

# CONCEPTS OF PROGRAMMING

©TTAA .BB INKKOWSKJ ,22019

MPCS 50101  
MODULE 1



THE UNIVERSITY OF  
CHICAGO

# WELCOME TO PROGRAMMING

MODULE 1  
MPCS 50101

© T.A. BINKOWSKI, 2020



THE UNIVERSITY OF  
CHICAGO

# PROGRAMMING

- Programming is how a person writes instructions to tell a computer what to do
- No matter how fancy AI/machine learning are using... you still need to tell a computer exactly what to do



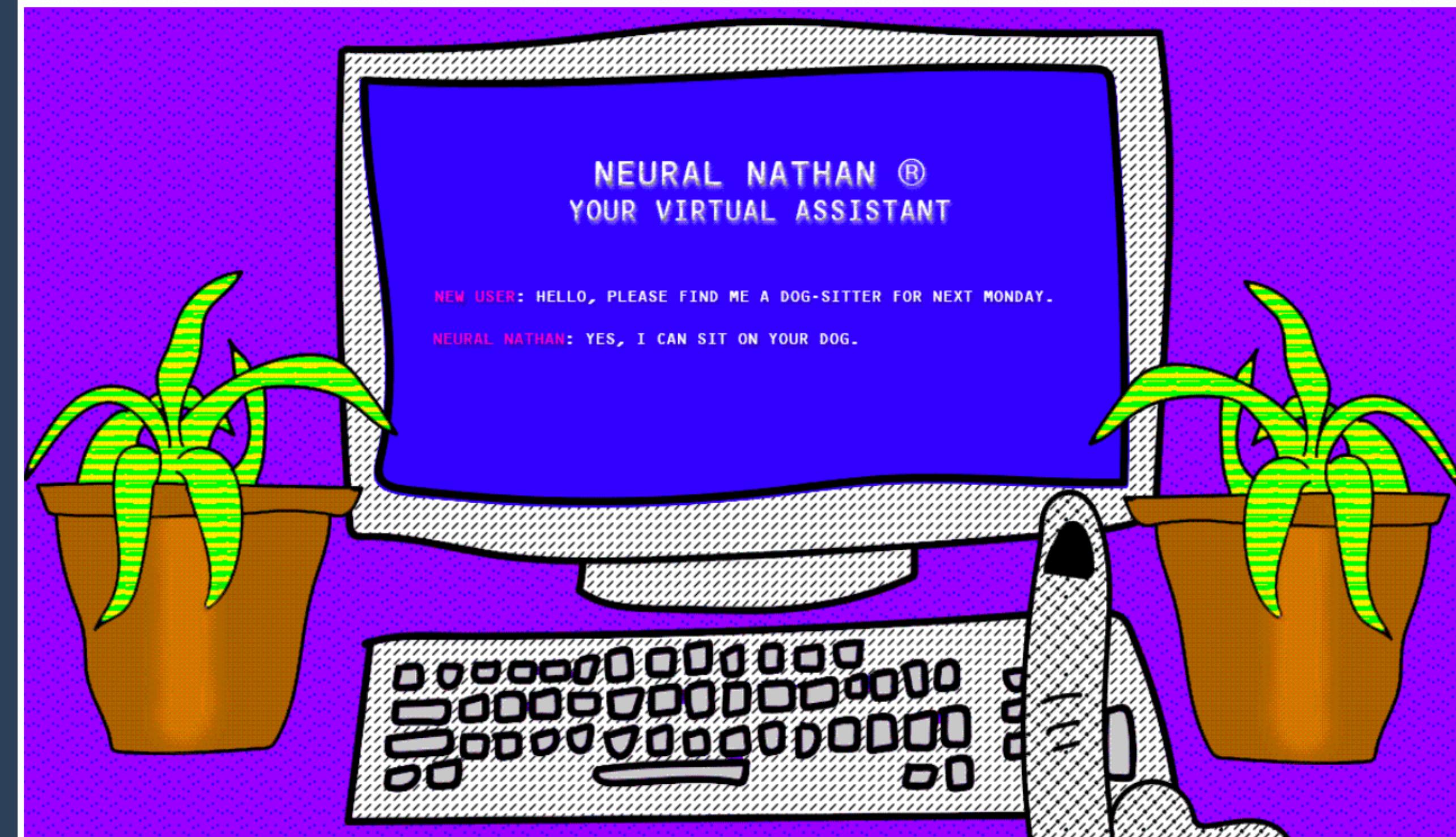
# PROGRAMMING

- Sometimes the best programs are actually humans

## The Humans Hiding Behind the Chatbots

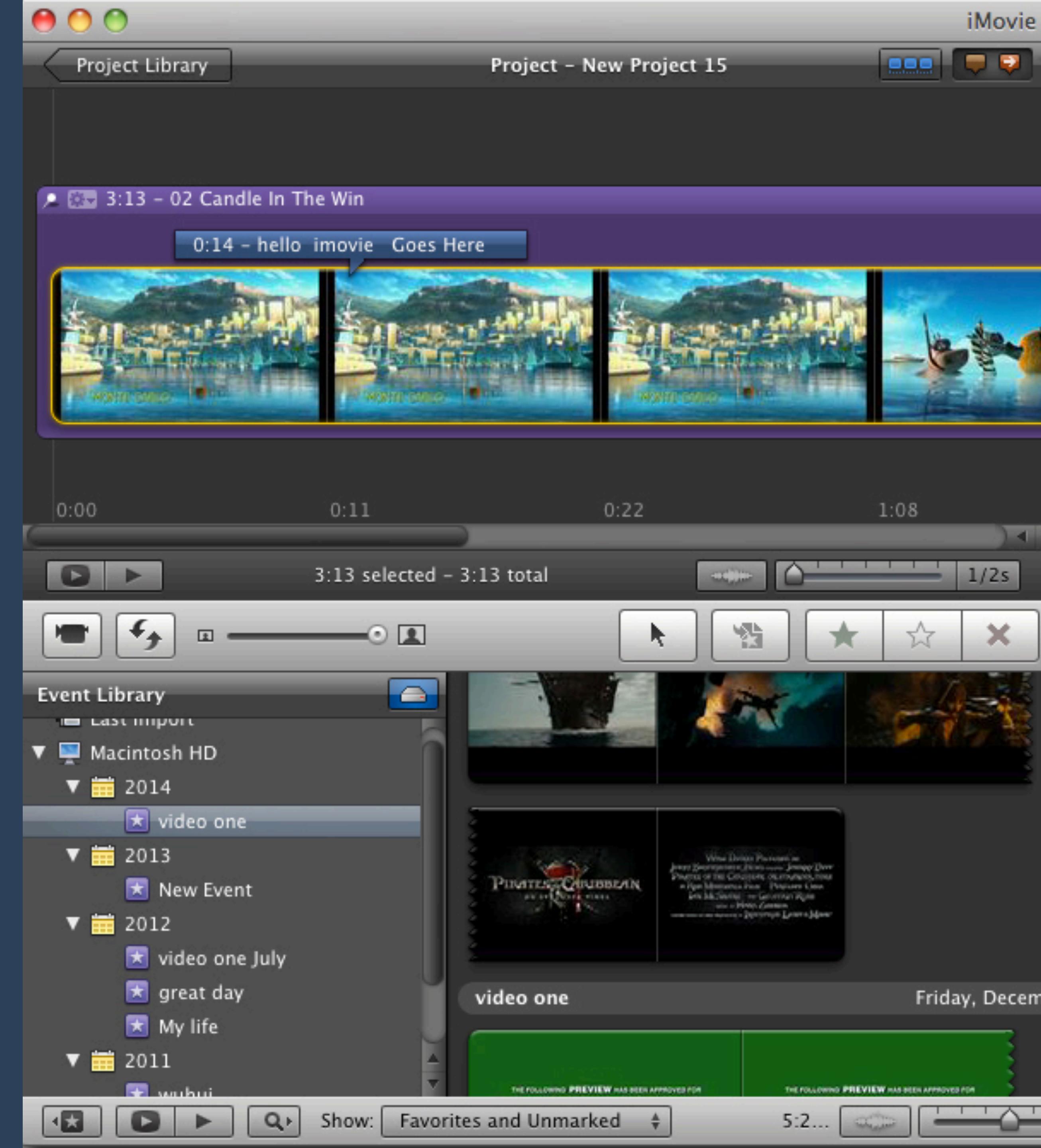
Behind the artificial intelligence personal assistants and concierges are actual people, reading e-mails and ordering Chipotle.

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



"learn python the hard way" filetype:pdf

All Books Videos Shopping Images More

About 5,870 results (0.34 seconds)

[PDF] [Learn Python the Hard Way - Sourav Sen Gupta](#)

[www.souravsengupta.com/int2pro2014/python/LPTHW.pdf](http://www.souravsengupta.com/int2pro2014/python/LPTHW.pdf) ▾

Learn Python the hard way : a very simple introduction to the terrifyingly beautiful computers and code / Zed A. Shaw.—Third edition. pages cm. Includes ...

[pyway/Learn Python The Hard Way, 3rd Edition .pdf at mast](#)

<https://github.com/.../Learn%20Python%20The%20Hard%20Way,%203...> ▾

Apr 24, 2014 - Learn Python The Hard Way. Contribute to pyway development by account on GitHub.

[python/download-learn-python-the-hard-way-2nd-edition.pdf](#)

<https://github.com/.../download-learn-python-the-hard-way-2nd-edition....> ▾

python scripts. Contribute to python development by creating an account on GitHub

[PDF] [Learn Python The Hard Way - 7Chan](#)

<https://7chan.org/pr/src/LearnPythonTheHardWay2ndEdition.pdf> ▾

by ZA Shaw - 2011 - Cited by 10 - Related articles

Jun 24, 2011 - Welcome to the 2nd Edition of Learn Python the hard way. You can

companion site to the book at <http://learnpythonthehardway.org/> ...

[PDF] [Learn Python the Hard Way: A Very Simple Introduction](#)

[ptgmedia.pearsoncmg.com/images/.../samplepages/9780321884916.pdf](http://ptgmedia.pearsoncmg.com/images/.../samplepages/9780321884916.pdf) ▾

Learn Python the hard way : a very simple introduction to the terrifyingly beautiful computers and code / Zed A. Shaw.—Third edition. pages cm. Includes ...

[PDF] [Learn Python The Hard Way - user web page](#)

[users.aims.ac.za/~lafras/books/pythonthehardway.pdf](http://users.aims.ac.za/~lafras/books/pythonthehardway.pdf) ▾

by ZA Shaw - 2010 - Cited by 10 - Related articles

# 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 computer's as the second.
    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 computer's
    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.

"Hello world" in different languages

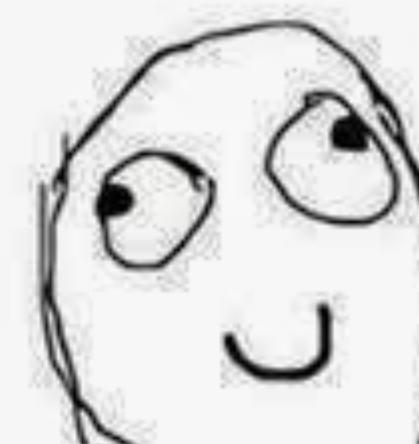
Python:

```
print "Hello world!"
```



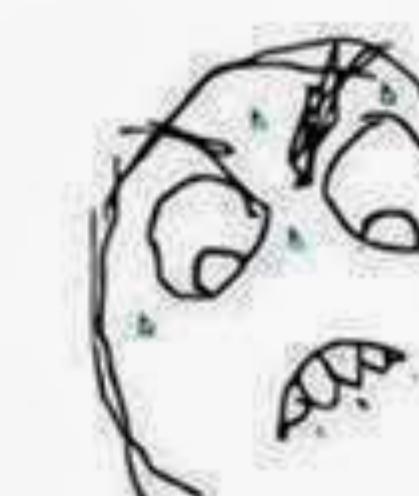
PHP:

```
<?php  
echo 'Hello world!';  
?>
```



C:

```
#include <stdio.h>  
main()  
{  
    printf ("Hello World!\n");  
}
```



Assembly:

```
.model small  
.stack 100h  
.data  
bonjour db "Hello world!\0"  
.code  
main proc  
    mov AX, 0data  
    mov DS, AX  
    mov DX, offset bonjour  
    mov BX, 0900h  
    int 21h  
    mov BX, 4C00h  
    int 21h  
    main endp  
end main
```



# 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

```
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
```

```
    print(' | |')
```

```
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
```

```
    print(' | |')
```

```
    print('-----')
```

```
    print(' | |')
```

```
+ board[6])
```

```
+ board[3])
```

MOST APPLICATIONS  
ARE WRITTEN IN A  
HIGH LEVEL  
LANGUAGE

```
want to be.
```

```
the first item, and t
```

```
's letter, the second
```

```
.def whoGoesFirst():
```

```
    # Randomly choose the player who goes first.
```

```
    if random.randint(0, 1) == 0:
```

```
        return 'computer'
```

```
    else:
```

```
        return 'player'
```

```
.def playAgain():
```

# PROGRAMMING LANGUAGES

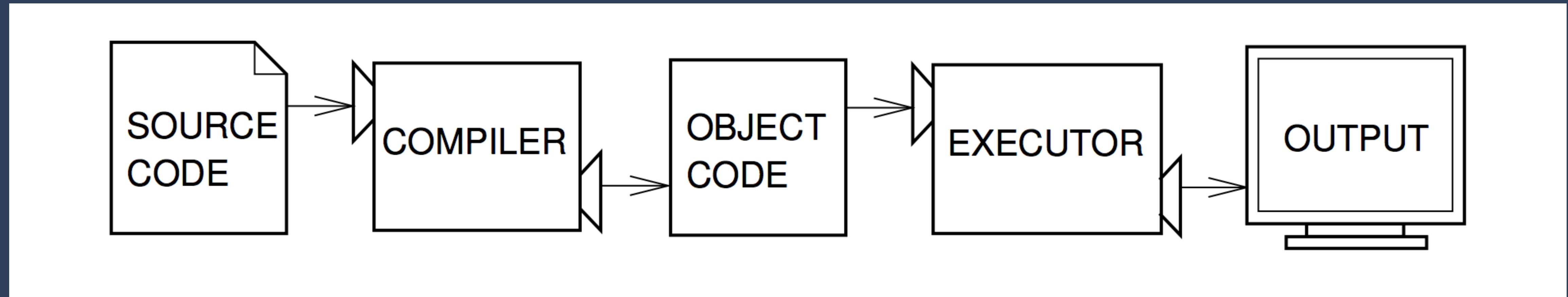
- High-level language vs. low-level language

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.

