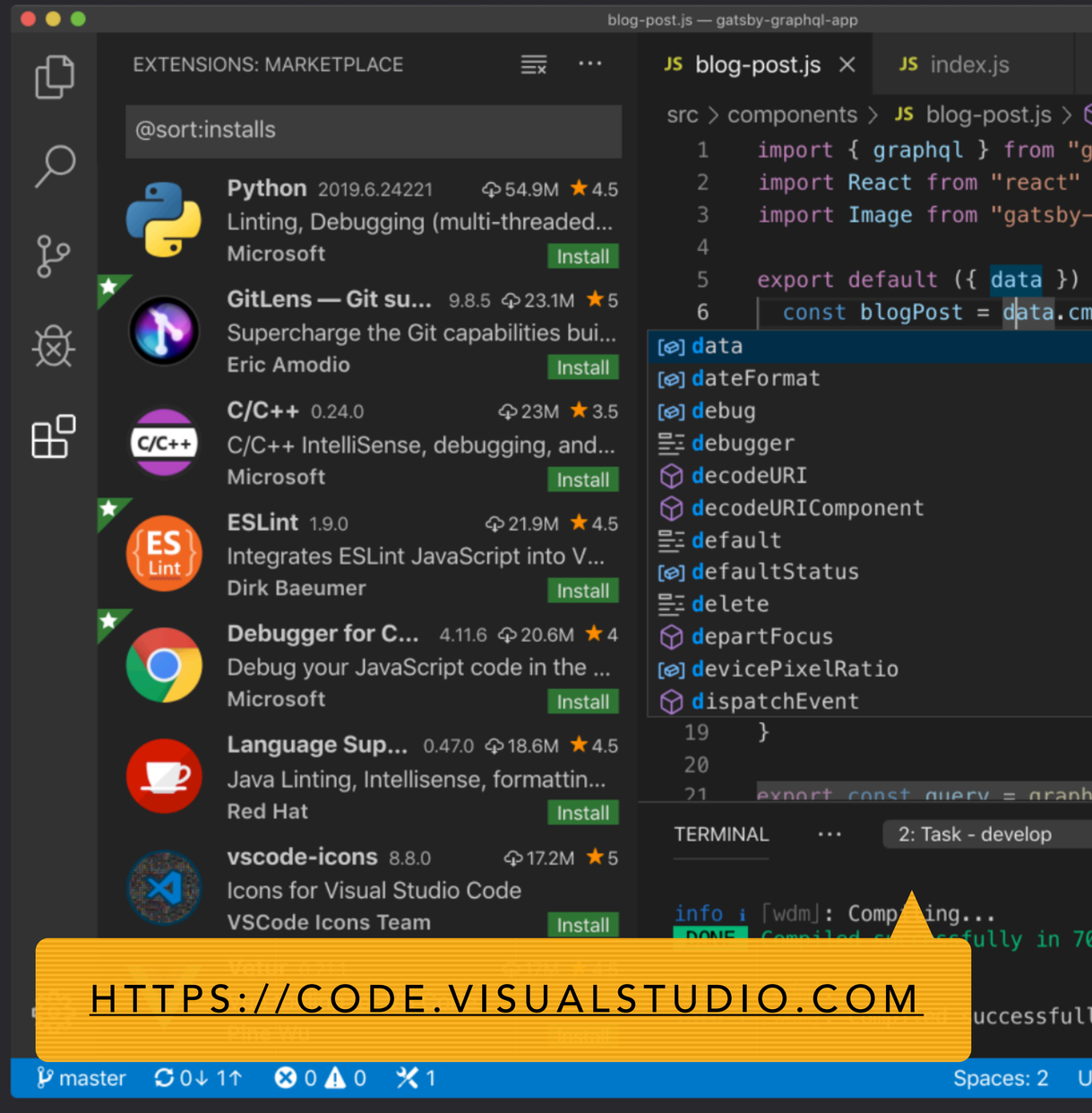


TEXT EDITOR

DEVELOPMENT ENVIRONMENT

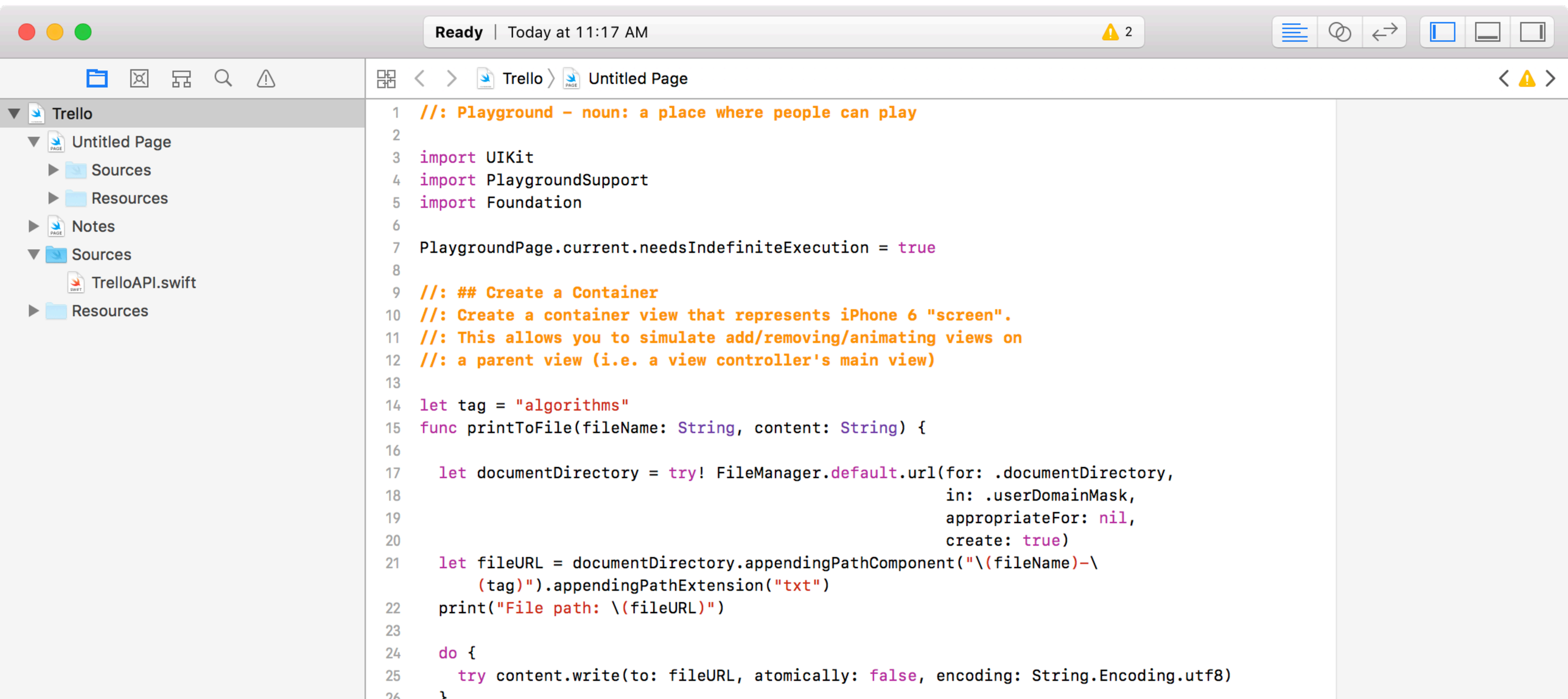
- Text Editors
  - Emacs/VI
  - BBEdit, Text Wrangler, Sublime, TextEdit, Atom, Visual Studio Code
  - IDE: Xcode, PyCharm, Android Studio