# PROGRAMMING LANGUAGES

- 
- High-level
- 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
- Low-level

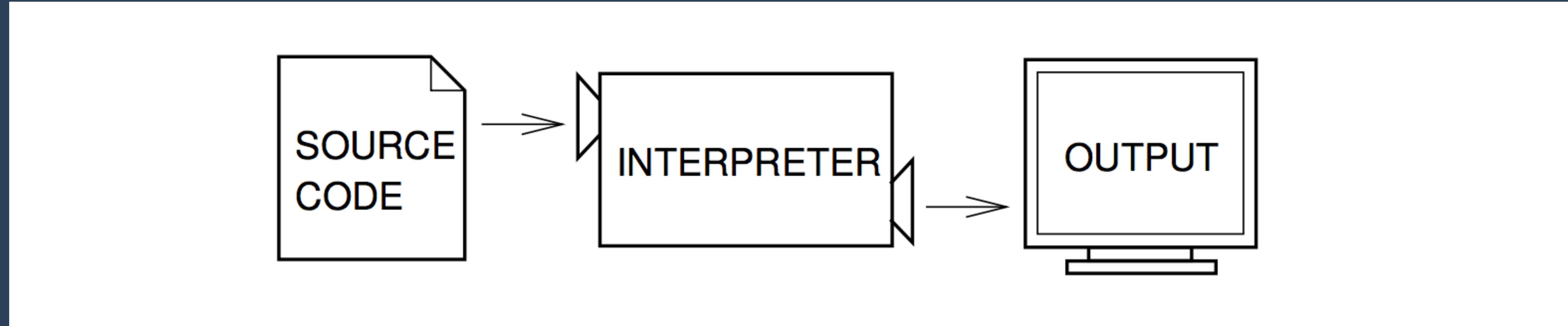
# PROGRAMMING LANGUAGES



- 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

The screenshot shows the Python Software Foundation website. At the top, there's a navigation bar with tabs for Python, PSF, Docs, PyPI, and Jobs. The Python tab is active. Below the navigation is the Python logo and a search bar. A main content area features a code snippet demonstrating Python lists and comprehensions, followed by a section titled "Compound Data Types" about lists. At the bottom, there are links for Get Started, Download, Docs, and Jobs.

python™

About Downloads Documentation Community Success Stories News

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

**Compound Data Types**

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists are indexed, sliced and manipulated with other built-in functions. [More about lists in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)

---

**Get Started**  
Whether you're new to programming or an experienced developer, it's easy to learn and use Python.  
[Start with our Beginner's Guide](#)

---

**Download**  
Python source code and installers are available for download for all versions! Not sure which version to use?  
[Check here.](#)  
Latest: Python 3.6.0 - Python 2.7.13

---

**Docs**  
Documentation for Python's standard library, along with tutorials and guides, are available online.  
[docs.python.org](#)

---

**Jobs**  
Looking for work? Python related jobs are available.  
You're trying to find a job? Our [job board](#) is the place to look.  
[jobs.python.org](#)

---

**Latest News**  
[» More](#)

---

**Upcoming Events**

# COURSE TECHNOLOGIES

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

The screenshot shows the official website for SciPy.org. The header includes links for "DUDGE REPORT 2014®", "Hacker News", "Google News", "Screen Time", "Analytics", "Journals", "GTasks", and "GrabL". Below the header, there are tabs for "Untitled 1", "Introduction t...", "uchicago-link...", "PLOS Collect...", and "Online Resou...". The main content area features the SciPy logo and a banner for "Sponsored By ENTHOUGHT". Below the banner are four icons: "Getting Started" (yellow circle with a white S), "Documentation" (blue circle with a white S and books), "Report Bugs" (blue circle with a white S and a red bug), and "Blogs" (orange square with a white RSS feed icon). To the right of these icons is a sidebar with a vertical menu:

- About
- Insta
- Getti
- Docu
- Bug R
- Topic
- Cook
- SciPy
- Wiki
- SciPy
- Blog
- Num

Below the sidebar, there is a section titled "CORE" with links to "NumPy", "SciPy", "Matplotlib", "IPython", "SymPy", and "Pandas". A "More information..." button is located at the bottom of this section. At the very bottom of the page, there is a footer with several items listed under "based":

- 09-07) See [Obtaining NumPy & SciPy libraries.](#)
- 08-09) See [Obtaining NumPy & SciPy libraries.](#)
- 05-03) See [Obtaining NumPy & SciPy libraries.](#)
- 03-26) See [Obtaining NumPy & SciPy libraries.](#)

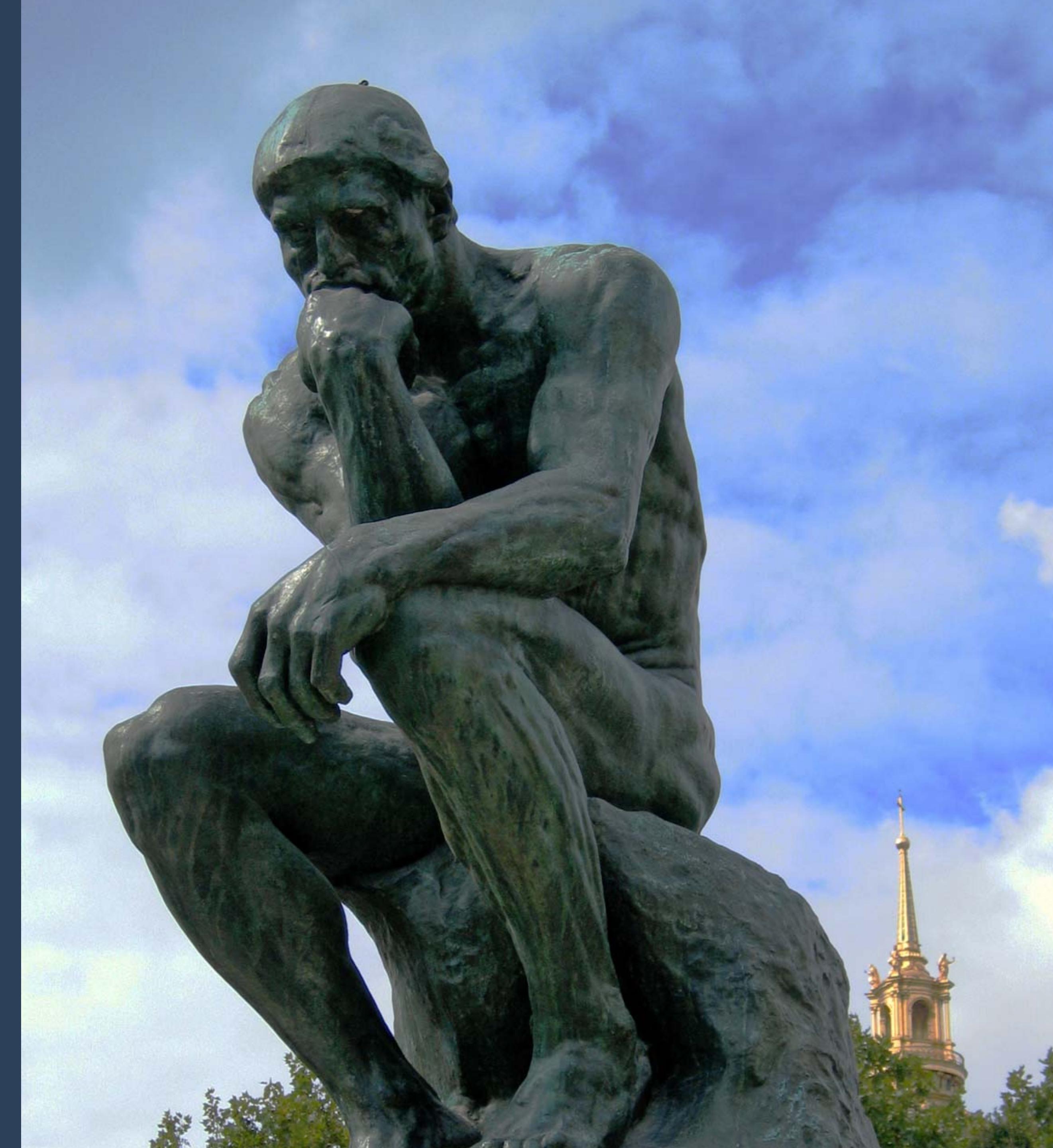
# PROGRAMMING LANGUAGES

- 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?

<b>C</b>	<b>D</b>	<b>Delphi</b>	<b>Hello</b>
#include <stdio.h> int main(void) { printf("Hello, world!\n"); return 0; }	module helloworld; import std.stdio; void main() { writeln("Hello, world!"); }	Program Hello_World; {\$APPTYPE CONSOLE} Begin WriteLn('Hello, world!'); End.	h
<b>Boo</b>		<b>Lisp</b>	
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)		"Hello, world!"	
<b>Java</b>	<b>C++</b>	<b>Groovy</b>	
class Hallo { public static void main( String[] args ) { System.out.println("Hello, world!"); } }	#include <iostream> using namespace std; int main() { cout << "Hello, world!" << endl; return 0; }	println "Hello, world!"	
<b>JavaScript</b>	<b>Objective-C</b>	<b>Perl</b>	
document.writeln('Hello, world!');	#import <Foundation/Foundation.h> int main() { NSLog(@"Hello, world!"); return 0; }	print "Hello, world!\n";	
<b>AutoIt</b>		<b>Ruby</b>	
MsgBox(0,"","Hello, world!")		puts 'Hello, world!'	
<b>BASIC</b>	<b>This=That</b>	<b>Lua</b>	
10 PRINT "Hello, world!"	x=Hello, world! x=print	print("Hello, world!")	
<b>Blitz Basic</b>	<b>Tcl</b>	<b>Ruby</b>	
Graphics 800,600 SetBuffer BackBuffer(); Text 10, 10, "Hello, world!" Flip WaitKey() End	puts "Hello, world!"	puts 'Hello, world!'	
<b>Brainfuck</b>	<b>Visual Basic</b>	<b>Whitespace</b>	
[−]++++++ ++[>++++++ >++++++ >++++>++++>+<<<<->++. .+++.>++++. .+++.<+++++++.---- -.+++.-----.---- .++>.	Imports System Module Module1 Sub Main() Console.WriteLine("Hello, world!") End Sub End Module		
<b>Haskell</b>	<b>C#</b>	<b>SQL</b>	
main = putStrLn "Hello, world!"	class MainClass { public static void Main() { System.Console.WriteLine("Hello, world!"); } }	SELECT "Hello, world!" AS message ;	
<b>B</b>	<b>HTML</b>	<b>Smalltalk</b>	
main() { printf("Hello, world!"); }	<!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>	Transcript show: 'Hello, world!'	
<b>COBOL</b>	<b>History</b>	<b>PHP</b>	
000100 IDENTIFICATION DIVISION. 000200 PROGRAM-ID. HELLOWORLD. 000900 PROCEDURE DIVISION. 001000 MAIN.	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	<? echo "Hello, world!" ?>	
<b>MACHINE CODE</b>		<b>troff/groff</b>	
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		\f(CW Hello, world!	
<b>Fortran</b>		<b>MACHINE CODE</b>	
program hello write(*,*) "Hello, world!" end program hello		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	
<b>LOLCODE</b>		<b>LOLCODE</b>	
		LOLCODE	