Visual Studio Code

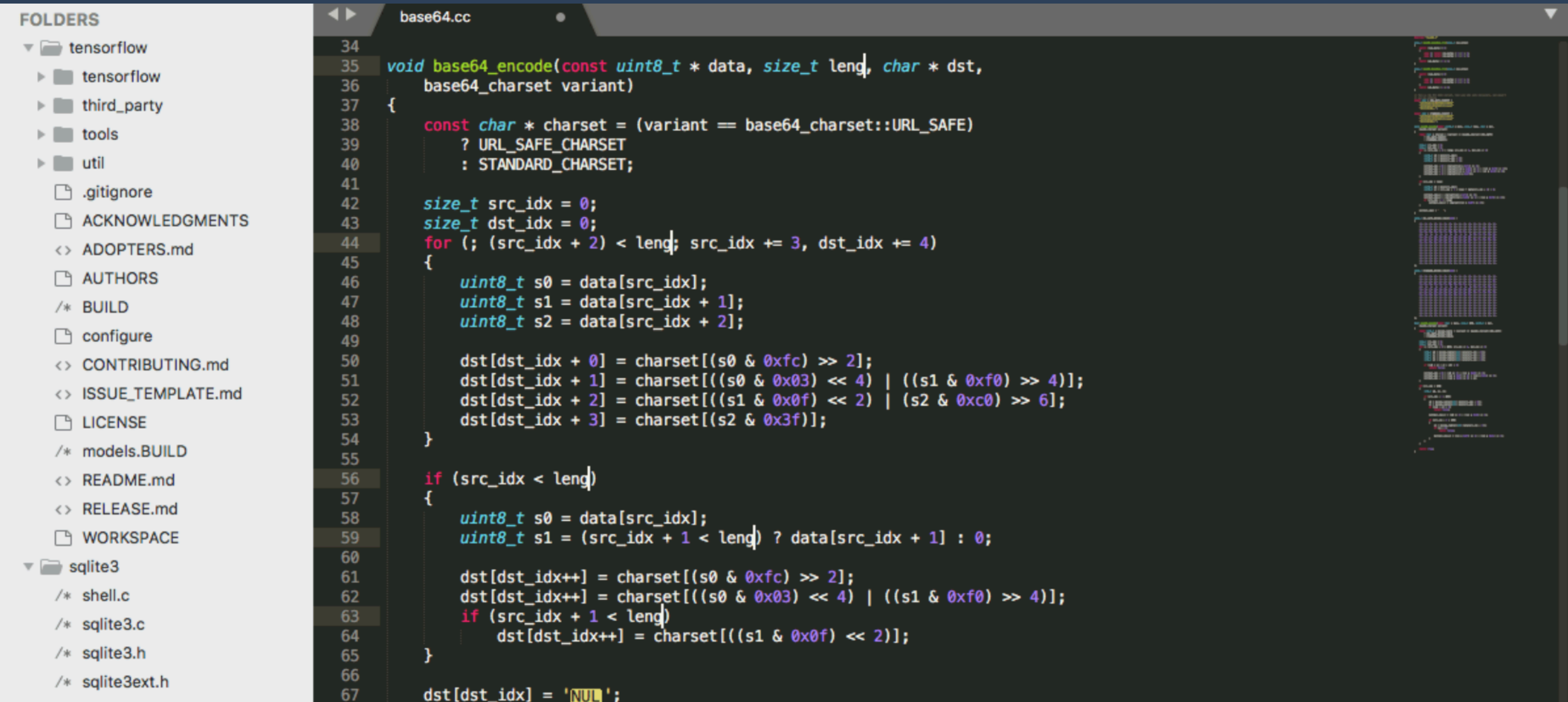


HTTPS://CODE.VISUALSTUDIO.COM

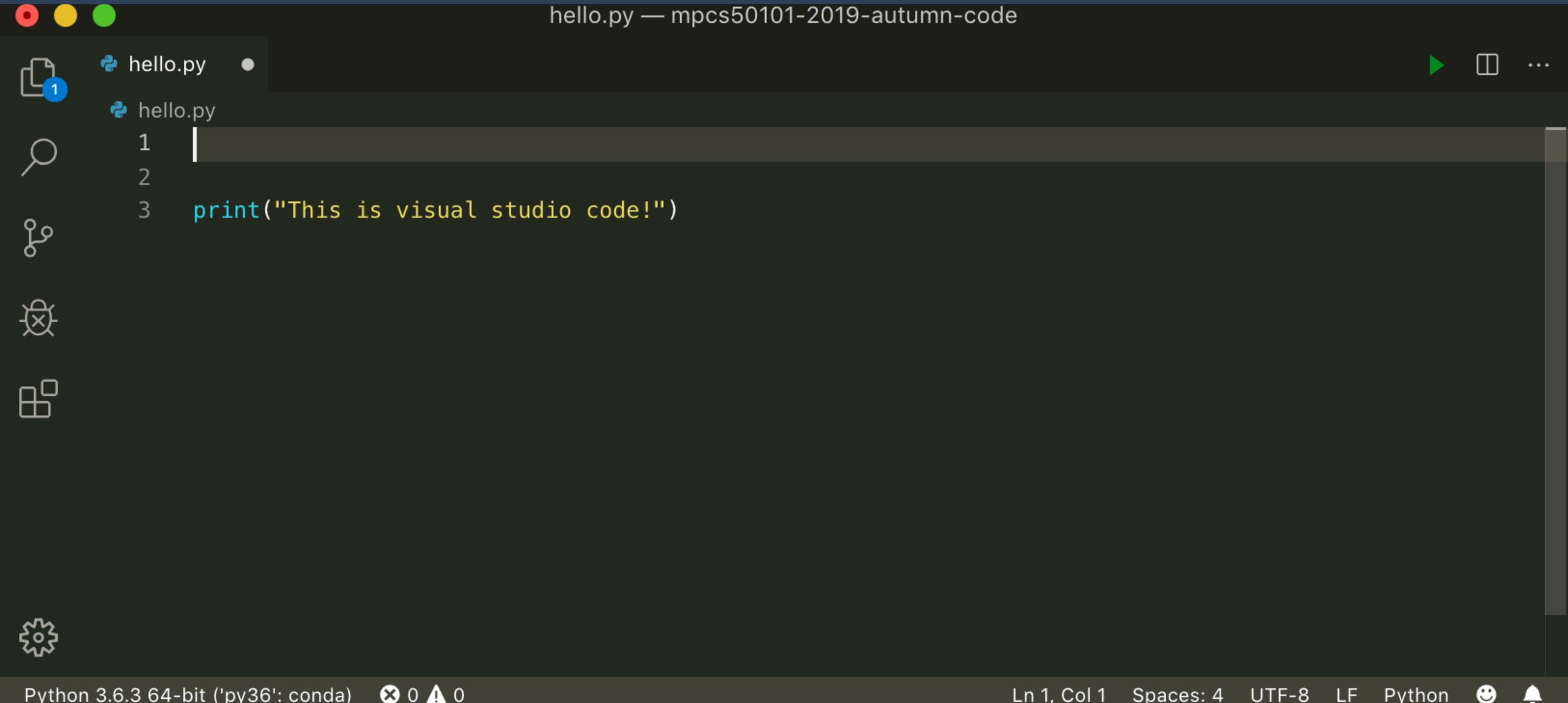
DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT

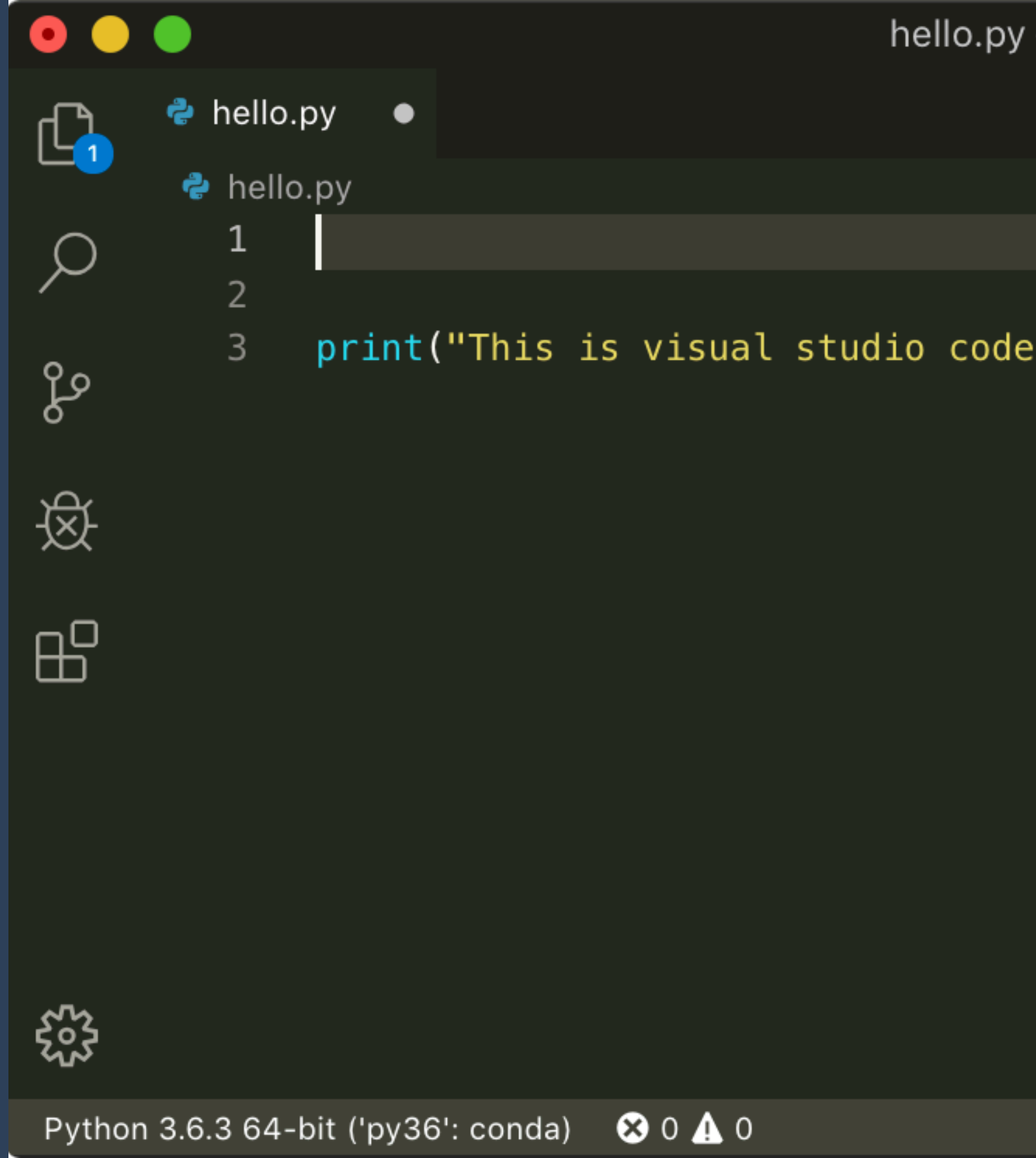


DEVELOPMENT ENVIRONMENT

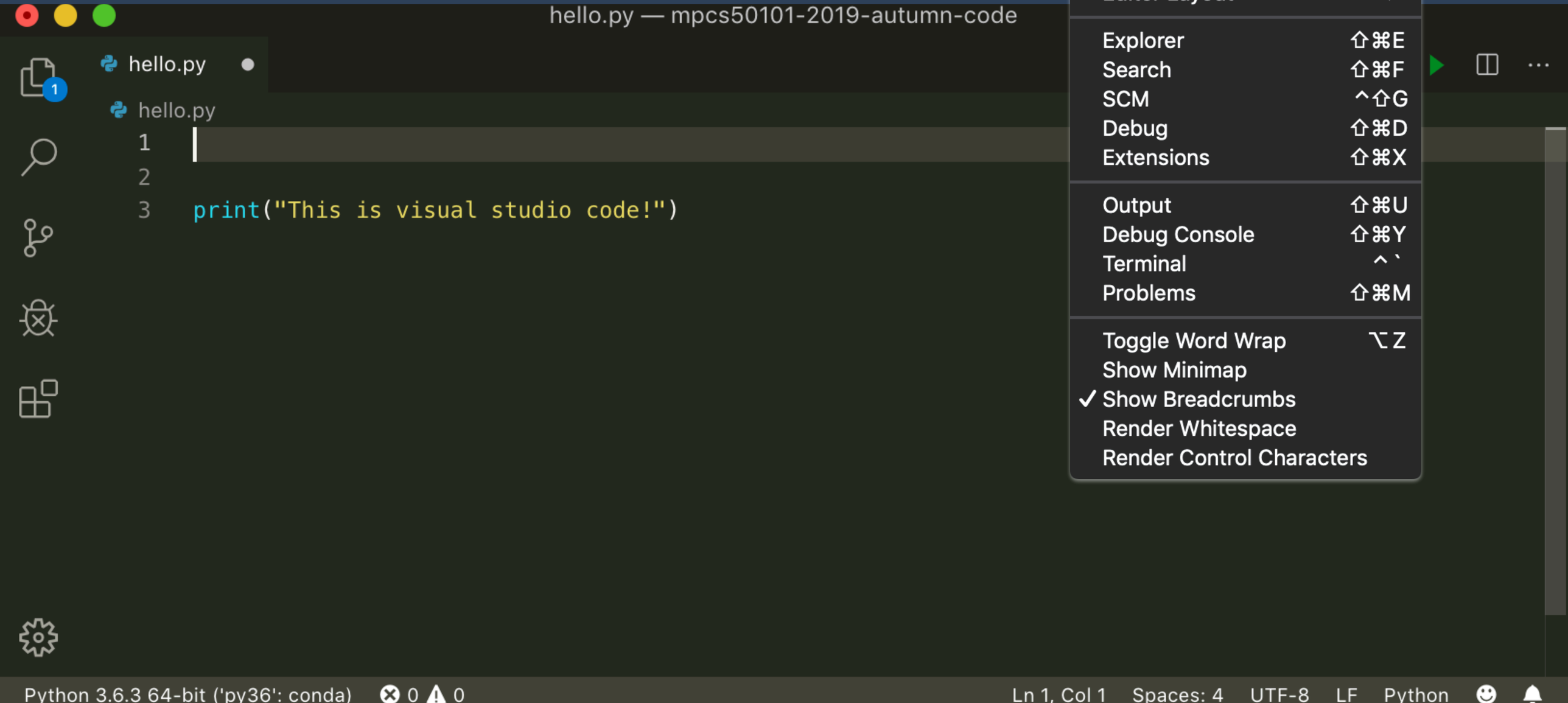


DEVELOPMENT ENVIRONMENT

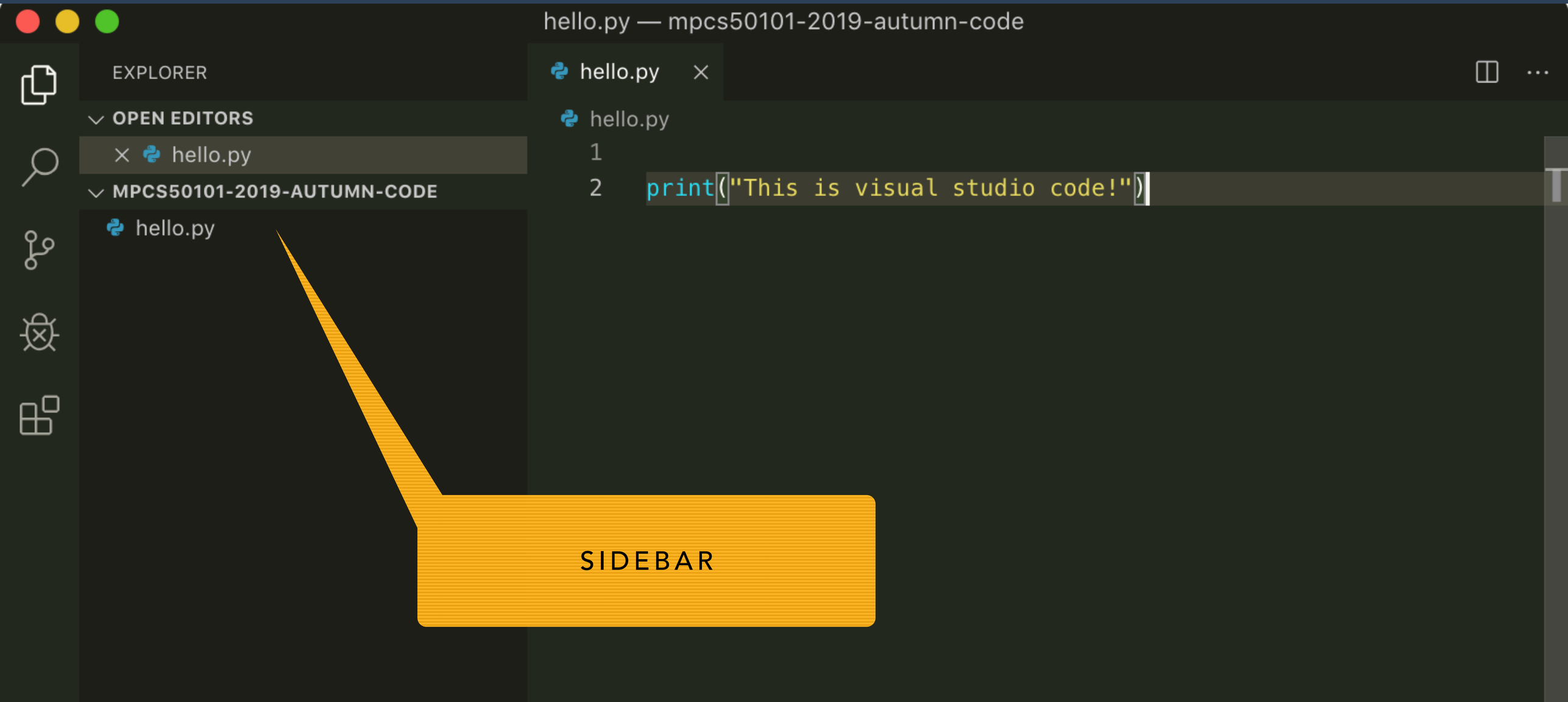
- Visual Studio Code text editor by Microsoft
- Highly customizable editor can be used by any programming language
- Extend the functionality by adding packages



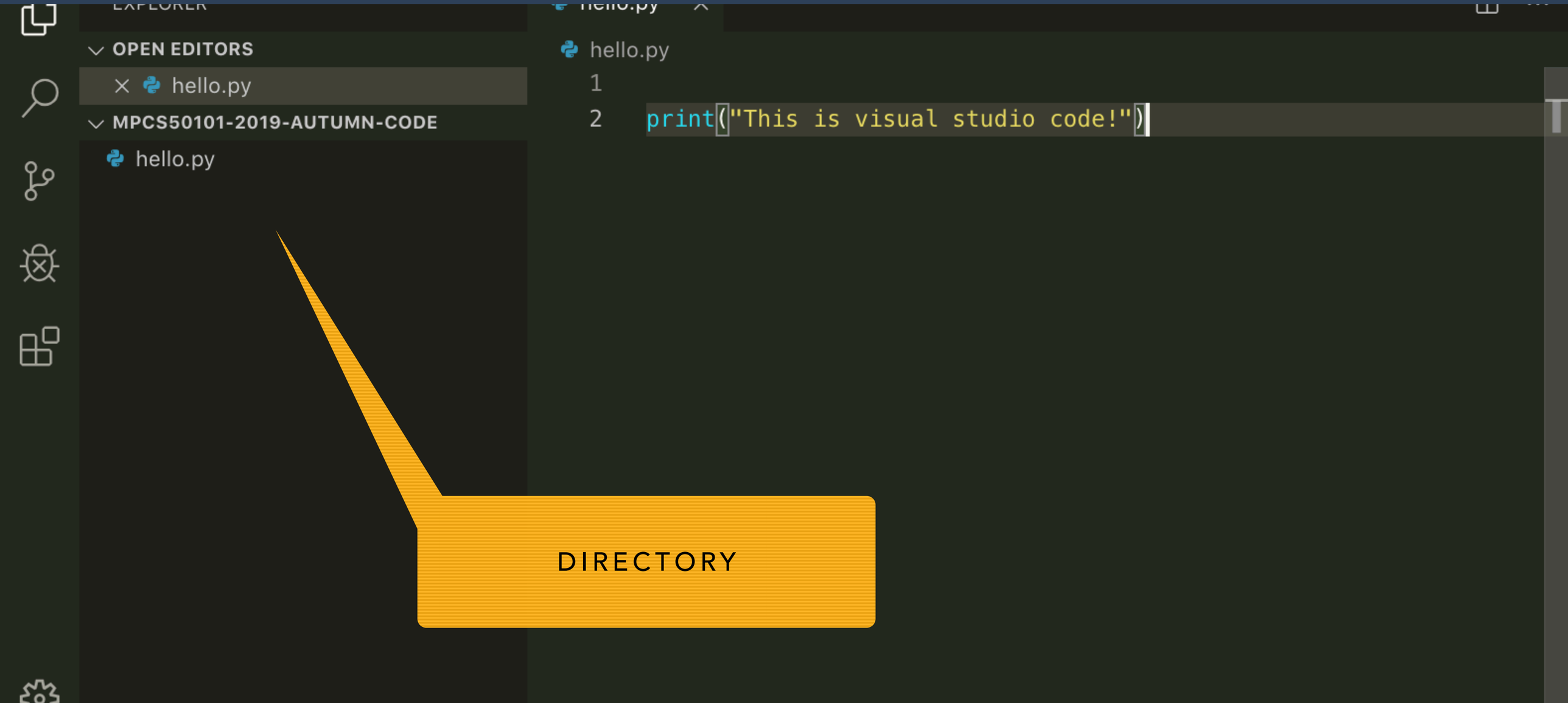
DEVELOPMENT ENVIRONMENT



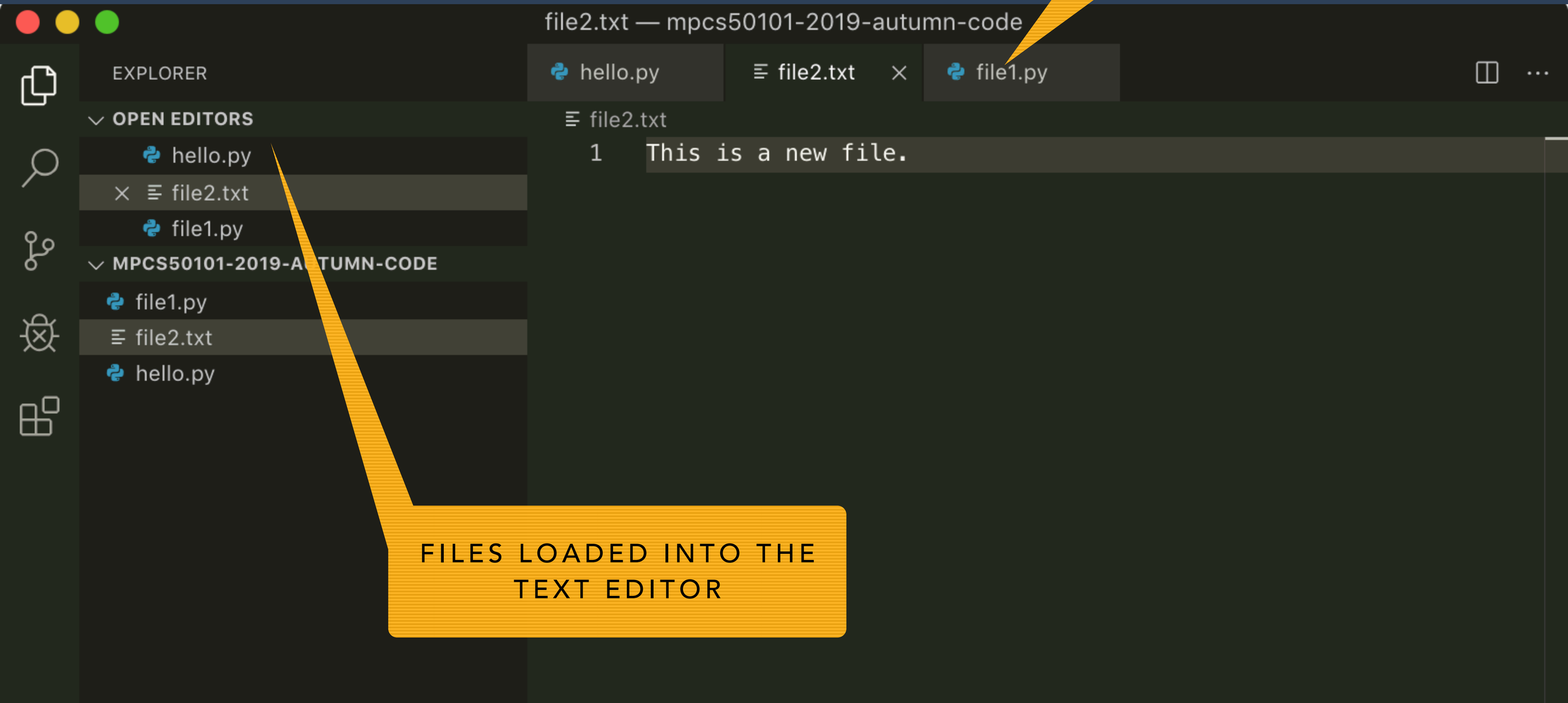
DEVELOPMENT ENVIRONMENT



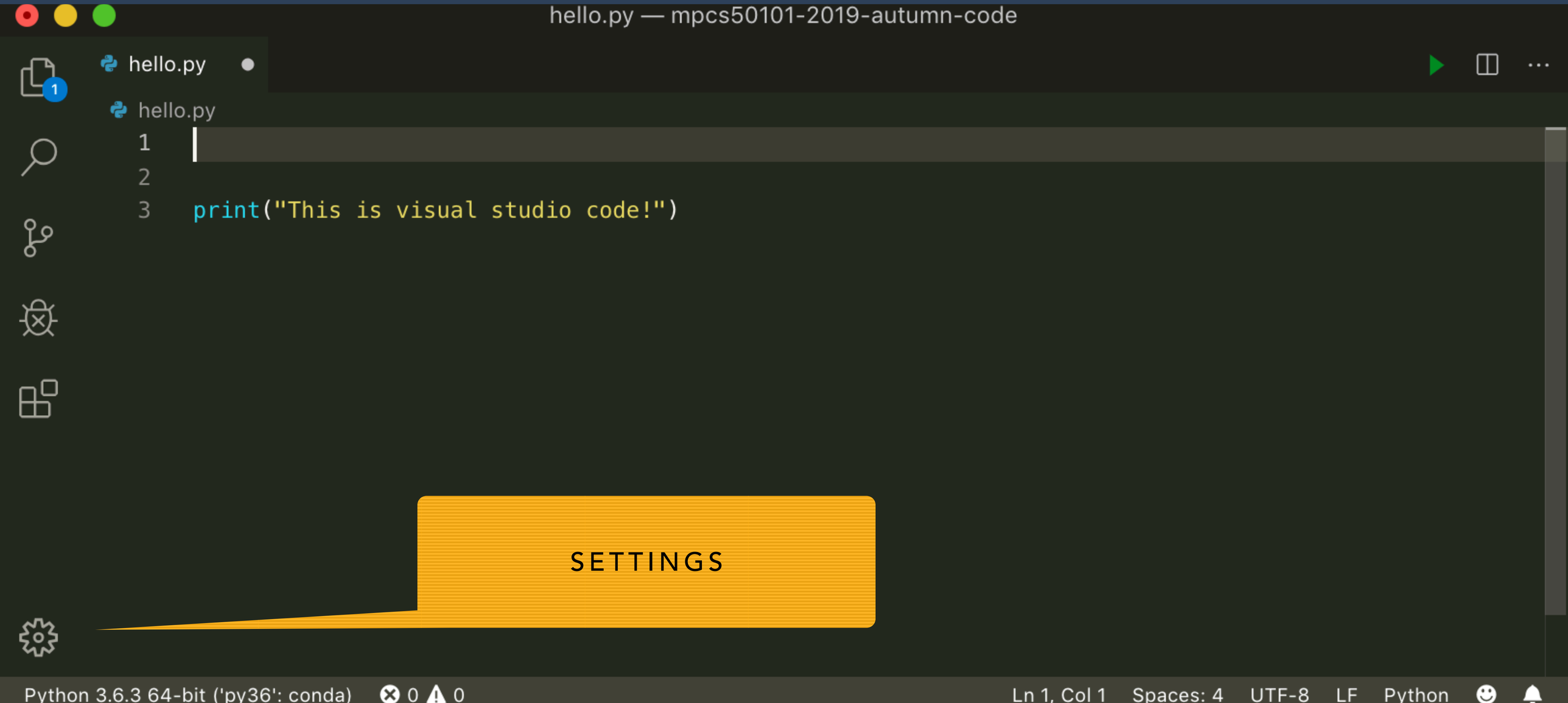
DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT



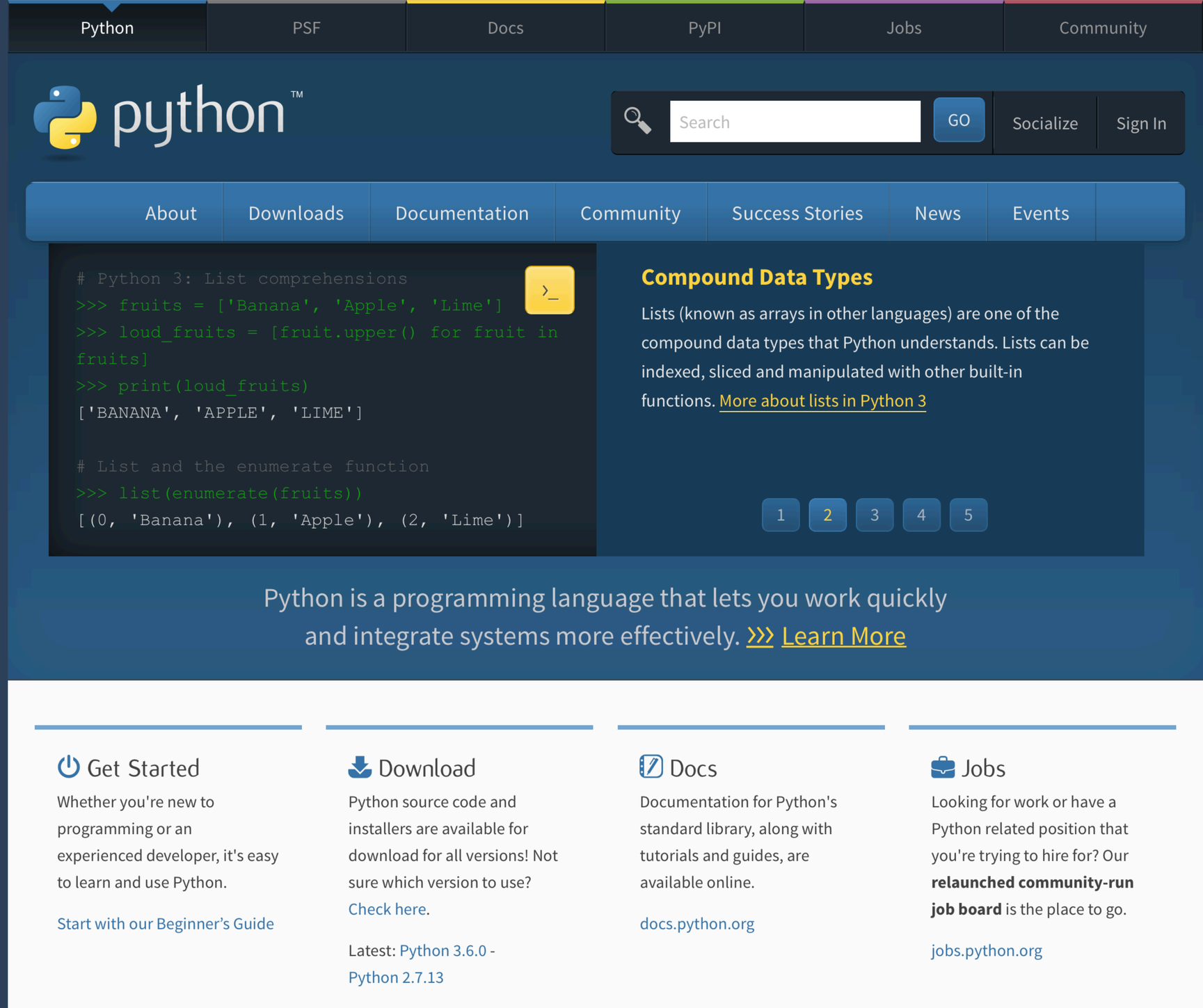
DEVELOPMENT ENVIRONMENT

- Don't install the Python extension
- We'll do it later

PROGRAMMING LANGUAGE

PROGRAMMING LANGUAGE

- We are using Python 3 in this course
  - Not much difference for beginners



PROGRAMMING LANGUAGE

```
# Python 2 vs. 3

# Print is a function in 3; in 2 it was a keyword


# Python 2
print "Hello"
print("Hello")


# Python 3
print("Hello")
```

PROGRAMMING LANGUAGE

```
# Python 2 vs. 3

# Python 2 str are ASCII, Python 3 is Unicode; convert
# 'u' in Python 2


# Python 2
print(u'Hello')


# Python 3
print('Hello')
```

PROGRAMMING LANGUAGE

```
[508 % python3                                                                  ]
 Python 3.7.4 (default, Sep  7 2019, 18:27:02)
 [Clang 10.0.1 (clang-1001.0.46.4)] on darwin
 Type "help", "copyright", "credits" or "license" for more information.
[>>> x="😀"                                                                     ]
[>>> print(x)                                                                  ]
 😀
 >>>
```

Your first program in Python

PROGRAMMING LANGUAGES

- Python is an interpreted language
- The interpreter can be run in two modes
  - Interactive mode
  - Script mode

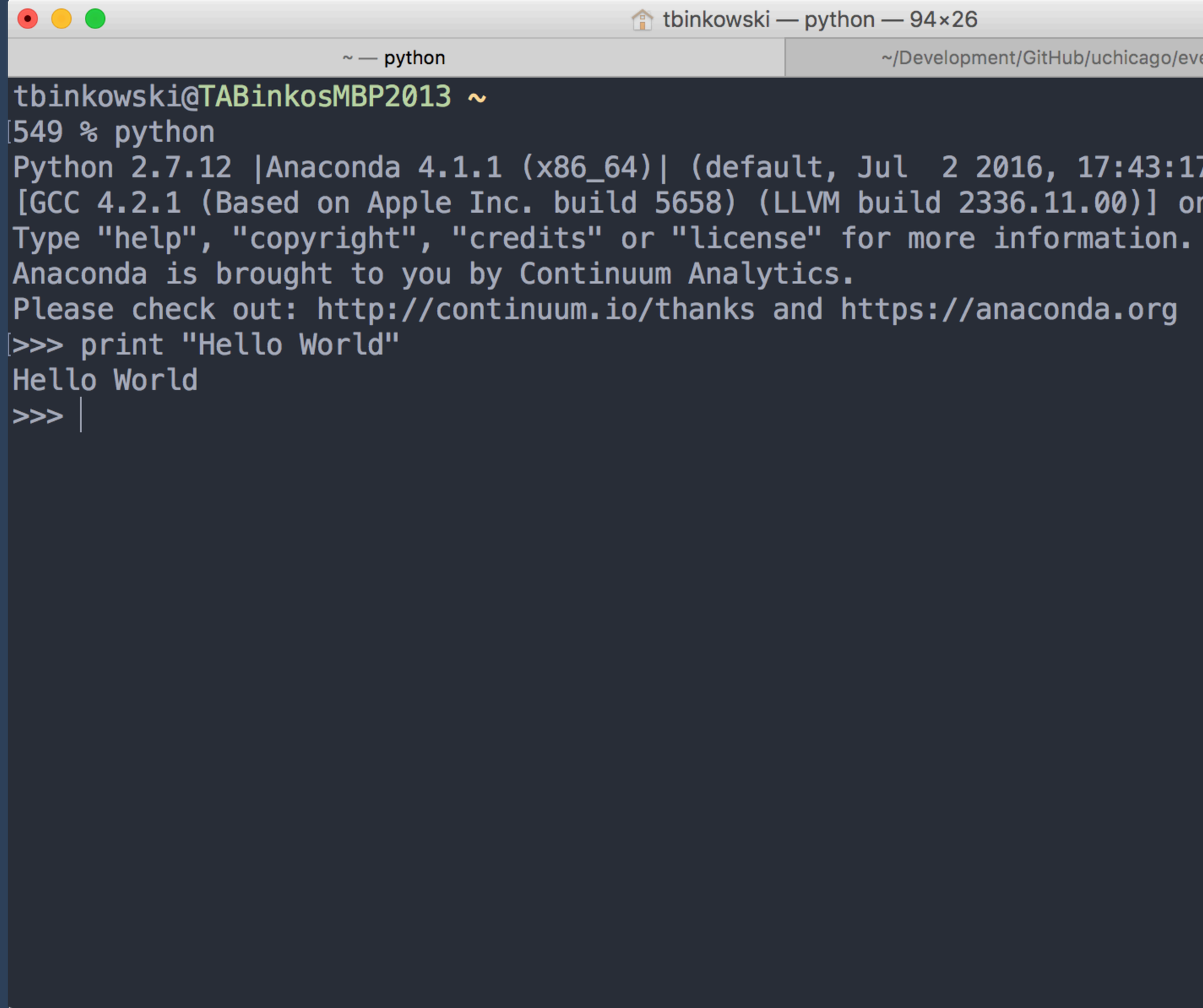
```
510 % python3
Python 3.7.4 (default, Sep  7 2019, 18:27:02)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "licens
>>> print("This is interactive mode")
This is interactive mode
>>>
```

hello.py

```
1    print("This is script mode")
```

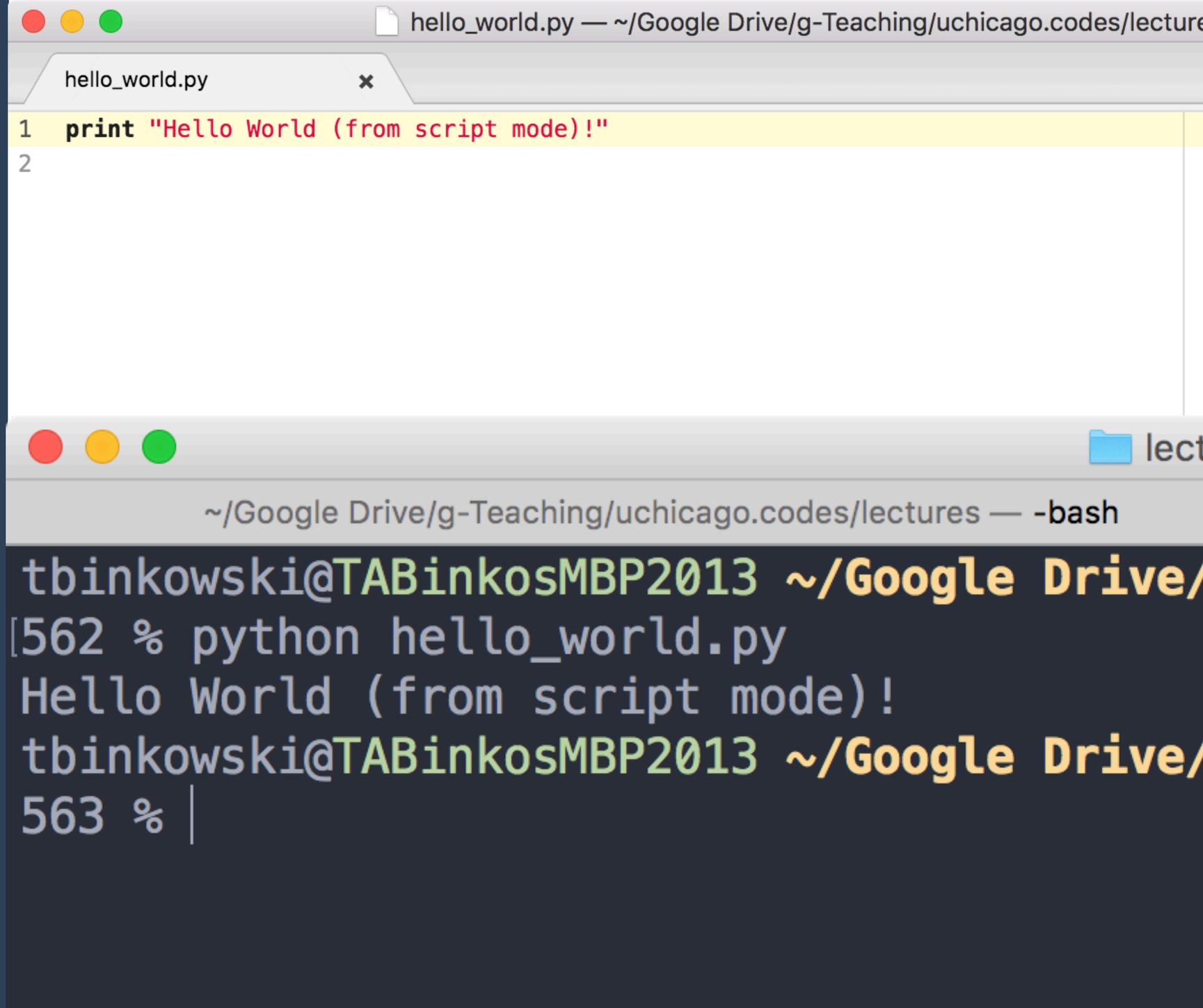
PROGRAMMING LANGUAGES

- Interactive mode executes each line of code
- Type at the prompt ">>>"
- Ctrl-D or exit() to quit



PROGRAMMING LANGUAGES

- Script mode
- Type code in text editor
- Execute from a terminal



PACKAGE MANAGER

DEVELOPMENT ENVIRONMENT

- Python has an active supporting community of contributors and users that also make their software available under open source license terms
  - Allows Python users to share and collaborate effectively
  - Simplifies installation and unifies environment
- Package managers for Python
  - pip
  - homebrew (Mac only)
  - Anaconda

DEVELOPMENT ENVIRONMENT

- Language Python version 3.9
  - Some version is probably installed on your computer
- Anaconda platform manages many packages that will be used throughout the program
  - Statistics, data science
  - Machine learning, AI

DEVELOPMENT ENVIRONMENT

Google

anaconda

Sign in

All Videos Images News Maps More Settings Tools

About 33,800,000 results (0.80 seconds)

Download Anaconda Now! | Continuum
https://www.continuum.io/downloads
Anaconda is the leading open data science platform powered by Python. The open source version of **Anaconda** is a high performance distribution of Python and ...