# PROGRAMMING LANGUAGES

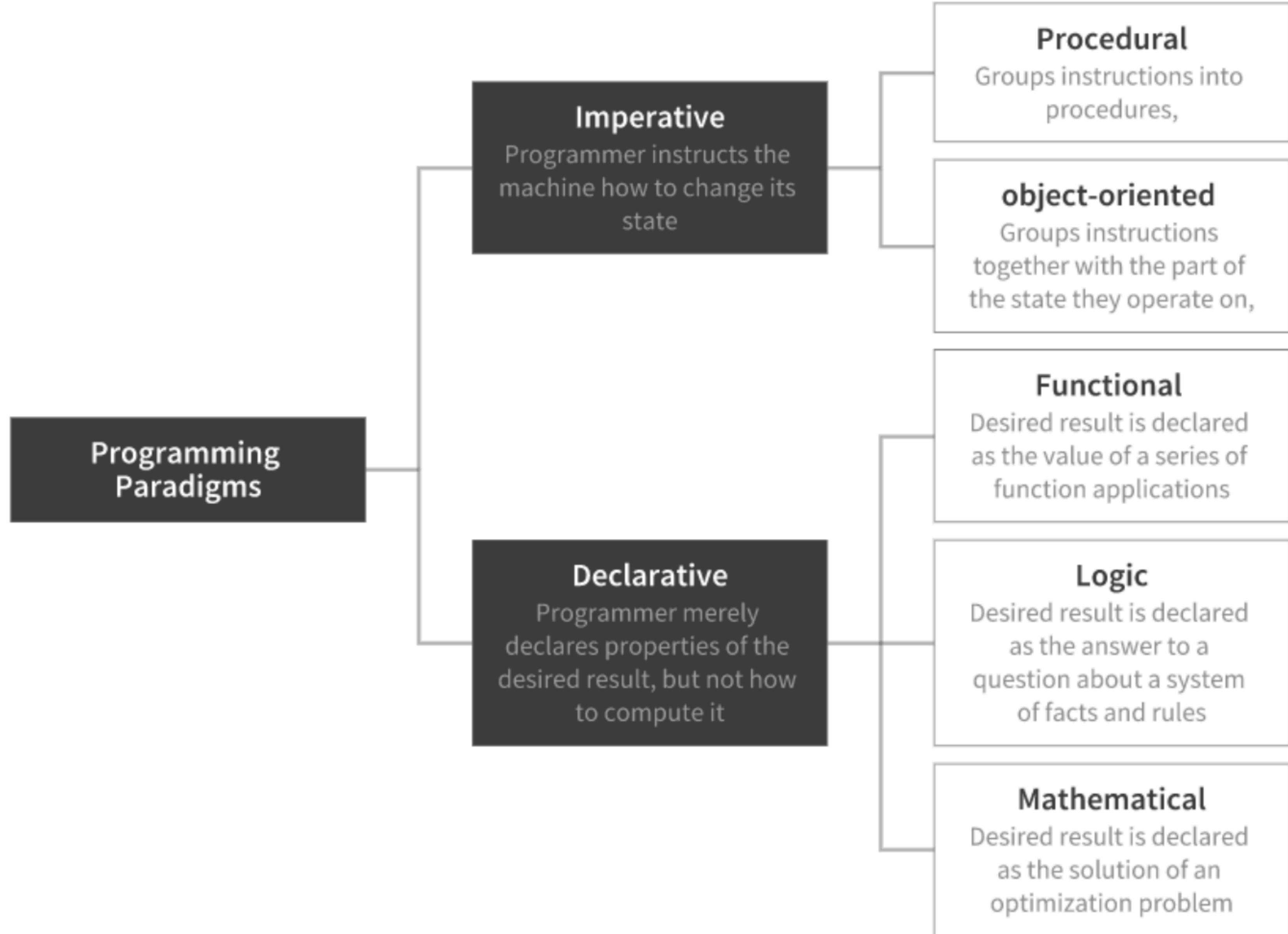
- You will spend more time thinking about how to solve the problem
  - Are you fighting the problem or the language?



# 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's just the style in which you program



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

```
f(3)
17
```

PROCEDURAL



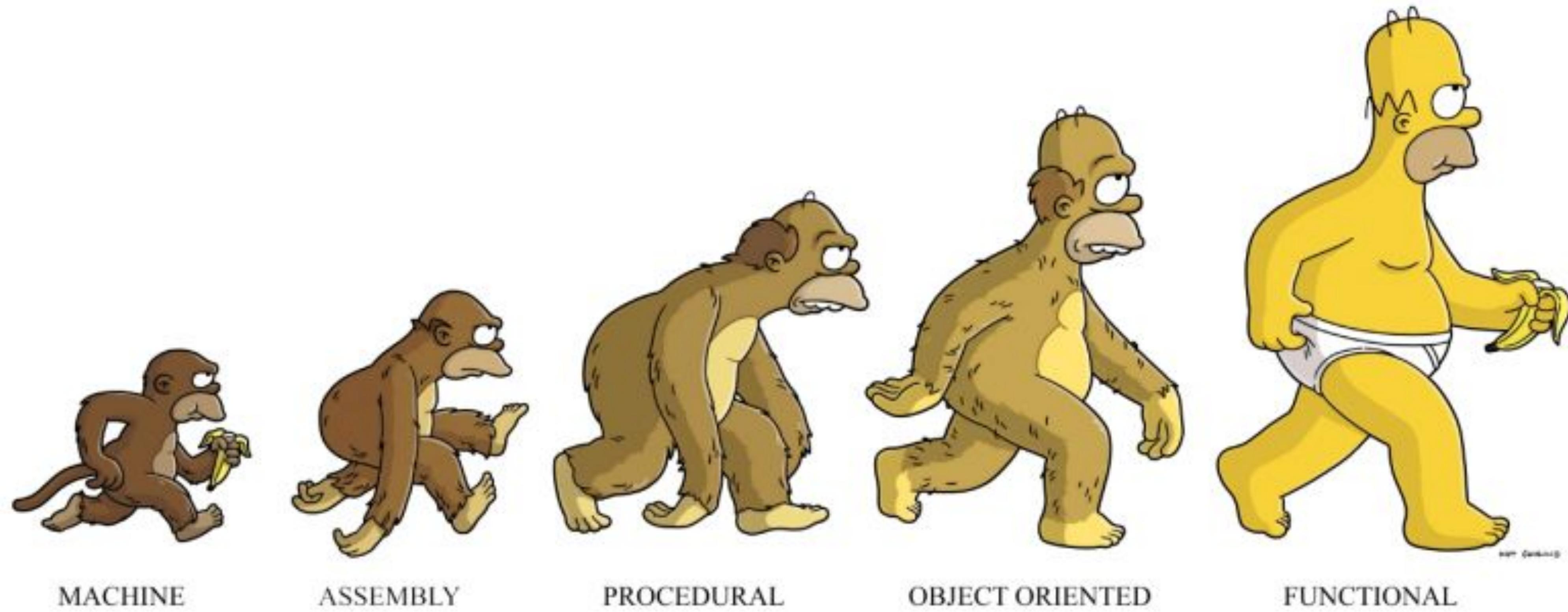
```
# Function to compute 5x+2 using Lambda function
>>> lambda x : 5*x+2
<function <lambda> at 0x101767ae8>
```



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

FUNCTIONAL

# PROGRAMMING PARADIGMS



# **DEBUGGING**

**(EVERYTHING THAT MAY DRIVE YOU  
MAD WHILE PROGRAMMING)**

# DEBUGGING

- Programming requires absolute precision and attention to detail

6 hours later



# DEBUGGING

- Programming requires absolute precision and attention to detail



**When You Find The  
Missing Semicolon**

# 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



# DEB U G G I N G

- 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

© T.A. BINKOWSKI, 2020

MODULE 1  
MPCS 50101

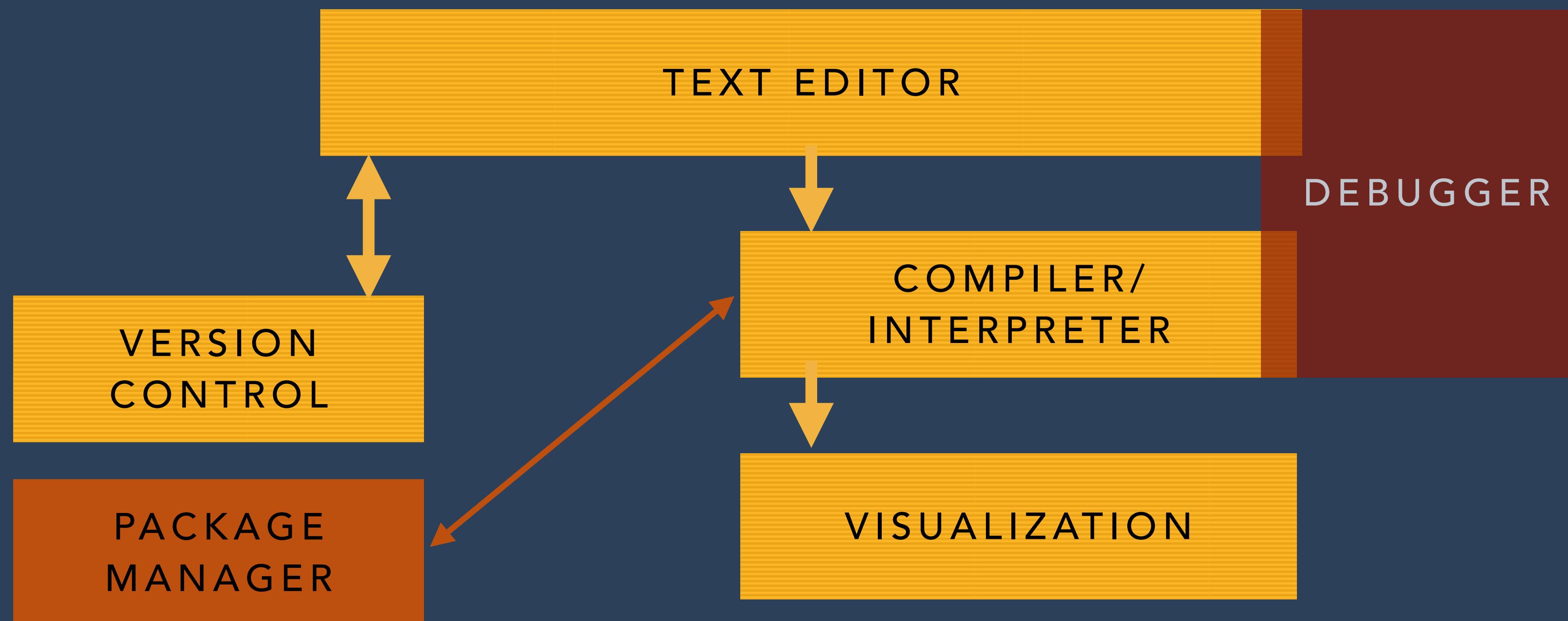


THE UNIVERSITY OF  
CHICAGO

# 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)

what is the best development environment

on to the top programming IDEs and code editors - JAXenter.com/overview-ides-code-editors-programmers-know-117138.html

is arguably the **best IDE** for developing desktop-enabled, mobile and web languages such as Java, Ruby, and PHP, ...

ed Development Environment (IDE) Software in 2016 | G2 ... crowd.com/categories/integrated-development-environment-ide

IDE. Visual Studio. IntelliJ IDEA. Xcode. PHPStorm. NetBeans. Eclipse. Visual Studio.

vs. IDEs: Which One Is Better For Programmers? .com/tag/text-editors-vs-ides-one-better-programmers/

DE Index, Eclipse and **Visual Studio** are the most popular IDEs at the time of this article. It's a cross-platform IDE that works well on Windows, OS X, Linux, and Solaris, and is used for Java, C, C++, PHP, and Python development.

egrated Developer Environments (IDEs) & Code Editors ... tbricks.com/top-integrated-developer-environments-ides/

that reason, there's not really a "best," one-size-fits-all IDE. These 48 are an excellent point if you're looking for an IDE to meet ...

Integrated Development Environments | Hackaday 010/08/24/top-5-integrated-development-environments/

any Integrated Development Environments are marketed towards ... This is the **best** IDE. I am not a big fan of Microsoft's Visual ...

ree IDE for Java Coding, Development & Programming solutions.com/.../the-top-11-free-ide-for-java-coding-development-pro.../

this article Alex takes a look at the top 11 Free IDE's for Java Coding, ... to compile a list of the best IDEs out there for Java Programmers, ...