People also ask

- Where anacondas can be found?
- What is meant by Anaconda in Linux?
- Do anacondas really exist?
- Do Anacondas live in Africa?

Why Anaconda? | Continuum
https://www.continuum.io/why-**anaconda**
Accelerate Time-to-Value with the Leading Open Data Science Platform. Open Data Science leverages the power and innovation from open source communities ...

See results about

Anaconda (Song by Nicki Minaj)
Artist: Nicki Minaj
Album: The Pinkprint

Green anaconda (Snake)
Scientific name: Eunectes murinus
Rank: Species

Anaconda 3: Offspring (2008 film)
Initial release: July 26, 2008
Director: Don E. FauntLeRoy

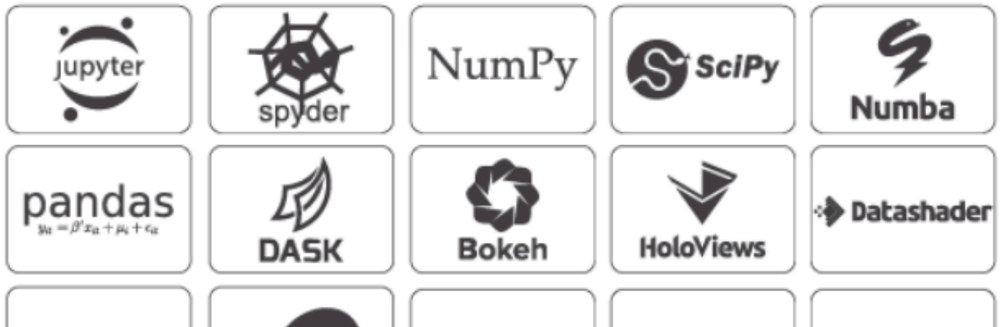
DEVELOPMENT ENVIRONMENT

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

Download

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide,



DEVELOPMENT ENVIRONMENT

DOWNLOAD THE LATEST VERSION

Python 3.7 version

Download

64-Bit Graphical Installer (653 MB)

64-Bit Command Line Installer (435 MB)

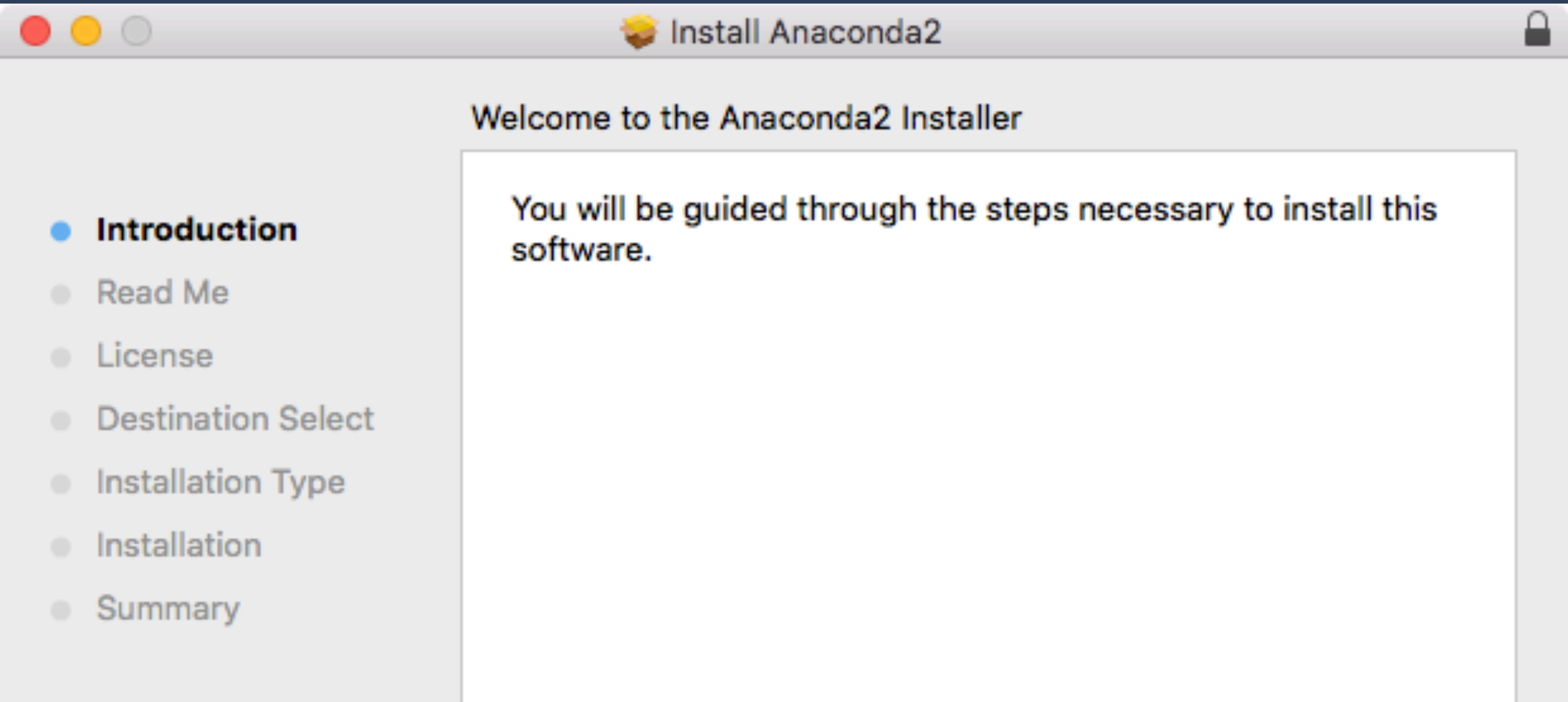
Python 2.7 version

Download

64-Bit Graphical Installer (634 MB)

64-Bit Command Line Installer (408 MB)

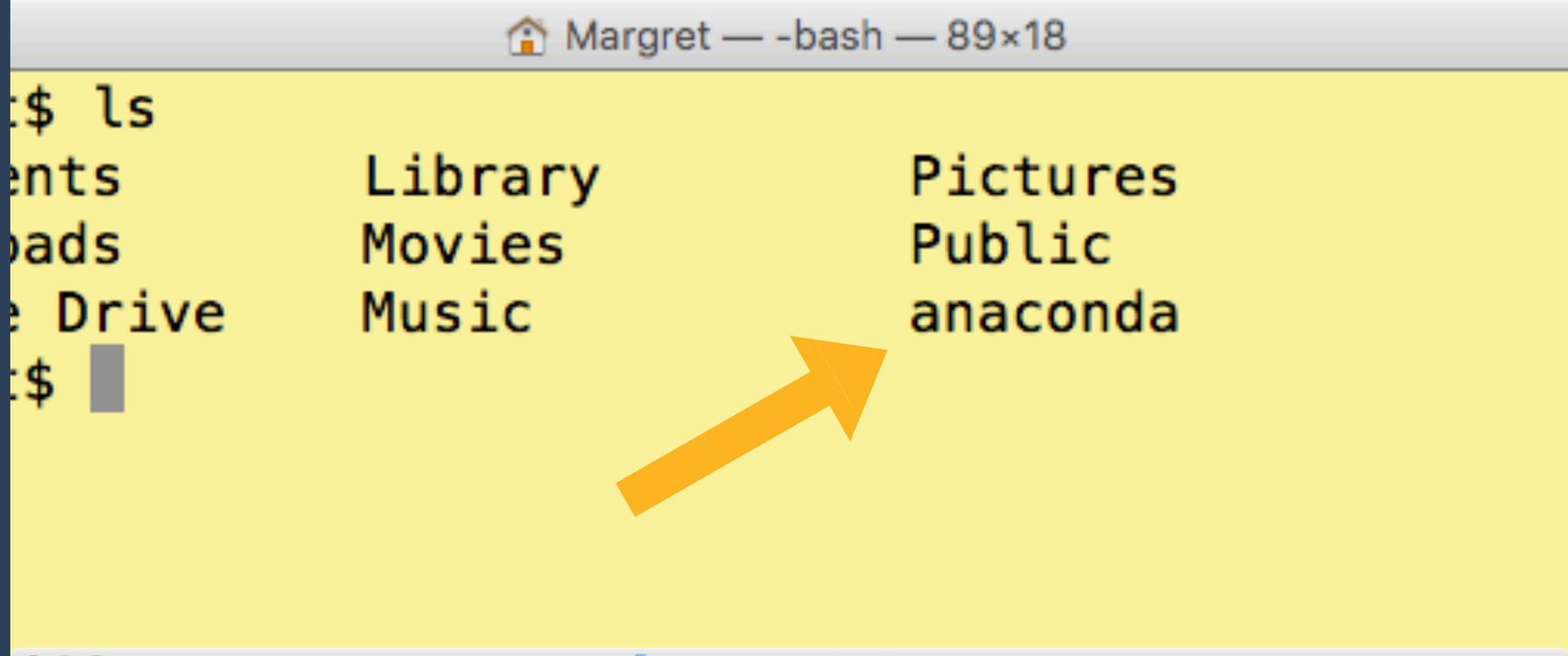
DEVELOPMENT ENVIRONMENT

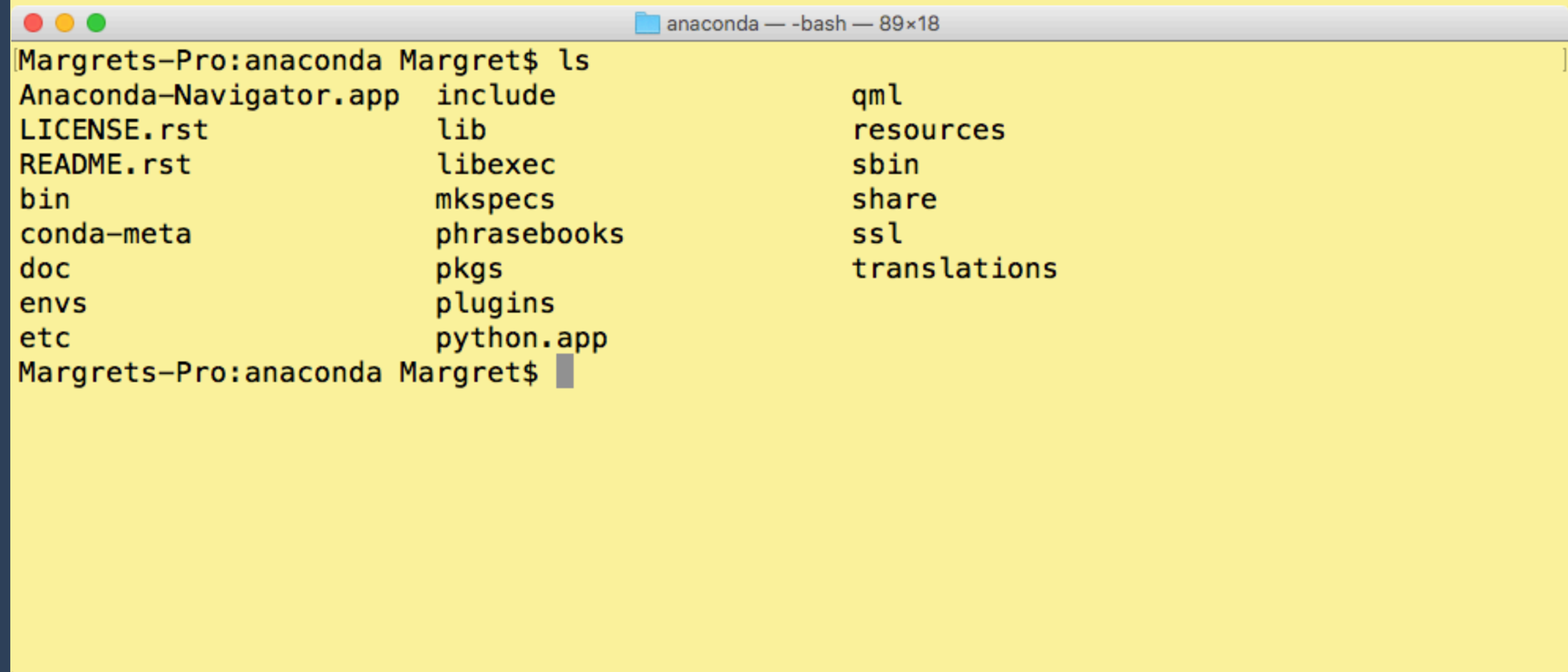


DEVELOPMENT ENVIRONMENT

- Installing will add "anaconda" folder to your home directory by default

```
% cd ~
% ls
```

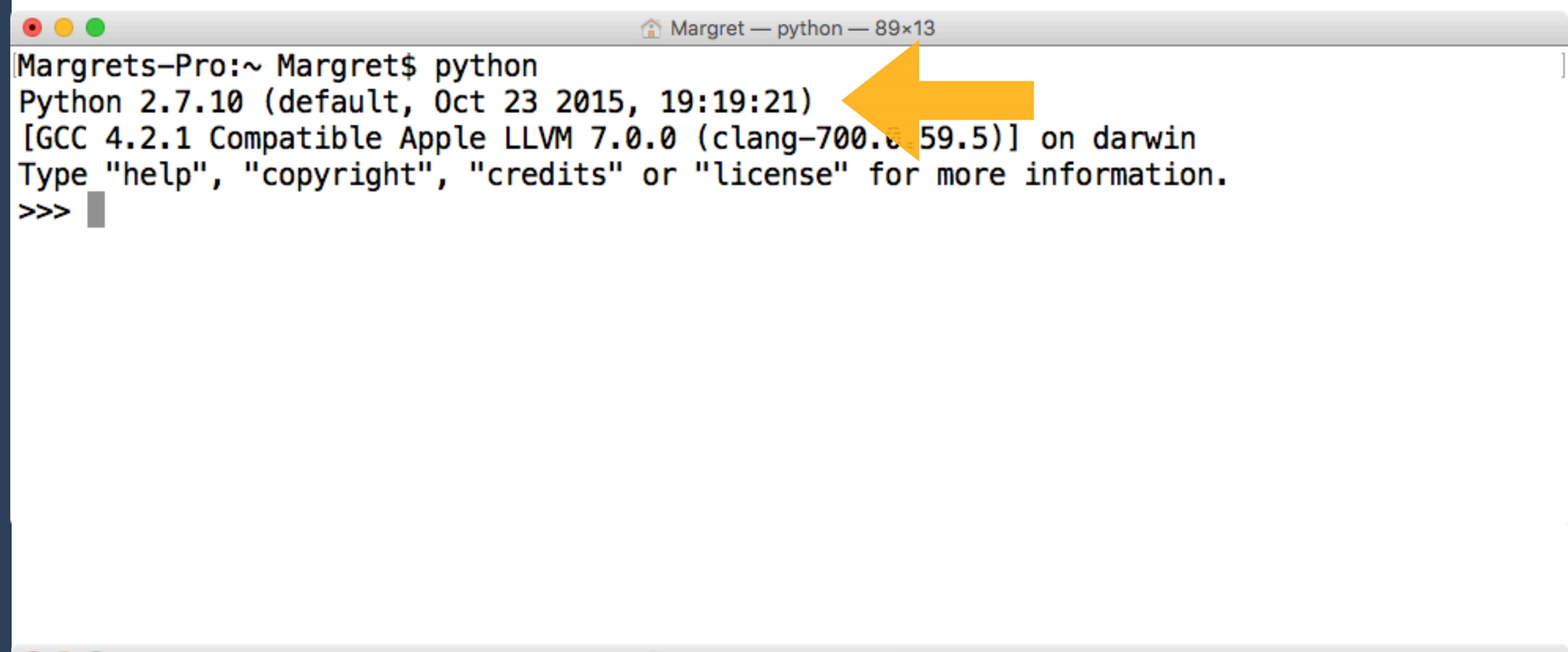


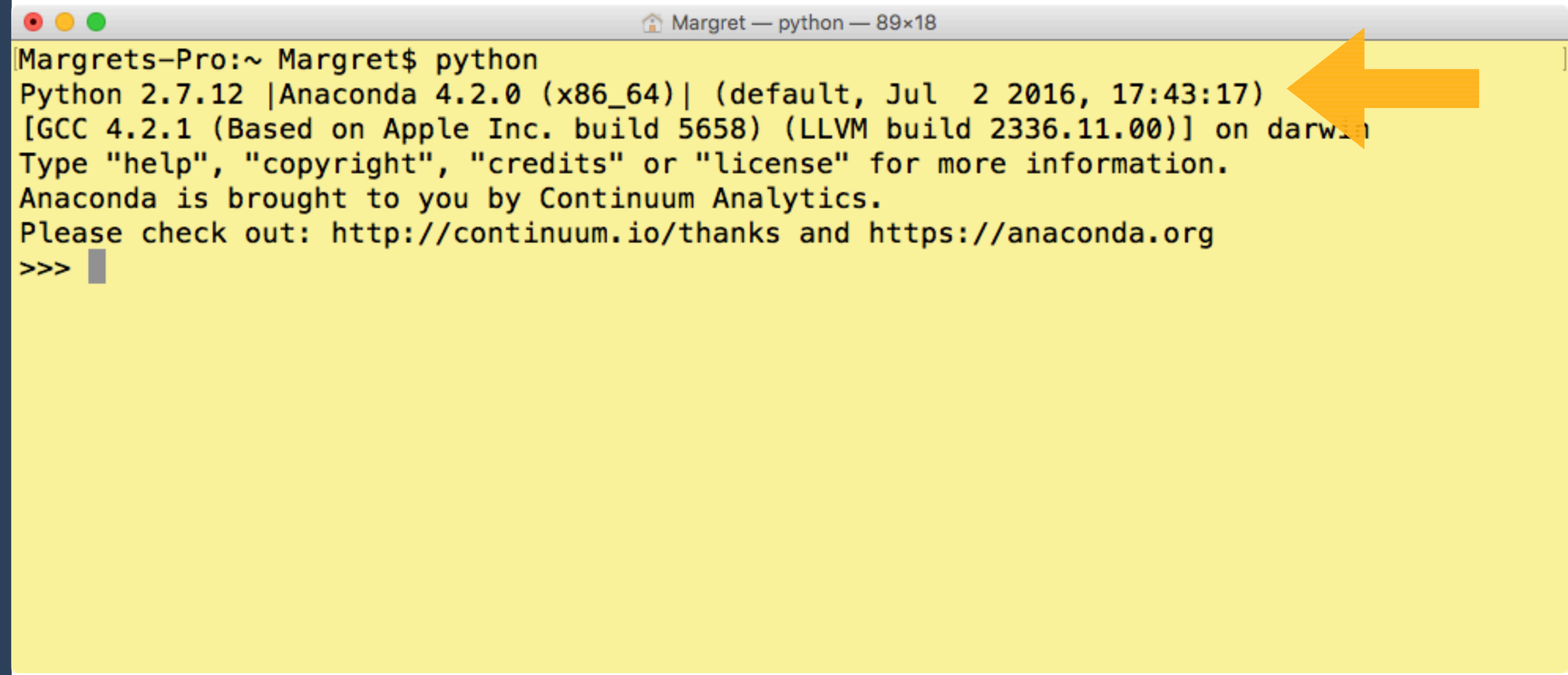


DEVELOPMENT ENVIRONMENT

- Anaconda changes your default Python
- Check the path of python

```
% which python
```





DEVELOPMENT ENVIRONMENT

Margret — -bash — 59×18

```
Margrets-Pro:~ Margret$ more .bash_profile

# added by Anaconda2 4.2.0 installer
export PATH="/Users/Margret/anaconda/bin:$PATH"
Margrets-Pro:~ Margret$
```

- How did that happen?

```
% cd ~
% more .bash_profile
```

DEVELOPMENT ENVIRONMENT

```
tbinkowski — -bash — 77×14
tbinkowski@TABinkosMBP2013 ~
502 % echo $PATH
/Users/tbinkowski/anaconda/bin:/Users/tbinkowski/google-clou
cal/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X1
ns/Server.app/Contents/ServerRoot/usr/bin:/Applications/Serv
erverRoot/usr/sbin:/Library/TeX/texbin:/usr/local/share/npm/
```

- How did that happen?
  - $PATH to anaconda was inserted before the default install