IDEs and Editors for Programmers – DevZum 15/02/20-best-free-ides-and-editors-for-programmers/

ut the most sought after things to a web developer is **best IDE** Integrated Development Environment. These are the best code play grounds. And if you ...

of integrated development environments - Wikipedia media.org/wiki/Comparison\_of\_integrated\_development\_environments

es list notable software packages that are nominal IDEs; standalone tools such as build tools and GUI builders are not included.

best IDE? - Quora

# 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 ...

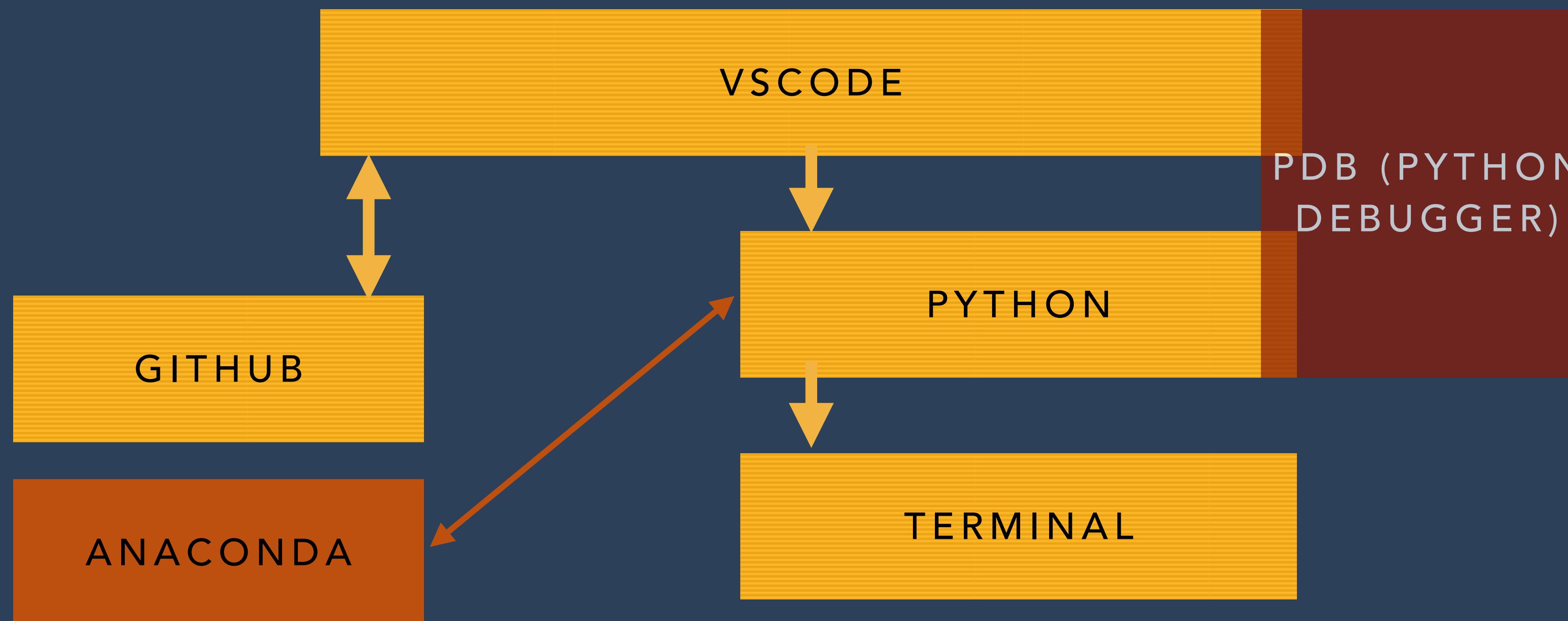
## Sublime Text: The text editor you'll fall in love with

<https://www.sublimetext.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)

The screenshot shows the PyCharm IDE interface. The top navigation bar displays the file path: registration/models.py - django-registration-0.8 - [~/Downloads/django-registration]. The main window has several panes:

- Project:** Shows the directory structure of the django-registration-0.8 project, including subfolders like docs, registration, backends, locale, management, tests, and files like \_\_init\_\_.py, admin.py, auth\_urls.py, forms.py, models.py, signals.py, urls.py, and views.py.
- Editor:** Displays the code for models.py. A specific function, create\_inactive\_user, is highlighted. The code creates a new User object with is\_active set to False, generates a RegistrationProfile, and sends an activation email if send\_email is True. It also defines a create\_profile function.
- Debug:** Shows the current stack trace in the Frames tab, with the main thread at the top. The frames listed are: \_setup, \_\_init\_\_.py:40; inner, functional.py:184; <module>, \_\_init\_\_.py:11; <module>, models.py:5; <module>, models.py:7; run, pydevd.py:1117; and <module>, pydevd.py:1473.
- Variables:** A list of variables and their values, such as ImproperlyConfigured (NoneType), \_\_builtins\_\_ (dict), \_\_doc\_\_ (NoneType), \_\_file\_\_ (NoneType), \_\_name\_\_ (NoneType), \_\_package\_\_ (NoneType), send (NoneType), and urllib (NoneType).
- Watches:** A pane for monitoring changes in watched variables.

A yellow callout bubble in the bottom right corner contains the text "PRODUCTIVE BUT OVERWHELMING".

```
def create_inactive_user(self, username, email, password, site, send_email=True):
    """
    Create a new, inactive ``User``, generate a ``RegistrationProfile`` and email its activation key to ``User``, returning the new ``User``.

    By default, an activation email will be sent to the new user. To disable this, pass ``send_email=False``.

    ...
    new_user = User.objects.create_user(username, email, password)
    new_user.is_active = False
    new_user.save()

    registration_profile = self.create_profile(new_user)

    if send_email:
        registration_profile.send_activation_email(site)

    return new_user
create_inactive_user = transaction.commit_on_success(create_inactive_user)

def create_profile(self, user):
    """
    Create a ``RegistrationProfile`` for a given ``User``, and return the ``RegistrationProfile``.

    The activation key for the ``RegistrationProfile`` will be a SHA1 hash, generated from a combination of the ``User`` username and a random salt.

    ...
    salt = hashlib.sha1(str(random.random())).hexdigest()[:15]
    username = user.username
    ...
    ...

    activation_key = hashlib.sha1(salt + user.username).hexdigest()
    user.registrationprofile.activation_key = activation_key
    user.registrationprofile.activation_key_expired = False
    user.registrationprofile.save()
    return user.registrationprofile
```

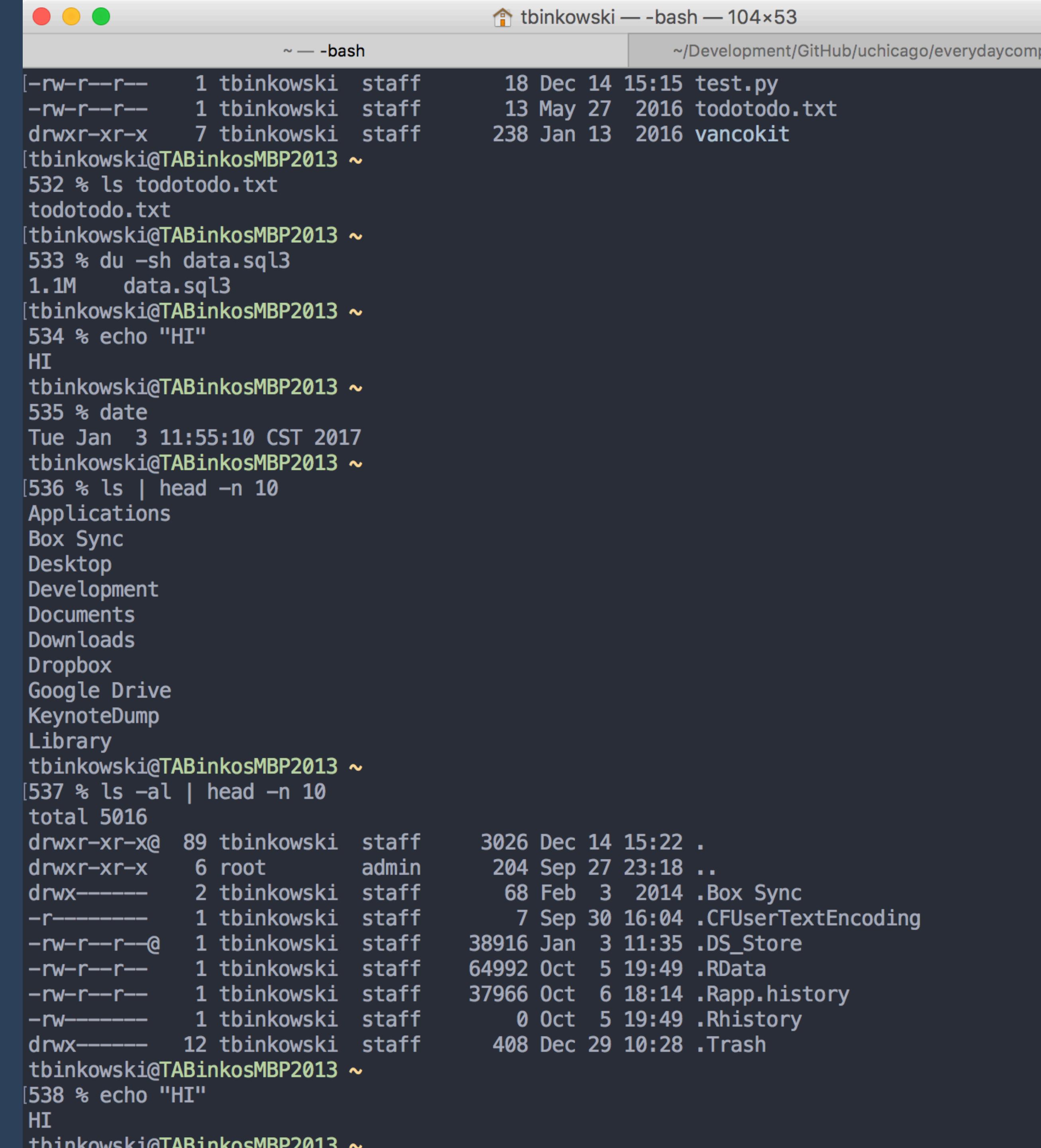
PRODUCTIVE BUT  
OVERWHELMING

# 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



A screenshot of a Mac OS X terminal window titled "tbinkowski — -bash — 104x53". The window shows a standard Unix-style command-line interface. The user has run several commands, including ls, du, echo, date, and ls -al, to demonstrate file listing and manipulation. The terminal also shows the user's home directory (~) and the path ~/Development/GitHub/uchicago/everydaycomp.

```
[tbinkowski@TABinkosMBP2013 ~]
532 % ls todotodo.txt
todotodo.txt
[tbinkowski@TABinkosMBP2013 ~]
533 % du -sh data.sql3
1.1M  data.sql3
[tbinkowski@TABinkosMBP2013 ~]
534 % echo "HI"
HI
tbinkowski@TABinkosMBP2013 ~
535 % date
Tue Jan  3 11:55:10 CST 2017
tbinkowski@TABinkosMBP2013 ~
[536 % ls | head -n 10
Applications
Box Sync
Desktop
Development
Documents
Downloads
Dropbox
Google Drive
KeynoteDump
Library
tbinkowski@TABinkosMBP2013 ~
[537 % ls -al | head -n 10
total 5016
drwxr-xr-x@ 89 tbinkowski  staff  3026 Dec 14 15:22 .
drwxr-xr-x   6 root       admin  204 Sep 27 23:18 ..
drwx-----  2 tbinkowski  staff   68 Feb  3  2014 .Box Sync
-r-----  1 tbinkowski  staff   7 Sep 30 16:04 .CFUserTextEncoding
-rw-r--r--@  1 tbinkowski  staff 38916 Jan  3 11:35 .DS_Store
-rw-r--r--  1 tbinkowski  staff  64992 Oct  5 19:49 .RData
-rw-r--r--  1 tbinkowski  staff  37966 Oct  6 18:14 .Rapp.history
-rw-----  1 tbinkowski  staff   0 Oct  5 19:49 .Rhistory
drwx----- 12 tbinkowski  staff  408 Dec 29 10:28 .Trash
tbinkowski@TABinkosMBP2013 ~
[538 % echo "HI"
HI
tbinkowski@TABinkosMBP2013 ~
```

# 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.

```
[tbinkowski@TABinkosMBP2013 ~] 532 % ls todotodo.txt
todotodo.txt
[tbinkowski@TABinkosMBP2013 ~] 533 % du -sh data.sql3
1.1M  data.sql3
[tbinkowski@TABinkosMBP2013 ~] 534 % echo "HI"
HI
tbinkowski@TABinkosMBP2013 ~
535 % date
Tue Jan  3 11:55:10 CST 2017
tbinkowski@TABinkosMBP2013 ~
[536 % ls | head -n 10
Applications
Box Sync
Desktop
Development
Documents
Downloads
Dropbox
Google Drive
KeynoteDump
Library
tbinkowski@TABinkosMBP2013 ~
[537 % ls -al | head -n 10
total 5016
drwxr-xr-x@  89 tbinkowski  staff   3026 Dec 14 15:22 .
drwxr-xr-x@    6 root       admin   204  Sep 27 23:18 ..
drwx-----@   2 tbinkowski  staff   68   Feb  3  2014 .Box Sync
-r-----@    1 tbinkowski  staff   7   Sep 30 16:04 .CFUserTextEncoding
-rw-r--r--@    1 tbinkowski  staff  38916 Jan  3 11:35 .DS_Store
-rw-r--r--@    1 tbinkowski  staff  64992 Oct  5 19:49 .RData
-rw-r--r--@    1 tbinkowski  staff  37966 Oct  6 18:14 .Rapp.history
-rw-----@    1 tbinkowski  staff   0   Oct  5 19:49 .Rhistory
drwx-----@   12 tbinkowski  staff  408  Dec 29 10:28 .Trash
tbinkowski@TABinkosMBP2013 ~
[538 % echo "HI"
HI
tbinkowski@TABinkosMBP2013 ~
[539 % df -h
Filesystem      Size   Used  Avail Capacity iused ifree %iused Mounted on
/dev/disk1     231Gi  215Gi  16Gi   94%  5222813 4289744466  0% /
devfs          182Ki  182Ki  0Bi   100%   628     0  100% /dev
map -hosts      0Bi   0Bi   0Bi   100%     0     0  100% /net
map auto_home    0Bi   0Bi   0Bi   100%     0     0  100% /home
localhost:/C5JsY5iABhJJYZvU_XQWta  231Gi  231Gi  0Bi   100%     0     0  100% /Volumes/Mobile
Backups
tbinkowski@TABinkosMBP2013 ~
540 %
```