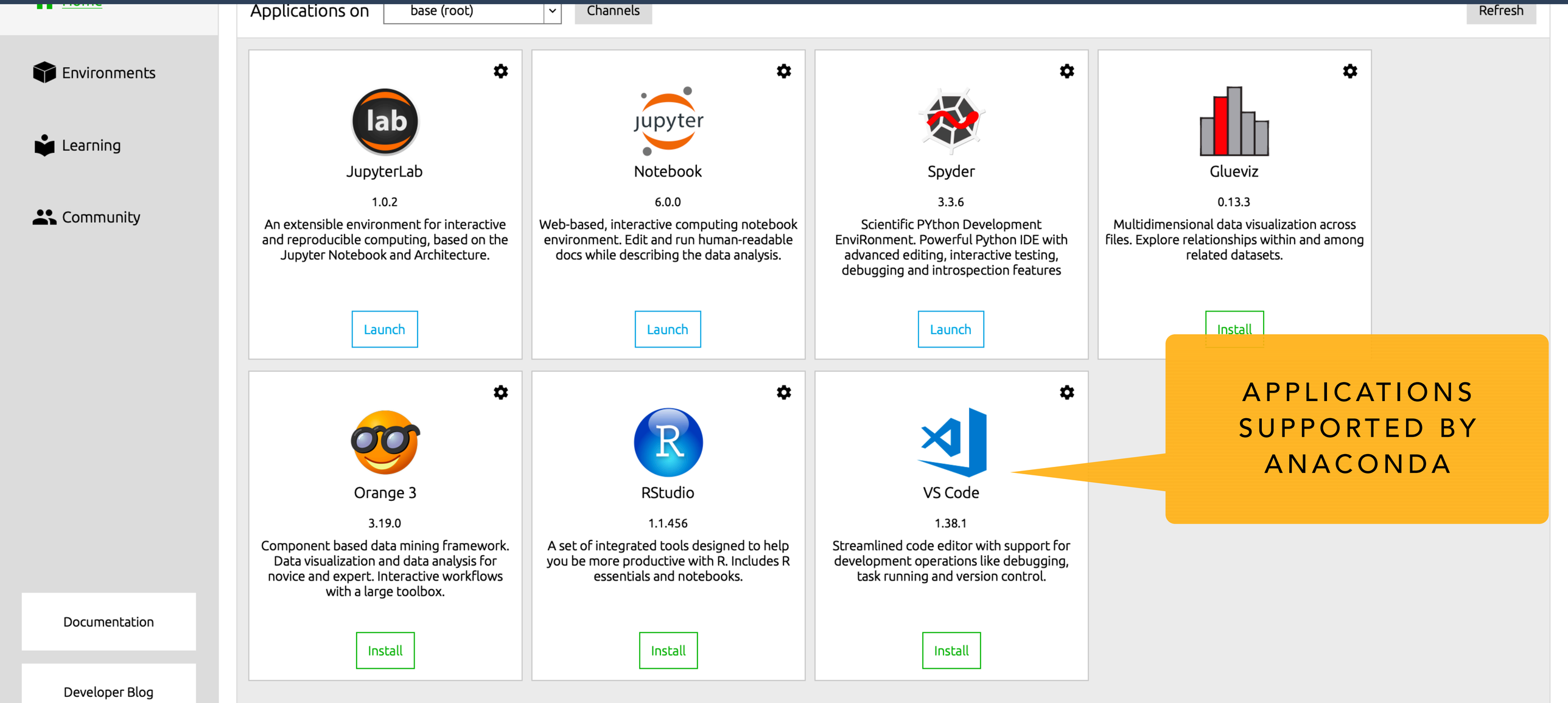
```
% echo $PATH
```

DEVELOPMENT ENVIRONMENT

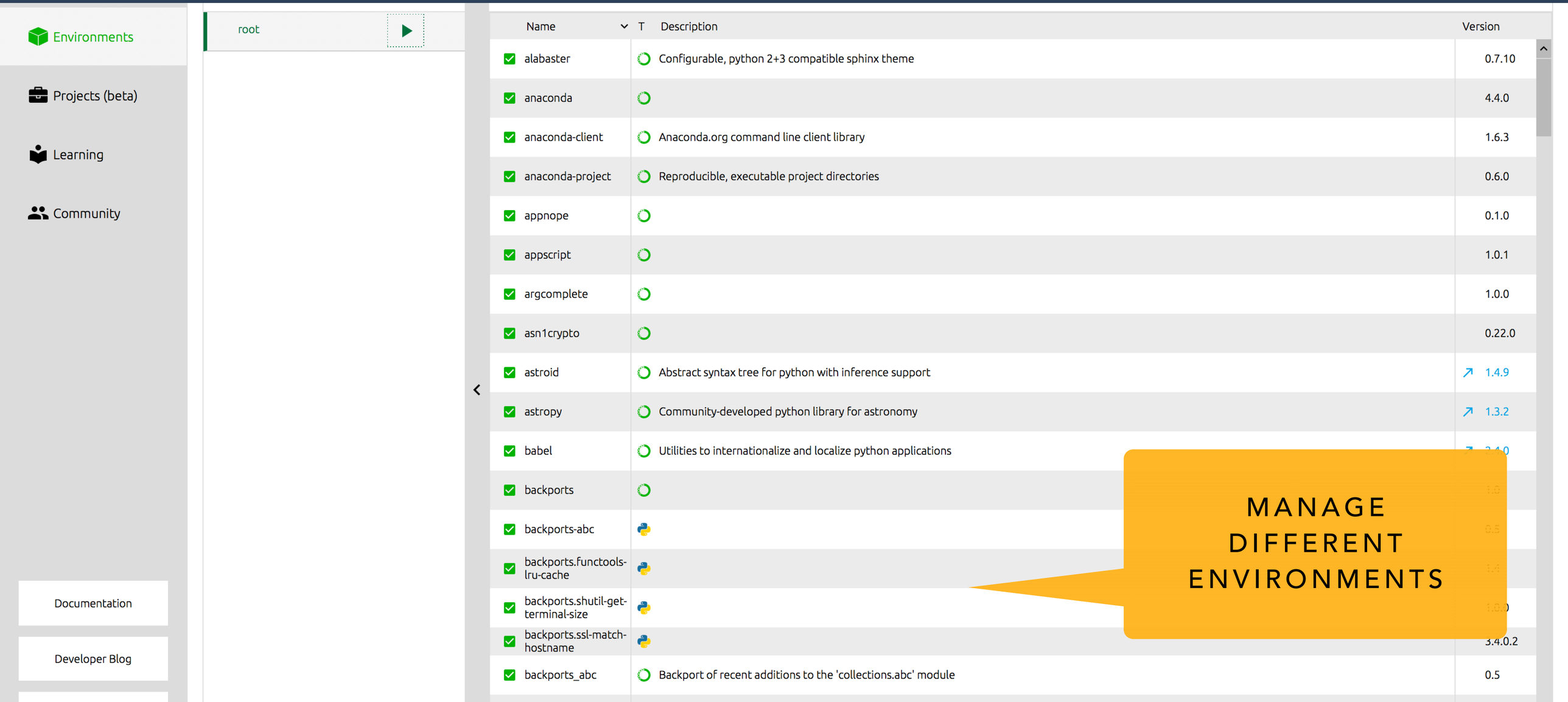
WE ARE NOT GOING TO BE USING VIRTUAL ENVIRONMENTS

- Virtual environments
  - Semi-isolated Python environment that allows packages to be installed for use by a particular application, rather than being installed system wide
- Examples
  - **pyenv** is the standard tool for creating virtual environments, and has been part of Python since Python 3.3
  - **virtualenv** is a third party alternative (and predecessor) to pyvenv. It allows virtual environments to be used on versions of Python prior to 3.4

DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT

- Environments
- Projects (beta)
- Learning
- Community

Python Tutorial	Python Reference	Anaconda Package List	Pandas Documentation	Numpy Documentation	Scipy Documentation	Matplotlib Documentation
Read	Read	Read	Read	Read	Read	Read
Bokeh User Guide	Anaconda Cloud Documentation	Anaconda Documentation	Anaconda Navigator Documentation	Continuum Training Program	Anaconda Skills Accelerator Program	
Read	Read	Read	Read	Explore	Explore	

- Documentation
- Developer Blog

LEARNING RESOURCES

DEVELOPMENT ENVIRONMENT

```
tabinkowski@Ts-MacBook-Pro ~
548 %  conda update --all
Fetching package metadata .........
Solving package specifications: .

Package plan for installation in environment /Applications/anaconda:

The following NEW packages will be INSTALLED:

    anaconda-project:          0.6.0-py27_0
    asn1crypto:                0.22.0-py27_0
    bkcharts:                  0.2-py27_0
    bleach:                    1.5.0-py27_0
    certifi:                   2016.2.28-py27_0
    functools_lru_cache:       1.4-py27_0
    html5lib:                  0.9999999-py27_0
    olefile:                   0.44-py27_0
```

EASILY UPDATE PYTHON AND PACKAGES

PACKAGES ARE CODE THAT OTHERS HAVE SHARED; SOME ARE DEPENDENCIES

DEBUGGER

DEBUGGER

- Python Debugger
- Lets you start/ stop a program while its running
- Keep track of values as they change through a program

27.3. pdb — The Python Debugger

Source code: Lib/pdb.py

The module `pdb` defines an interactive source code debugger for Python program
breakpoints and single stepping at the source line level, inspection of stack
evaluation of arbitrary Python code in the context of any stack frame. It also supp
can be called under program control.

The debugger is extensible – it is actually defined as the class `Pdb`. This is cu
understood by reading the source. The extension interface uses the modules `bdb` a

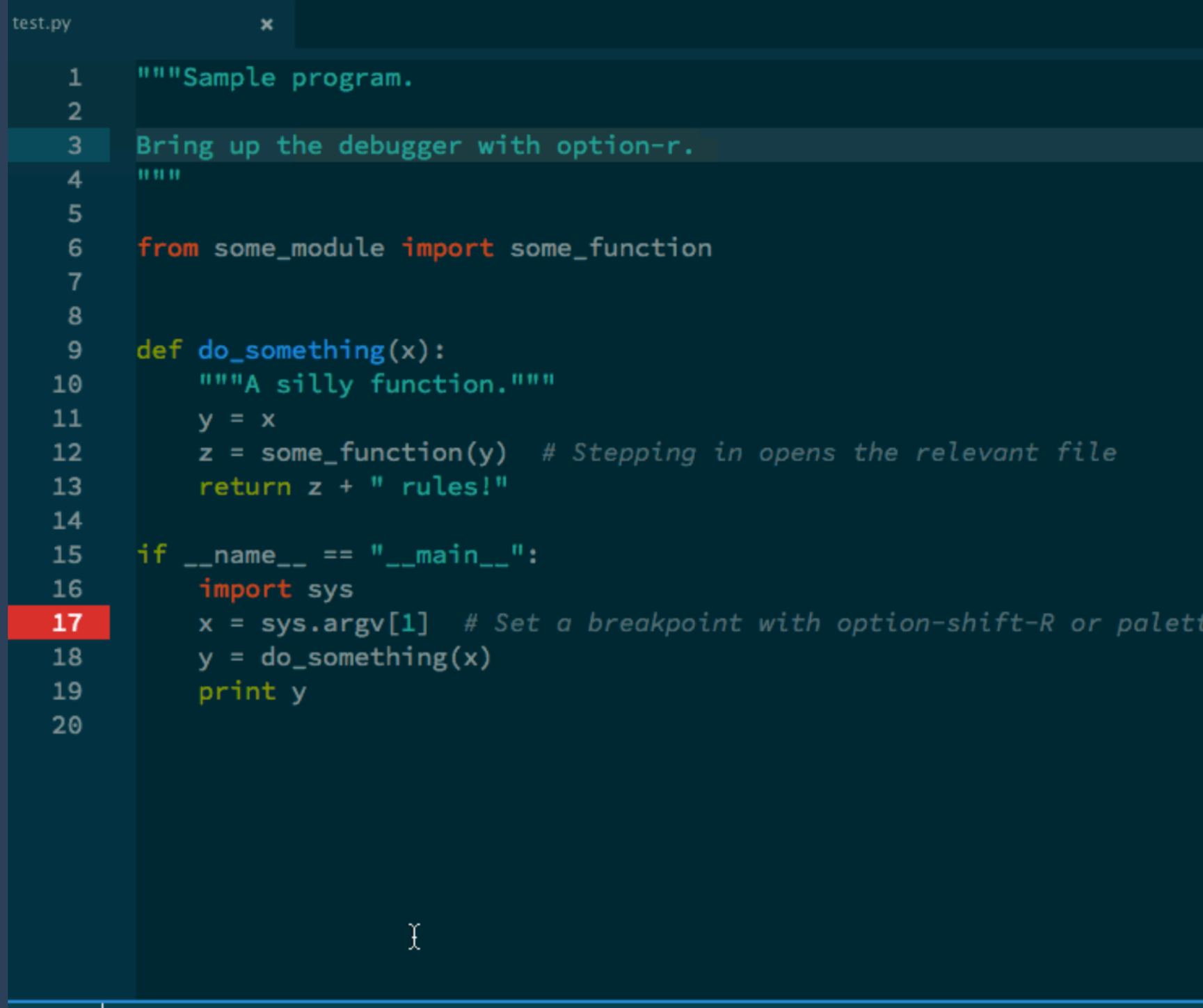
The debugger's prompt is `(Pdb)`. Typical usage to run a program under control of

```
>>> 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 availab

DEBUGGER

- Access pdb via text editor and/ or command line



DEBUGGER

- We won't worry about pdb for the first few weeks

```
x = 100
print(x)

y = 200
print(y)

z = x + y
print(x)
```

ALL THE DEBUGGING YOU'LL NEED

VERSION CONTROL

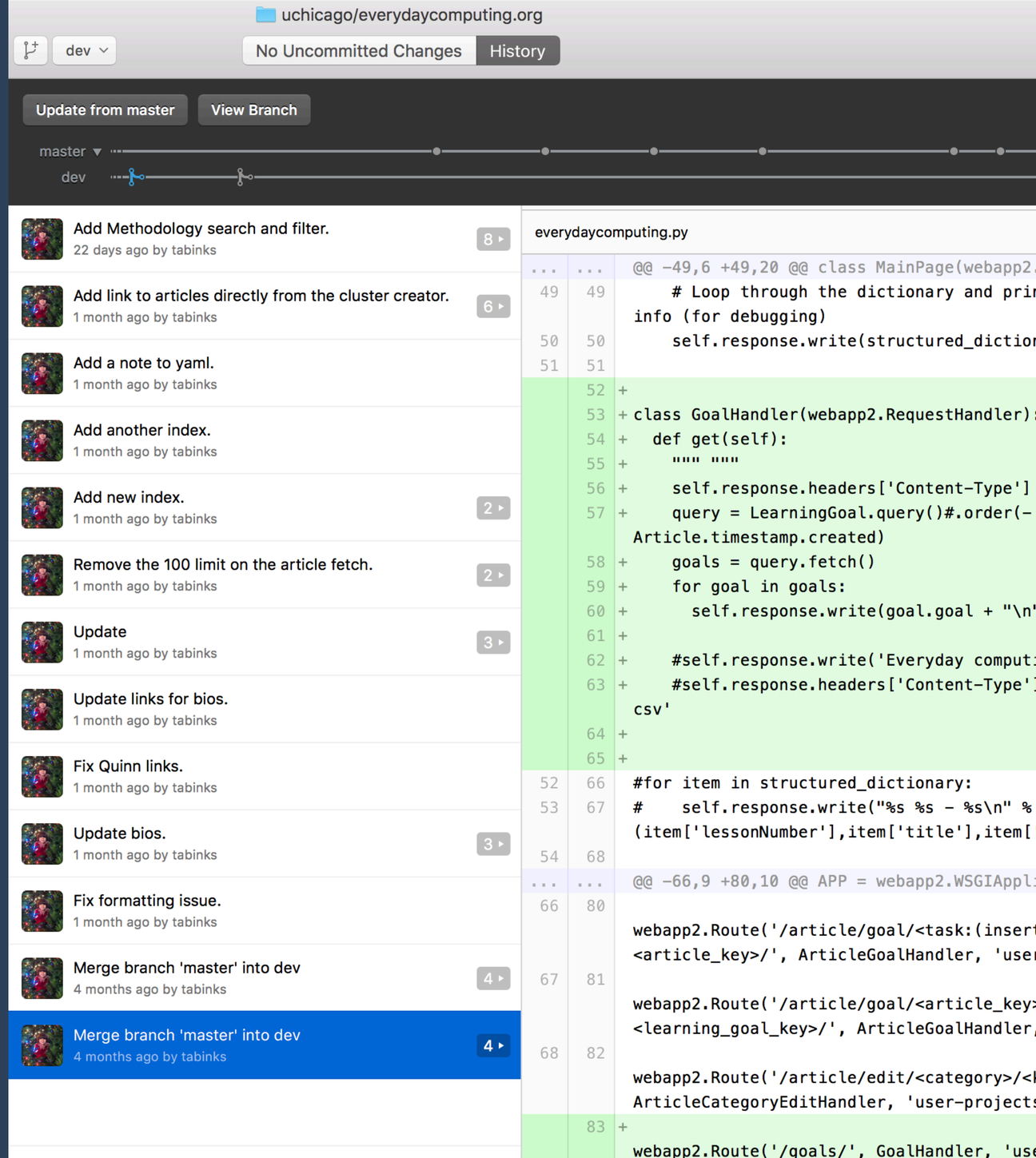
VERSION CONTROL

- Version control lets you save your work at different stages
- Compare changes
- Restore your work from previous saves

```
 6   6      return ItemView.extend ({
 7   7          attributes: {
 8   8              'class': 'review-activity'
 9   9          },
10  10          events: {
11  11              'click .review-activity--close-butto
12  12          },
13  13          ui: {
14  14              content: '.review-activity--content'
15  15          },
16  16          template: headerTemplate,
17              onRender: function () {
    17          initialize: function () {
18  18              this.collectionView = new ReviewActi
19                      el: this.ui.content.get(0),
20  19                  collection: this.collection
21                  }).render();
    20              });
22  21          },
    22          onRender: function () {
    23              this.collectionView.setElement(this.
    24              this.collectionView.render();
    25          },
23  26          triggerOnClose: function () {
```

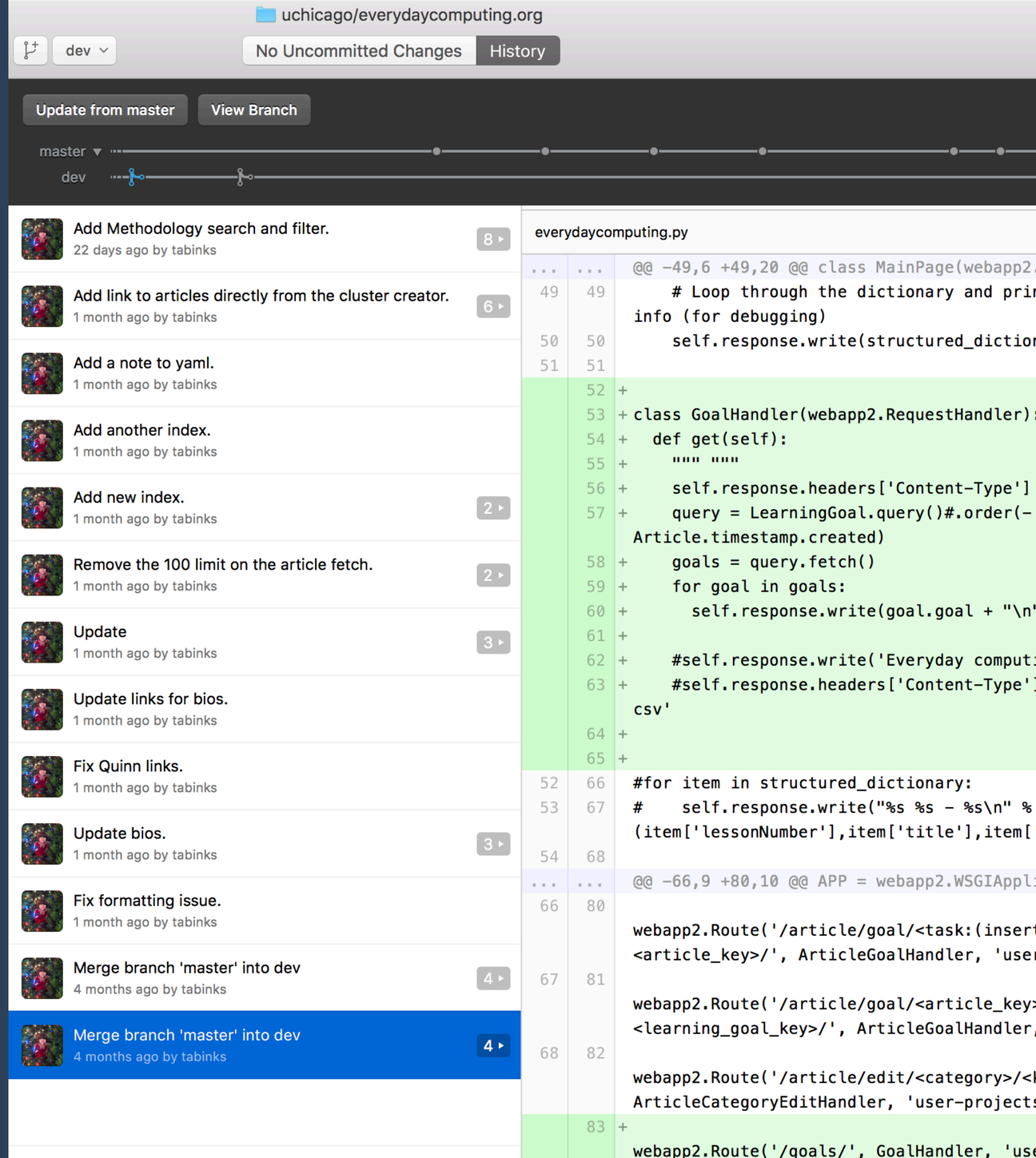
VERSION CONTROL

- Git is a version control system
- GitHub is service that provides git server accessible online
  - Desktop app provides a gentle entry point to using git



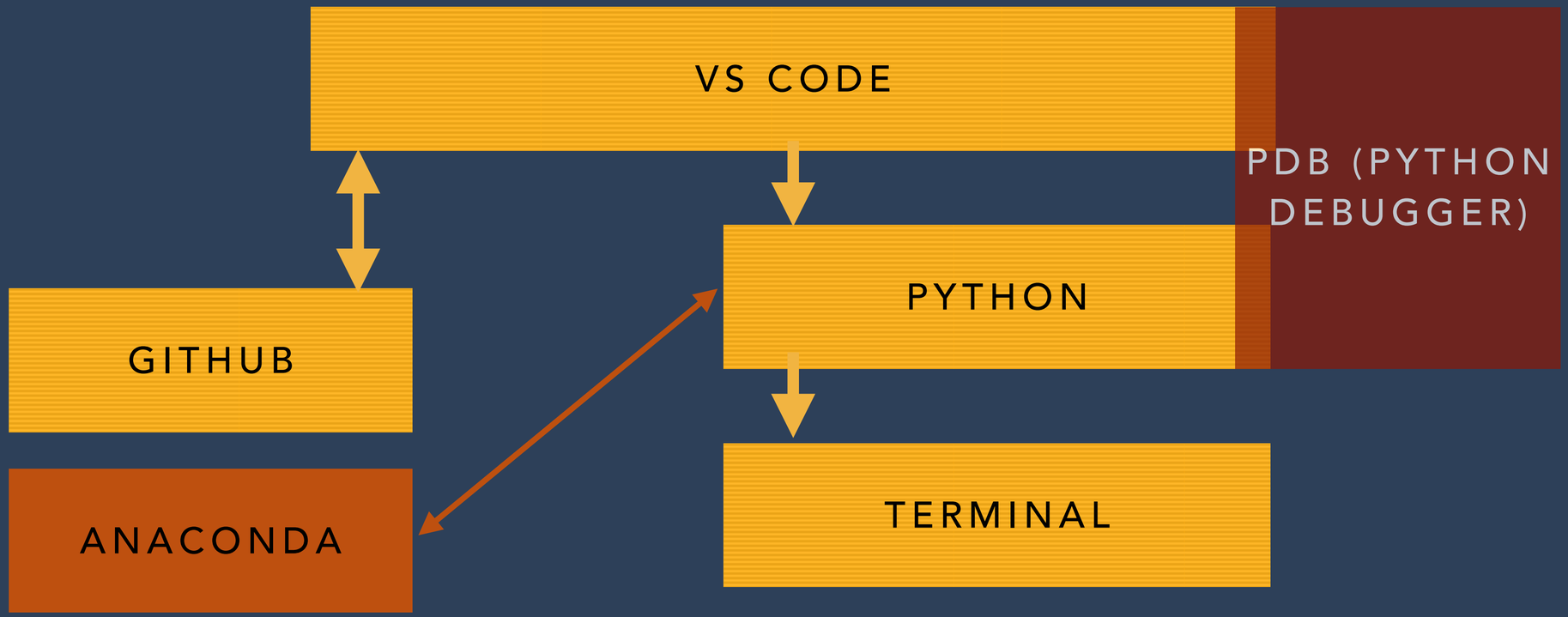
VERSION CONTROL

- We will be using GitHub to distribute and collect assignments
  - Not for assignment 1



PUTTING IT ALL TOGETHER

DEVELOPMENT ENVIRONMENT



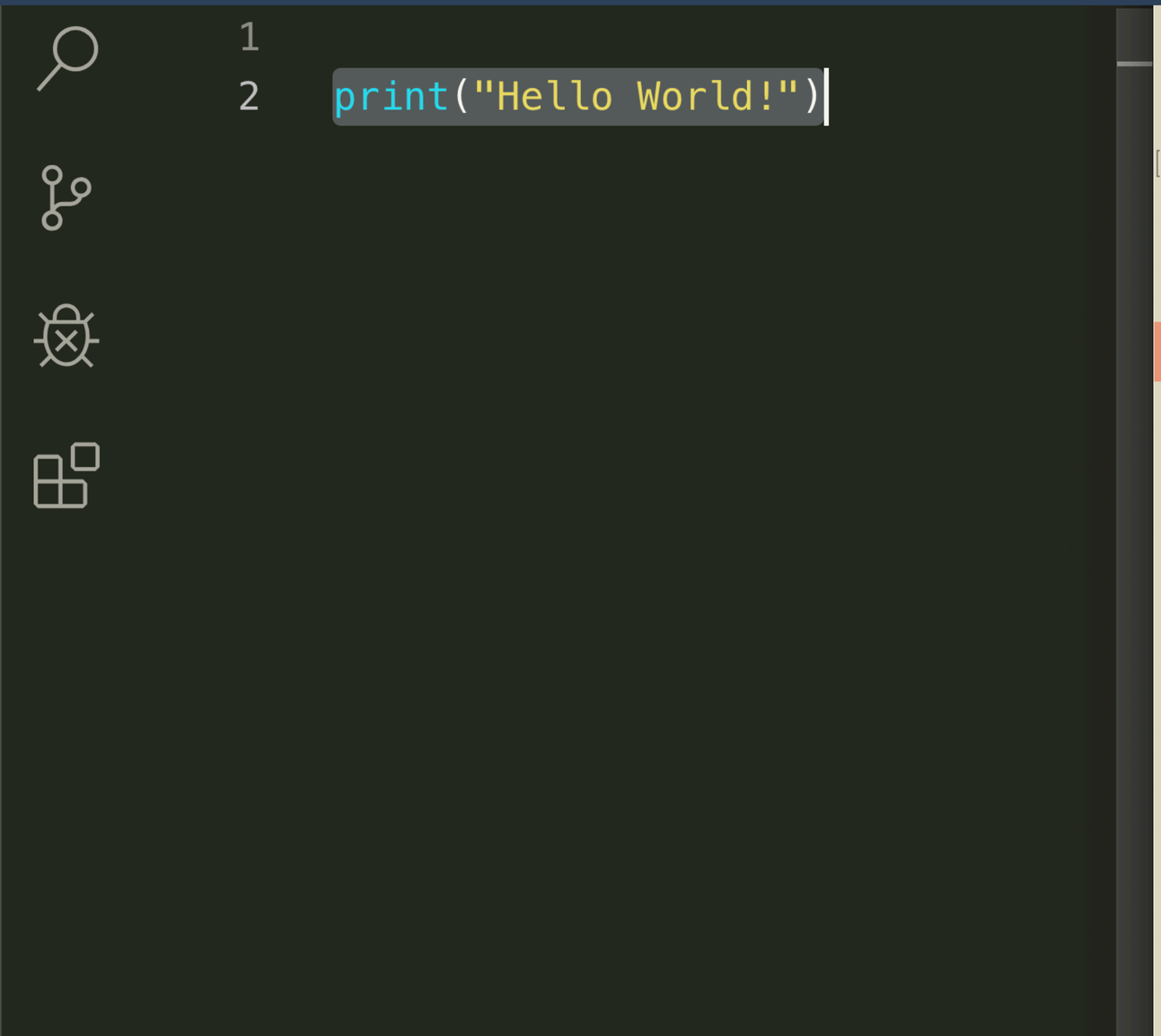
DEVELOPMENT ENVIRONMENT

- Create a folder to work in
- Create a python file in shell
- Edit it in viscose
- Run it with python

```
% cd ~/Documents/

% mkdir mpcs50101/

% touch hello_world.py

% code hello_world.py

% python hello_world.py
```

DEVELOPMENT ENVIRONMENT

SCRIPT MODE



THE END

MPCS 50101
MODULE 1

THE UNIVERSITY OF CHICAGO