# SHELL

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

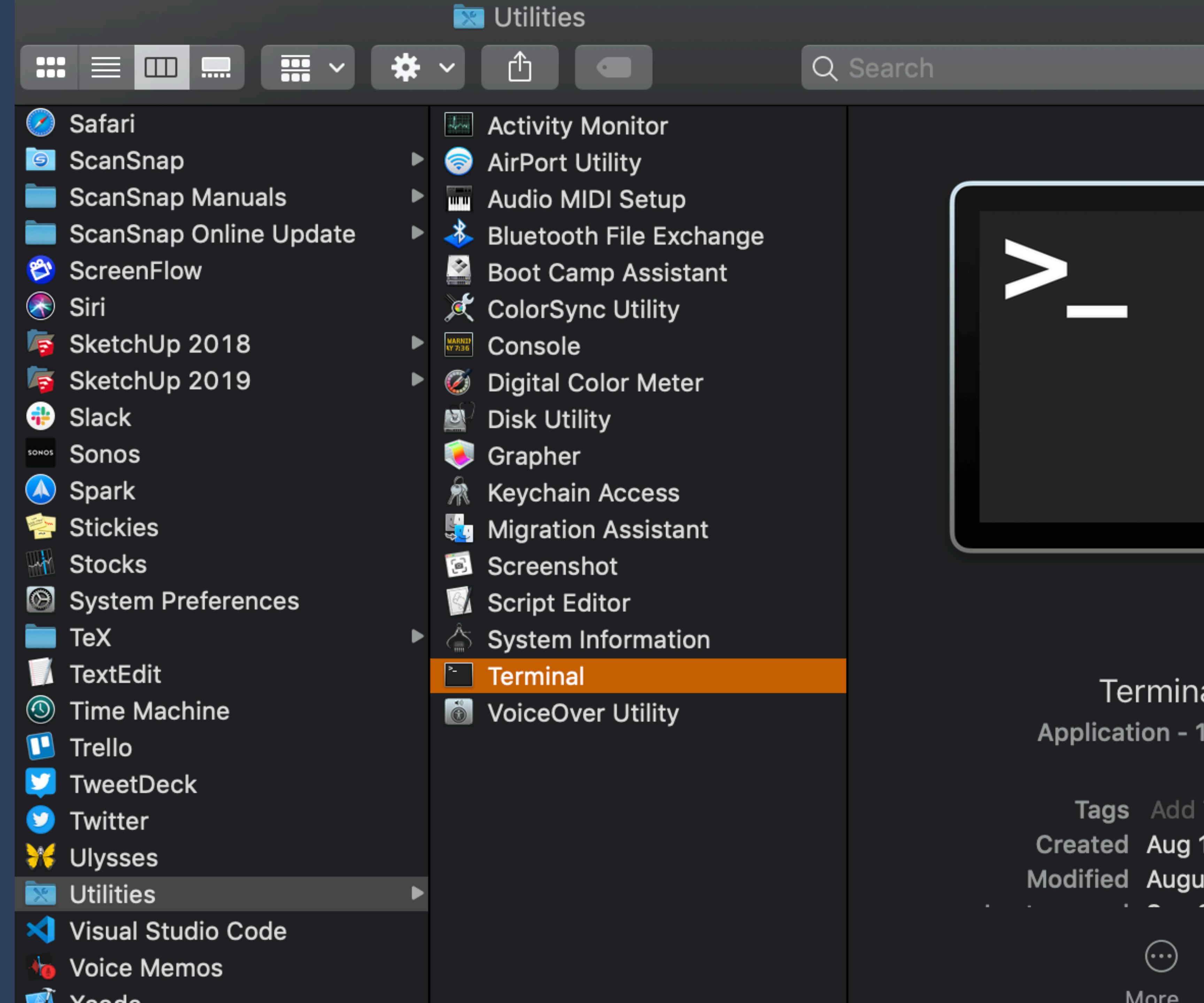
```
% ls  
% cd  
% mv  
% mkdir  
% rmdir  
% man  
% touch  
% rm -i  
% clear  
% cp  
% which  
% whoami
```

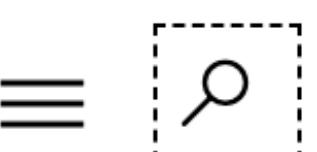
# SHELL

```
% touch "myfile.txt"  
% cp myfile.txt ~/Documents/myfile.txt  
% cd ~/Documents/  
% pwd  
% ls  
% rm myfile.txt
```

# SHELL

- Where is it?





# SHELL

- Powershell

Store ▾



EVERYONE

ESRB

**Windows Terminal (Preview)**

Microsoft Corporation • [Developer tools > Utilities](#)

Wish list

207

EARLY PREVIEW BUILD

This very early preview release includes basic support for assistive technology, however it has not undergone formal testing. You may still [More](#)

**Free**

[Get](#) [...](#)

[See System Requirements](#)

[Overview](#) [System Requirements](#) [Reviews](#) [Related](#)

**Available on**

PC

# TEXT EDITOR

# DEVELOPMENT ENVIRONMENT

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

The screenshot shows the Visual Studio Code interface. The left sidebar has icons for file, search, and extensions. The main area shows the 'EXTENSIONS: MARKETPLACE' tab with a list of installed extensions:

- Python 2019.6.24221 (by Microsoft) - Linting, Debugging (multi-threaded...)
- GitLens — Git supercharged 9.8.5 (by Eric Amodio) - Supercharge the Git capabilities built into VS Code
- C/C++ 0.24.0 (by Microsoft) - C/C++ IntelliSense, debugging, and build tools
- ESLint 1.9.0 (by Dirk Baeumer) - Integrates ESLint JavaScript into VS Code
- Debugger for Chrome 4.11.6 (by Microsoft) - Debug your JavaScript code in the browser
- Language Support for Java 0.47.0 (by Red Hat) - Java Linting, Intellisense, formattin...
- vscode-icons 8.8.0 (by VSCode Icons Team) - Icons for Visual Studio Code

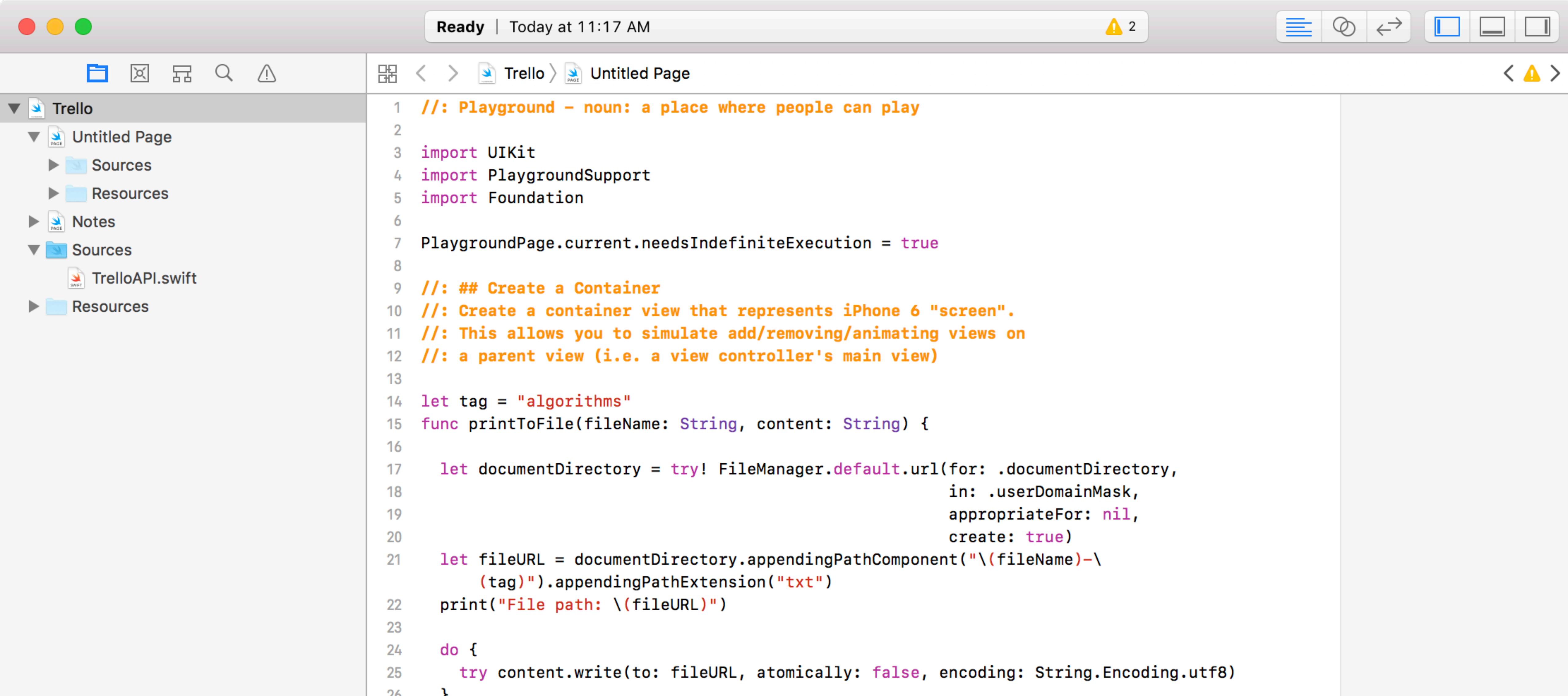
The code editor on the right shows a file named 'blog-post.js' with the following content:

```
import { graphql } from "gatsby"
import React from "react"
import Image from "gatsby-image"

export default ({ data }) =>
```

A yellow callout box at the bottom points to the URL [HTTPS://CODE.VISUALSTUDIO.COM](https://code.visualstudio.com).

# DEVELOPMENT ENVIRONMENT



The screenshot shows the Xcode IDE interface. The title bar says "Ready | Today at 11:17 AM" with a yellow warning icon showing "2". The toolbar includes standard icons for file operations. The navigation bar shows "Trello > Untitled Page". The left sidebar displays the project structure for "Trello": "Untitled Page" (with "Sources" and "Resources" subfolders), "Notes", "Sources" (containing "TrelloAPI.swift"), and "Resources". The main editor area contains the following Swift code:

```
//: Playground - noun: a place where people can play
import UIKit
import PlaygroundSupport
import Foundation

PlaygroundPage.current.needsIndefiniteExecution = true

//: ## Create a Container
//: Create a container view that represents iPhone 6 "screen".
//: This allows you to simulate add/removing/animating views on
//: a parent view (i.e. a view controller's main view)

let tag = "algorithms"
func printToFile(fileName: String, content: String) {

    let documentDirectory = try! FileManager.default.url(for: .documentDirectory,
                                                          in: .userDomainMask,
                                                          appropriateFor: nil,
                                                          create: true)
    let fileURL = documentDirectory.appendingPathComponent("\(fileName)-\
>tag").appendingPathExtension("txt")
    print("File path: \(fileURL)")

    do {
        try content.write(to: fileURL, atomically: false, encoding: String.Encoding.utf8)
    }
}
```

# DEVELOPMENT ENVIRONMENT

## FOLDERS

```
tensorflow
├── tensorflow
├── third_party
├── tools
└── util
    ├── .gitignore
    └── ACKNOWLEDGMENTS
        └── ADOPTERS.md
    └── AUTHORS
    └── BUILD
    └── configure
    └── CONTRIBUTING.md
    └── ISSUE_TEMPLATE.md
    └── LICENSE
    └── models.BUILD
    └── README.md
    └── RELEASE.md
    └── WORKSPACE
sqlite3
└── shell.c
└── sqlite3.c
└── sqlite3.h
└── sqlite3ext.h
```

```
base64.cc

34
35 void base64_encode(const uint8_t * data, size_t leng, char * dst,
36                     base64_charset variant)
37 {
38     const char * charset = (variant == base64_charset::URL_SAFE)
39             ? URL_SAFE_CHARSET
40             : STANDARD_CHARSET;
41
42     size_t src_idx = 0;
43     size_t dst_idx = 0;
44     for (; (src_idx + 2) < leng; src_idx += 3, dst_idx += 4)
45     {
46         uint8_t s0 = data[src_idx];
47         uint8_t s1 = data[src_idx + 1];
48         uint8_t s2 = data[src_idx + 2];
49
50         dst[dst_idx + 0] = charset[(s0 & 0xfc) >> 2];
51         dst[dst_idx + 1] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
52         dst[dst_idx + 2] = charset[((s1 & 0x0f) << 2) | (s2 & 0xc0) >> 6];
53         dst[dst_idx + 3] = charset[(s2 & 0x3f)];
54     }
55
56     if (src_idx < leng)
57     {
58         uint8_t s0 = data[src_idx];
59         uint8_t s1 = (src_idx + 1 < leng) ? data[src_idx + 1] : 0;
60
61         dst[dst_idx++] = charset[(s0 & 0xfc) >> 2];
62         dst[dst_idx++] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
63         if (src_idx + 1 < leng)
64             dst[dst_idx++] = charset[((s1 & 0x0f) << 2)];
65     }
66
67     dst[dst_idx] = '\0';
```

# DEVELOPMENT ENVIRONMENT

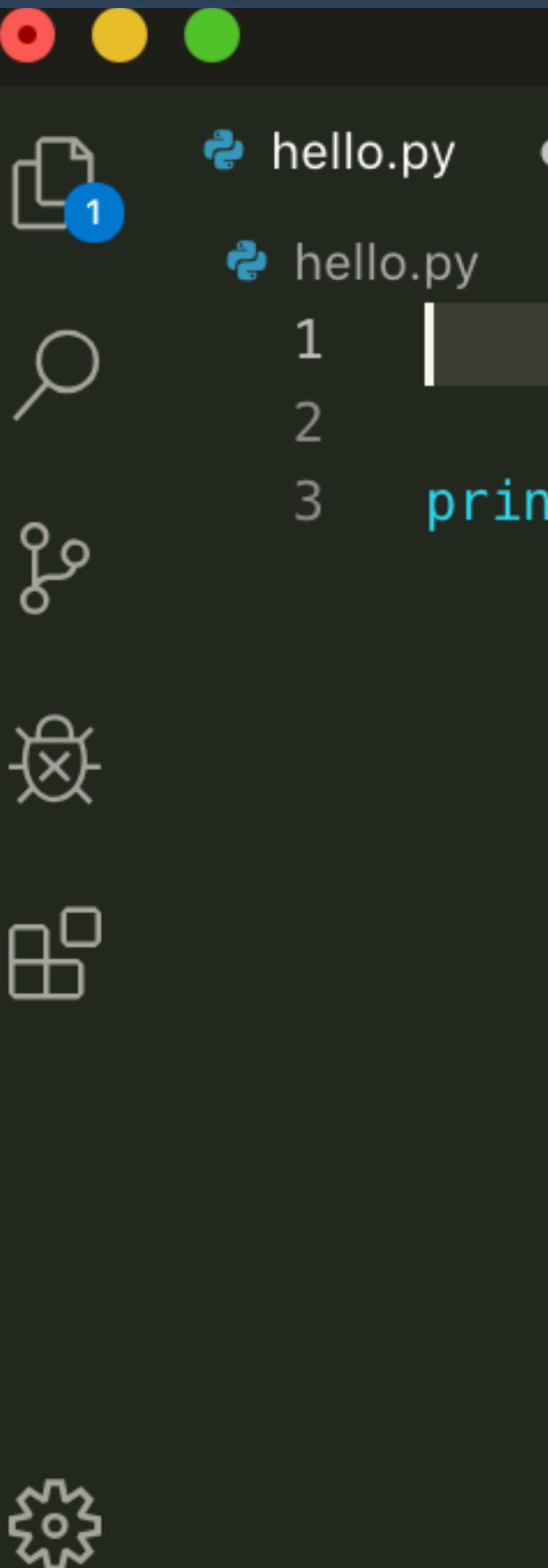
# DEVELOPMENT ENVIRONMENT

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

```
hello.py
hello.py
1
2
3 print("This is visual studio code")
```

Python 3.6.3 64-bit ('py36': conda) ✘ 0 ▲ 0

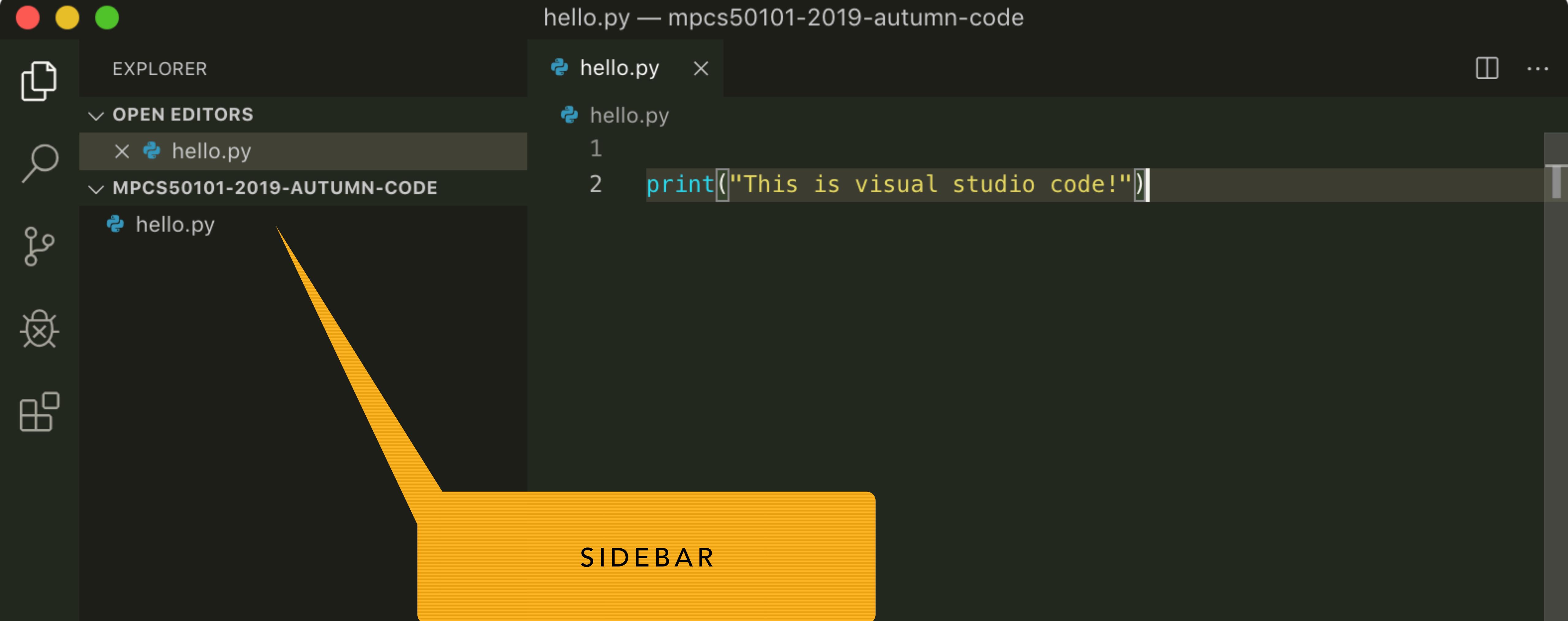
# DEVELOPMENT ENVIRONMENT



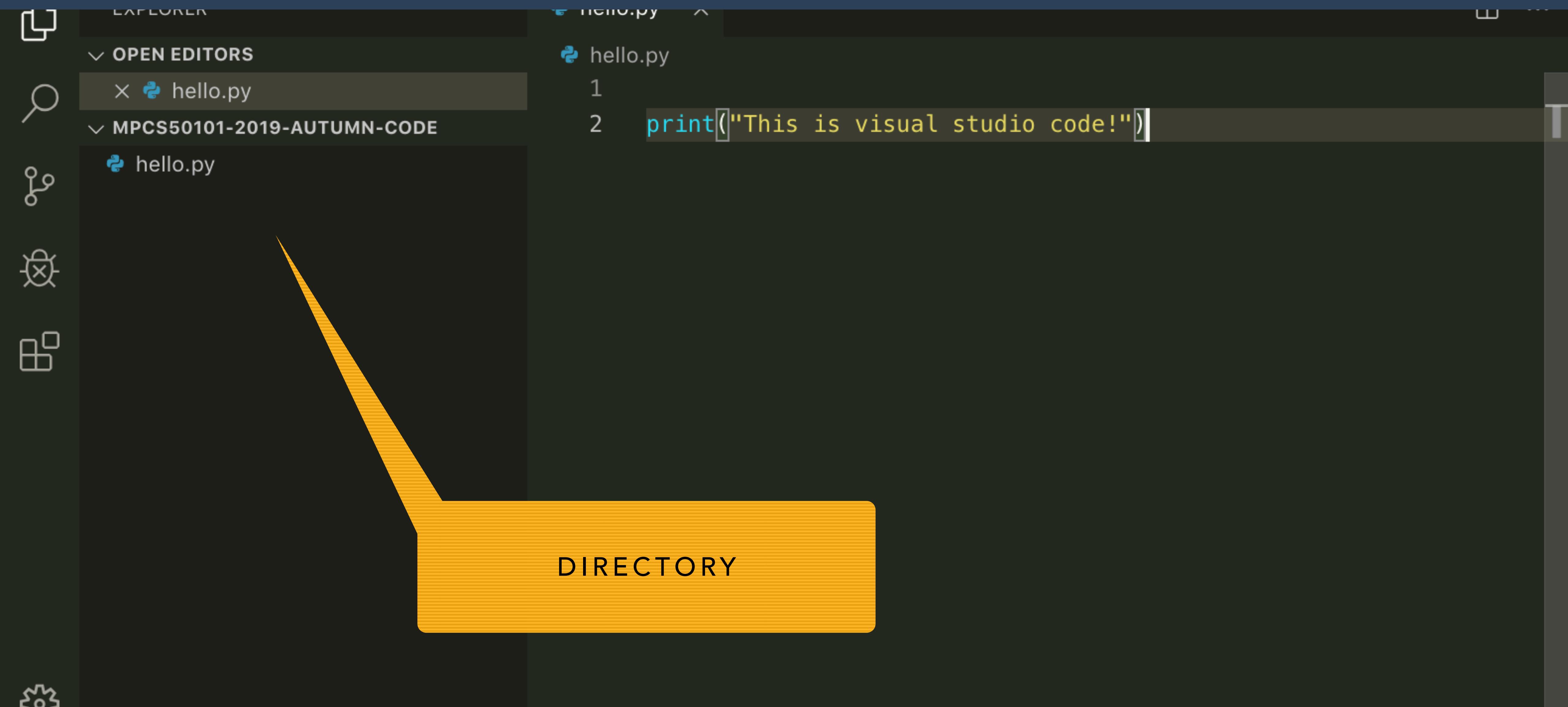
hello.py — mpcs50101-2019-autumn-code

- Command Palette... ⌘P
- Open View...
- Appearance
- Editor Layout
- Explorer ⌘E
- Search ⌘F
- SCM ⌘G
- Debug ⌘D
- Extensions ⌘X
- Output ⌘U
- Debug Console ⌘Y
- Terminal ⌘`
- Problems ⌘M
- Toggle Word Wrap ⌘Z
- Show Minimap
- ✓ Show Breadcrumbs
- Render Whitespace
- Render Control Characters

# DEVELOPMENT ENVIRONMENT

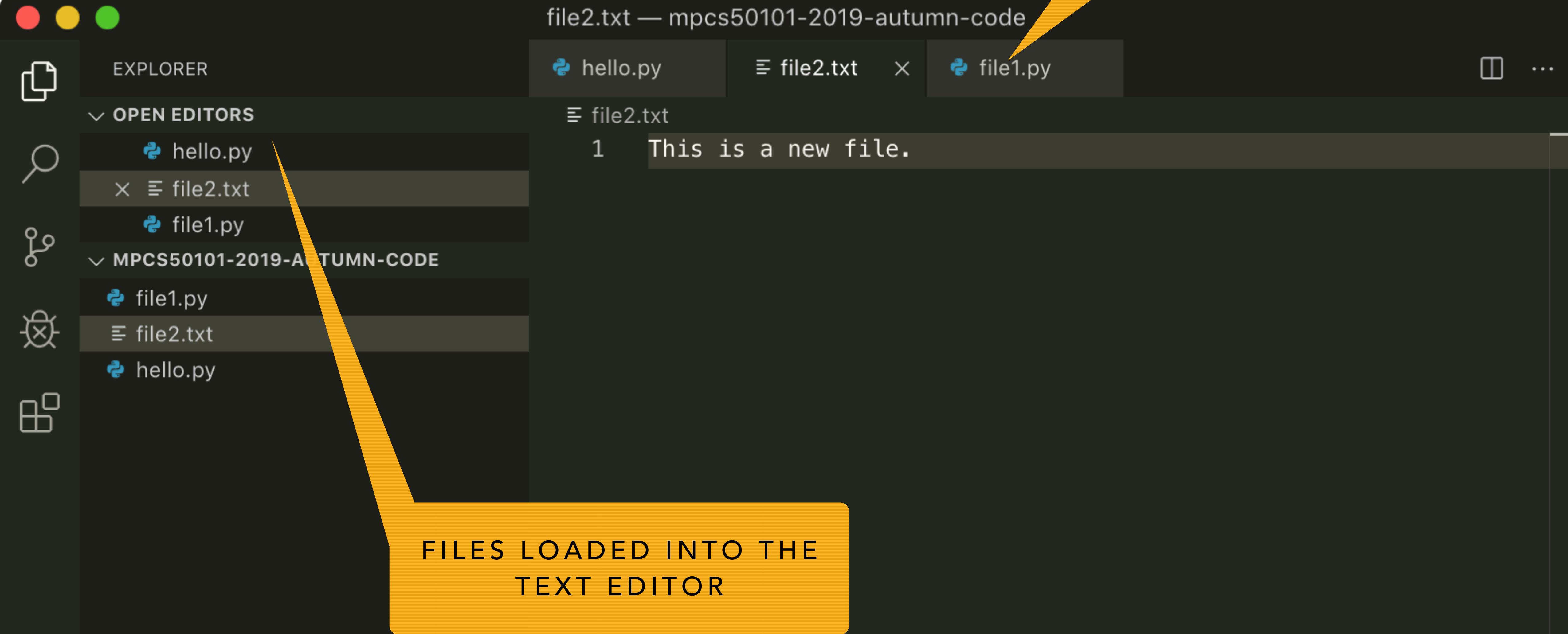


# DEVELOPMENT ENVIRONMENT



# DEVELOPMENT ENVIRONMENT

FILES LOADED INTO THE TEXT EDITOR



# DEVELOPMENT ENVIRONMENT

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "hello.py — mpcs50101-2019-autumn-code". The left sidebar has icons for recent files, search, file tree, and settings. The main editor area contains a single Python file named "hello.py" with the following code:

```
1
2
3 print("This is visual studio code!")
```

A yellow rectangular overlay covers the bottom portion of the screen, containing the word "SETTINGS" in black capital letters. In the bottom-left corner, there is a small gear icon.

# DEVELOPMENT ENVIRONMENT

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



TOPICS

Tutorial



## Getting Started with Python in VS Code



In this tutorial, you use Python 3 to create the simplest Python "Hello World" application in Visual Studio Code. By using the Python extension, you make VS Code into a great lightweight Python IDE (which you may find a productive alternative to PyCharm).

This tutorial introduces you to VS Code as a Python environment, primarily how to edit, run, and debug code through the following tasks:

- Install the Python extension for VS Code
- Download and Install Python 3 (for Linux/macOS or Windows)
- Write, run, and debug a Python "Hello World" Application
- Learn how to install packages by creating Python virtual environments
- Write a simple Python script to plot figures within VS Code

This tutorial is not intended to teach you Python itself. Once you are familiar with

# PROGRAMMING LANGUAGE

# PROGRAMMING LANGUAGE

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

The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation is the Python logo and a search bar. A main menu with tabs for About, Downloads, Documentation, Community, Success Stories, News, and Events is visible. A large central area displays Python code examples:

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

To the right of the code, a section titled "Compound Data Types" discusses lists, with a link to "More about lists in Python 3". Below this are five numbered buttons (1, 2, 3, 4, 5). Further down, a promotional message reads: "Python is a programming language that lets you work quickly and integrate systems more effectively. [»»» Learn More](#)". At the bottom, there are four footer sections: "Get Started", "Download", "Docs", and "Jobs".

**Compound Data Types**

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. [More about lists in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [»»» Learn More](#)

---

**Get Started**  
Whether you're new to programming or an experienced developer, it's easy to learn and use Python.  
[Start with our Beginner's Guide](#)

---

**Download**  
Python source code and installers are available for download for all versions! Not sure which version to use?  
[Check here.](#)  
Latest: [Python 3.6.0](#) - [Python 2.7.13](#)

---

**Docs**  
Documentation for Python's standard library, along with tutorials and guides, are available online.  
[docs.python.org](#)

---

**Jobs**  
Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the place to go.  
[jobs.python.org](#)

# 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

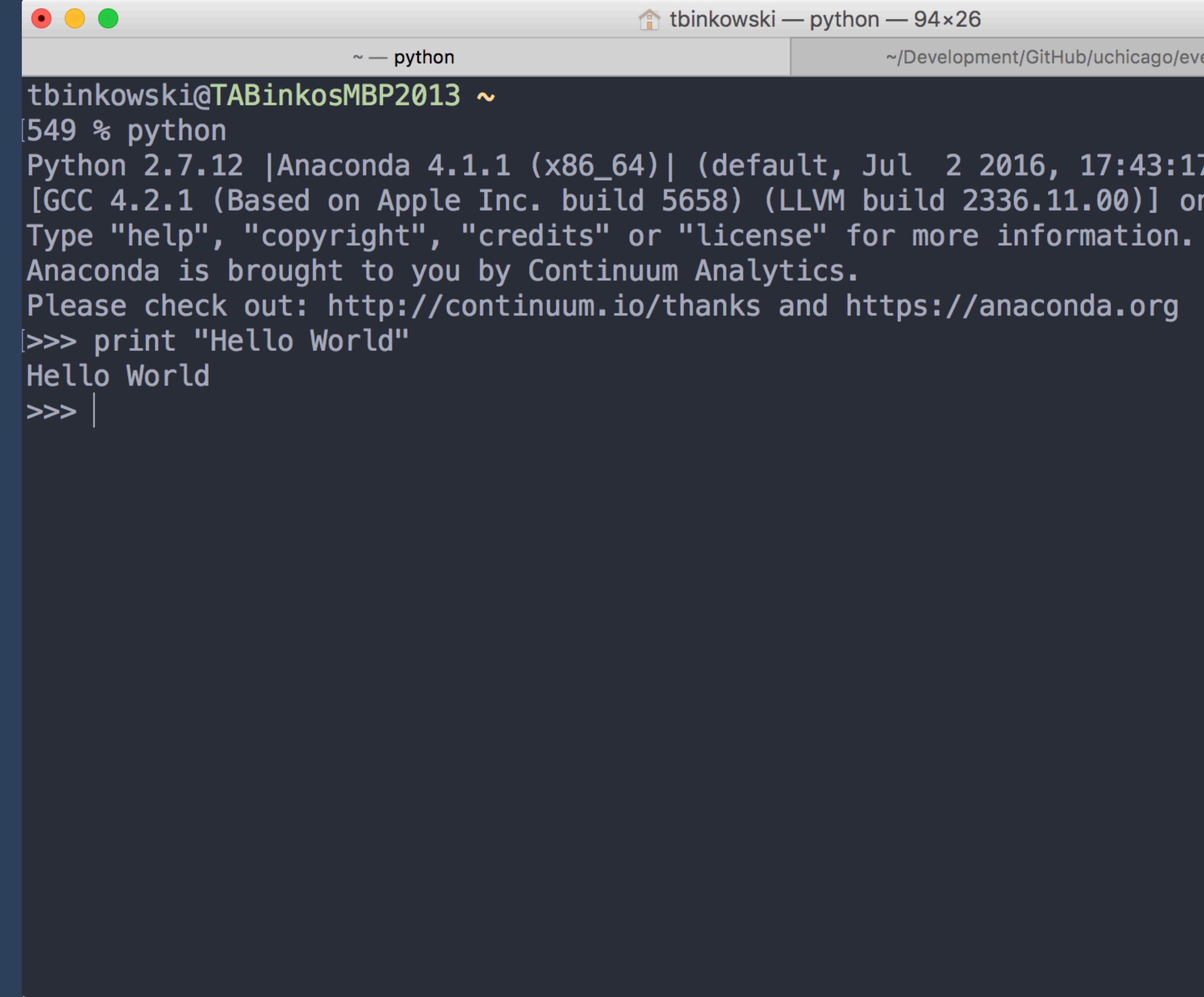
- Python is an interpreted language
- The interpreter can be run in two modes
  - Interactive mode
  - Script mode

```
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



A screenshot of a Mac OS X terminal window titled "tbinkowski — python — 94x26". The window shows the Python 2.7.12 interactive mode. The title bar includes the window icon, title, and dimensions. The terminal window has two panes: the left pane shows the command line (~ — python) and the right pane shows the output of the Python session. The output includes the Python version information, copyright notice, Anaconda branding, and a simple "Hello World" print statement.

```
tbinkowski@TABinkosMBP2013 ~
[549 % python
Python 2.7.12 |Anaconda 4.1.1 (x86_64)| (default, Jul 2 2016, 17:43:17)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
[>>> print "Hello World"
Hello World
>>> |
```

# PROGRAMMING LANGUAGES

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

The image shows a Mac OS X desktop with two windows. The top window is a code editor titled "hello\_world.py" showing a single line of Python code: "print "Hello World (from script mode)!"". The bottom window is a terminal window titled "-bash" showing the command "python hello\_world.py" being run, which outputs "Hello World (from script mode)".

```
hello_world.py — ~/Google Drive/g-Teaching/uchicago.codes/lecture
hello_world.py
x
1 print "Hello World (from script mode)!"
2

~/Google Drive/g-Teaching/uchicago.codes/lectures — -bash
tbinkowski@TABinkosMBP2013 ~/Google Drive/
[562 % python hello_world.py
Hello World (from script mode) !
tbinkowski@TABinkosMBP2013 ~/Google Drive/
563 % |
```

# 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

DA  
ence Platform  
thon

N

**ANACONDA FUSION**

Excel® Data Science

**ANACONDA FUSI**

Excel® Data Science

**ANACONDA**  
ENTERPRISE NOTE

Data Science Collab

**ANACONDA SCAL**

Distributed Comput

**ANACONDA ACC**

High Performance C

**ANACONDA REPO**

Data Science Packag

**ANACONDA DIS**

Open Data Science

# 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



ANACONDA COMMUNITY

**ANACONDA**  
Leading Open Data Science Platform  
Powered by Python

[DOWNLOAD NOW](#)

**ANACONDA FUSION**

Excel® Data Science

# DEVELOPMENT ENVIRONMENT

Google anaconda

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? ▾

**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



**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 ...

Nicki Minaj - Anaconda - YouTube

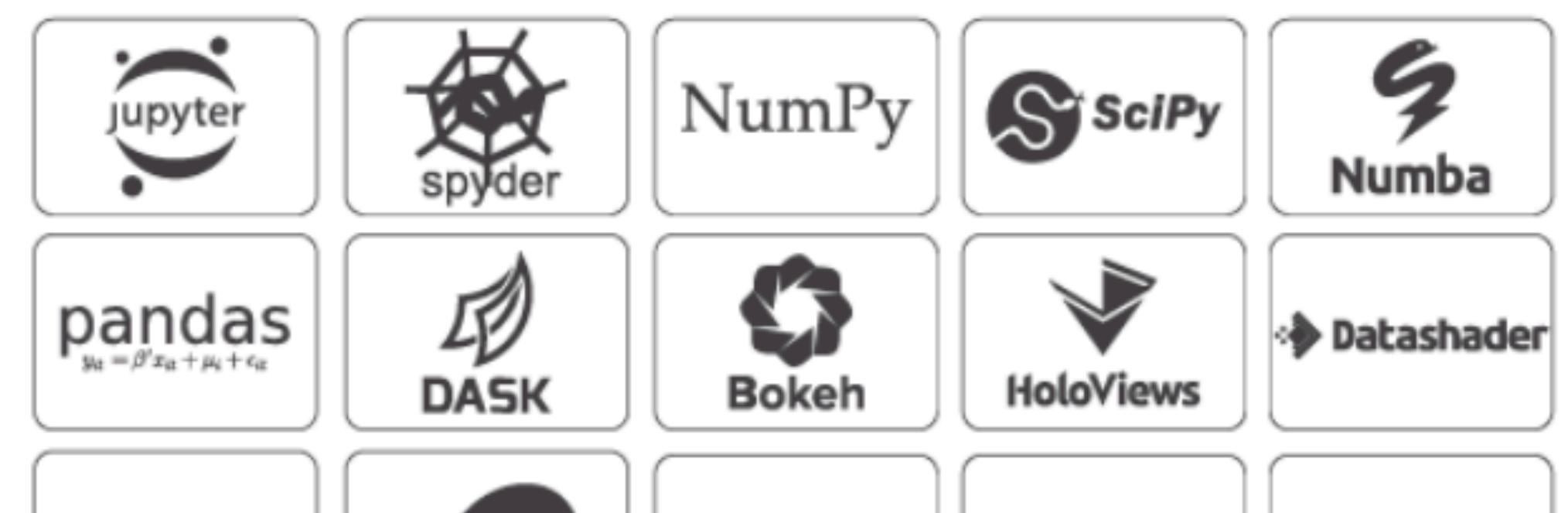
# DEVELOPMENT ENVIRONMENT

## 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, it is the industry standard for development, testing,



# DEVELOPMENT ENVIRONMENT

## Python 3.7 version

DOWNLOAD THE  
LATEST 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

The screenshot shows a Mac OS X window titled "Install Anaconda2". The window has standard OS X window controls (red, yellow, green buttons) and a lock icon in the top right corner. On the left, a vertical navigation bar lists steps: "Introduction" (selected), "Read Me", "License", "Destination Select", "Installation Type", "Installation", and "Summary". The main content area displays the text: "Welcome to the Anaconda2 Installer" followed by "You will be guided through the steps necessary to install this software."

Welcome to the Anaconda2 Installer

You will be guided through the steps necessary to install this software.

- **Introduction**
- Read Me
- License
- Destination Select
- Installation Type
- Installation
- Summary

# DEVELOPMENT ENVIRONMENT

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

```
% cd ~  
% ls
```

The screenshot shows a Mac OS X desktop with two terminal windows. The top terminal window, titled 'Margret — -bash — 89x18', displays the contents of the user's home directory (~). It shows standard system folders like Library, Movies, and Music. A large orange arrow points from the 'anaconda' folder in the bottom terminal window towards the 'anaconda' folder in the top window, indicating its location. The bottom terminal window, titled 'anaconda — -bash — 89x18', shows the detailed contents of the 'anaconda' directory, including subfolders like include, lib, and share.

```
$ ls  
Library  
Movies  
Music  
Pictures  
Public  
anaconda
```

```
Margrets-Pro:anaconda Margret$ ls  
Anaconda-Navigator.app  include  
LICENSE.rst              lib  
README.rst               libexec  
bin                      mkspecs  
conda-meta                phrasebooks  
doc                      pkgs  
envs                     plugins  
etc                      python.app  
Margrets-Pro:anaconda Margret$
```

# DEVELOPMENT ENVIRONMENT

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

```
% which python
```

```
Margrets-Pro:~ Margret$ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

```
Margrets-Pro:~ Margret$ python
Python 2.7.12 |Anaconda 4.2.0 (x86_64)| (default, Jul 2 2016, 17:43:17)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> █
```

# DEVELOPMENT ENVIRONMENT

```
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@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
erRoot/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

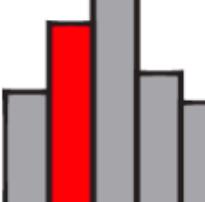
WE ARE NOT GOING  
TO BE USING VIRTUAL  
ENVIRONMENTS

- 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 pyenv. It allows virtual environments to be used on versions of Python prior to 3.4

# DEVELOPMENT ENVIRONMENT

Home Applications on base (root) Channels Refresh

Environments Learning Community Documentation Developer Blog

 JupyterLab 1.0.2 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. <a href="#">Launch</a>	 Notebook 6.0.0 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. <a href="#">Launch</a>	 Spyder 3.3.6 Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features <a href="#">Launch</a>	 Glueviz 0.13.3 Multidimensional data visualization across files. Explore relationships within and among related datasets. <a href="#">Install</a>
 Orange 3 3.19.0 Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. <a href="#">Install</a>	 RStudio 1.1.456 A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. <a href="#">Install</a>	 VS Code 1.38.1 Streamlined code editor with support for development operations like debugging, task running and version control. <a href="#">Install</a>	<p>APPLICATIONS SUPPORTED BY ANACONDA</p>

# DEVELOPMENT ENVIRONMENT

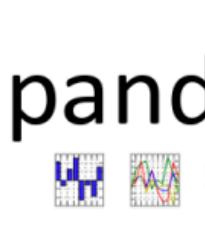
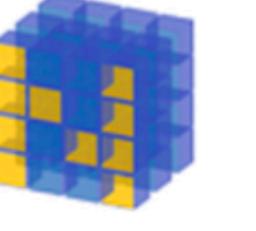
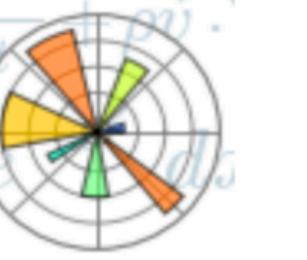
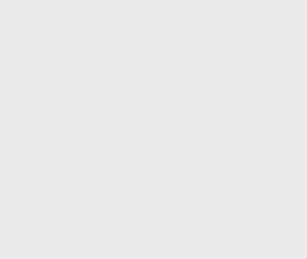
The screenshot shows a software interface for managing development environments. On the left is a sidebar with icons for Environments, Projects (beta), Learning, Community, Documentation, and Developer Blog. The main area has a title 'root' and a play button icon. Below is a table listing packages:

Name	Description	Version
alabaster	Configurable, python 2+3 compatible sphinx theme	0.7.10
anaconda		4.4.0
anaconda-client	Anaconda.org command line client library	1.6.3
anaconda-project	Reproducible, executable project directories	0.6.0
appnope		0.1.0
appscript		1.0.1
argcomplete		1.0.0
asn1crypto		0.22.0
astroid	Abstract syntax tree for python with inference support	1.4.9
astropy	Community-developed python library for astronomy	1.3.2
babel	Utilities to internationalize and localize python applications	2.1.0
backports		1.0
backports-abc		0.5
backports.functools-lru-cache		1.4
backports.shutil-get-terminal-size		1.0
backports.ssl-match-hostname		3.4.0.2
backports._abc	Backport of recent additions to the 'collections.abc' module	0.5

A yellow callout box with the text "MANAGE DIFFERENT ENVIRONMENTS" points to the table.

# DEVELOPMENT ENVIRONMENT

-  Environments
-  Projects (beta)
-  Learning
-  Community

		 ANACONDA®	 pandas			
Python Tutorial	Python Reference	Anaconda Package List	Pandas Documentation	Numpy Documentation	Scipy Documentation	Matplotlib Documentation
<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Read</a>
	 ANACONDA®	 ANACONDA®	 ANACONDA®	 ANACONDA®	 ANACONDA®	 ANACONDA®
Bokeh User Guide	Anaconda Cloud Documentation	Anaconda Documentation	Anaconda Navigator Documentation	Continuum Training Program	Anaconda Skills Accelerator Program	
<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Read</a>	<a href="#">Explore</a>	<a href="#">Explore</a>	

Documentation

Developer Blog

LEARNING RESOURCES

# DEVELOPMENT ENVIRONMENT

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

EASILY UPDATE  
PYTHON AND  
PACKAGES

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

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 programs 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 supports can be called under program control.

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

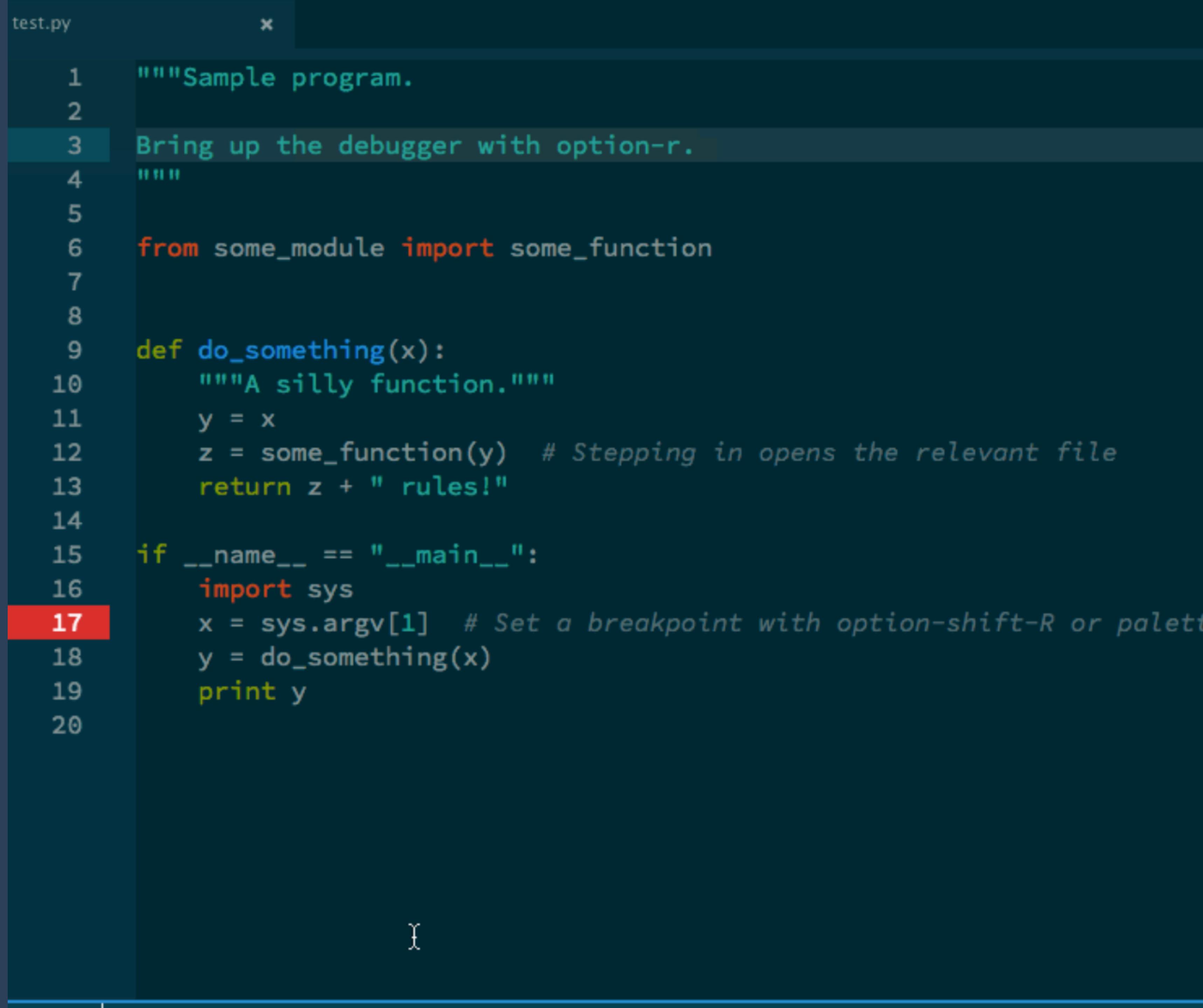
The debugger's prompt is `(Pdb)`. Typical usage to run a program under control of the debugger:

```
>>> 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.*

# DEBUGGER

- Access pdb via text editor and/or command line



A screenshot of a code editor showing a Python script named `test.py`. The code contains several numbered lines (1 through 20) and some explanatory text. Lines 17 and 20 are highlighted with red rectangles. The code includes a multi-line string, imports, a function definition, and a conditional block. A note about stepping in and breakpoints is present.

```
1 """Sample program.  
2  
3 Bring up the debugger with option-r.  
4 """  
5  
6 from some_module import some_function  
7  
8  
9 def do_something(x):  
10     """A silly function."""  
11     y = x  
12     z = some_function(y) # Stepping in opens the relevant file  
13     return z + " rules!"  
14  
15 if __name__ == "__main__":  
16     import sys  
17     x = sys.argv[1] # Set a breakpoint with option-shift-R or palette  
18     y = do_something(x)  
19     print y  
20 }
```

# DEBUGGER

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



```
x = 100  
print(x)  
  
y = 200  
print(y)  
  
z = x + y  
print(z)
```

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-button':  
12 12        },  
13 13    ui: {  
14 14      content: '.review-activity--content'  
15 15    },  
16 16    template: headerTemplate,  
17 17    onRender: function () {  
18 18      initialize: function () {  
19 19        this.collectionView = new ReviewActivityView({  
20 19          el: this.ui.content.get(0),  
21 21          collection: this.collection  
22 22        }).render();  
23 23      } );  
24 24    },  
25 25    onRender: function () {  
26 26      this.collectionView.setElement(this.el);  
27 27      this.collectionView.render();  
28 28    },  
29 29    triggerOnClose: function () {  
30 30      this.collectionView.remove();  
31 31    }  
32 32  });  
33 33  
```

# 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

The screenshot shows a GitHub repository interface. At the top, it displays the repository path: uchicago/everydaycomputing.org. Below this, there are tabs for 'dev' (selected), 'No Uncommitted Changes', and 'History'. A navigation bar at the top right includes 'Update from master' and 'View Branch'. The main area shows two branches: 'master' and 'dev'. The 'dev' branch has a blue icon indicating it is checked out. Below the branches is a list of commits:

- Add Methodology search and filter. (22 days ago by tabinks)
- Add link to articles directly from the cluster creator. (1 month ago by tabinks)
- Add a note to yaml. (1 month ago by tabinks)
- Add another index. (1 month ago by tabinks)
- Add new index. (1 month ago by tabinks)
- Remove the 100 limit on the article fetch. (1 month ago by tabinks)
- Update (1 month ago by tabinks)
- Update links for bios. (1 month ago by tabinks)
- Fix Quinn links. (1 month ago by tabinks)
- Update bios. (1 month ago by tabinks)
- Fix formatting issue. (1 month ago by tabinks)
- Merge branch 'master' into dev (4 months ago by tabinks)
- Merge branch 'master' into dev (4 months ago by tabinks)

On the right side, a detailed view of the 'everydaycomputing.py' file is shown, comparing changes between the 'master' and 'dev' branches. The code is color-coded: purple for deleted lines, green for added lines, and grey for unchanged lines. The diff highlights several changes, including the addition of a GoalHandler class and various updates to the MainPage and Article handlers.

```
@@ -49,6 +49,20 @@ class MainPage(webapp2.RequestHandler):
    # Loop through the dictionary and print info (for debugging)
    self.response.write(structured_dictionary)

+class GoalHandler(webapp2.RequestHandler):
+    def get(self):
+        """ """
+        self.response.headers['Content-Type'] = 'text/plain'
+        query = LearningGoal.query().order(-Article.timestamp.created)
+        goals = query.fetch()
+        for goal in goals:
+            self.response.write(goal.goal + "\n")
+
+        #self.response.write('Everyday computing')
+        #self.response.headers['Content-Type'] = 'text/csv'
+
+    @for item in structured_dictionary:
+        # self.response.write("%s %s - %s\n" %
+        (item['lessonNumber'], item['title'], item['description']))

... ...
@@ -66,9 +80,10 @@ APP = webapp2.WSGIApplication([
    webapp2.Route('/article/goal/<task:(insert|update)', ArticleGoalHandler, 'user-project-goals'),
    webapp2.Route('/article/goal/<article_key>/<learning_goal_key>', ArticleGoalHandler),
    webapp2.Route('/article/edit/<category>/<article_id>', ArticleCategoryEditHandler, 'user-project-categories'),
    webapp2.Route('/goals/', GoalHandler, 'user-project-goals')

+webapp2.Route('/goals/', GoalHandler, 'user-project-goals')
```

# VERSION CONTROL

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

The screenshot shows a GitHub repository interface. At the top, it displays the repository path: uchicago/everydaycomputing.org. Below this, there are buttons for 'dev' (selected), 'No Uncommitted Changes', and 'History'. A timeline at the top shows the 'master' branch above the 'dev' branch, which is currently selected.

The main area shows a list of commits on the 'dev' branch:

- Add Methodology search and filter. (22 days ago by tabinks)
- Add link to articles directly from the cluster creator. (1 month ago by tabinks)
- Add a note to yaml. (1 month ago by tabinks)
- Add another index. (1 month ago by tabinks)
- Add new index. (1 month ago by tabinks)
- Remove the 100 limit on the article fetch. (1 month ago by tabinks)
- Update (1 month ago by tabinks)
- Update links for bios. (1 month ago by tabinks)
- Fix Quinn links. (1 month ago by tabinks)
- Update bios. (1 month ago by tabinks)
- Fix formatting issue. (1 month ago by tabinks)
- Merge branch 'master' into dev (4 months ago by tabinks)
- Merge branch 'master' into dev (4 months ago by tabinks)

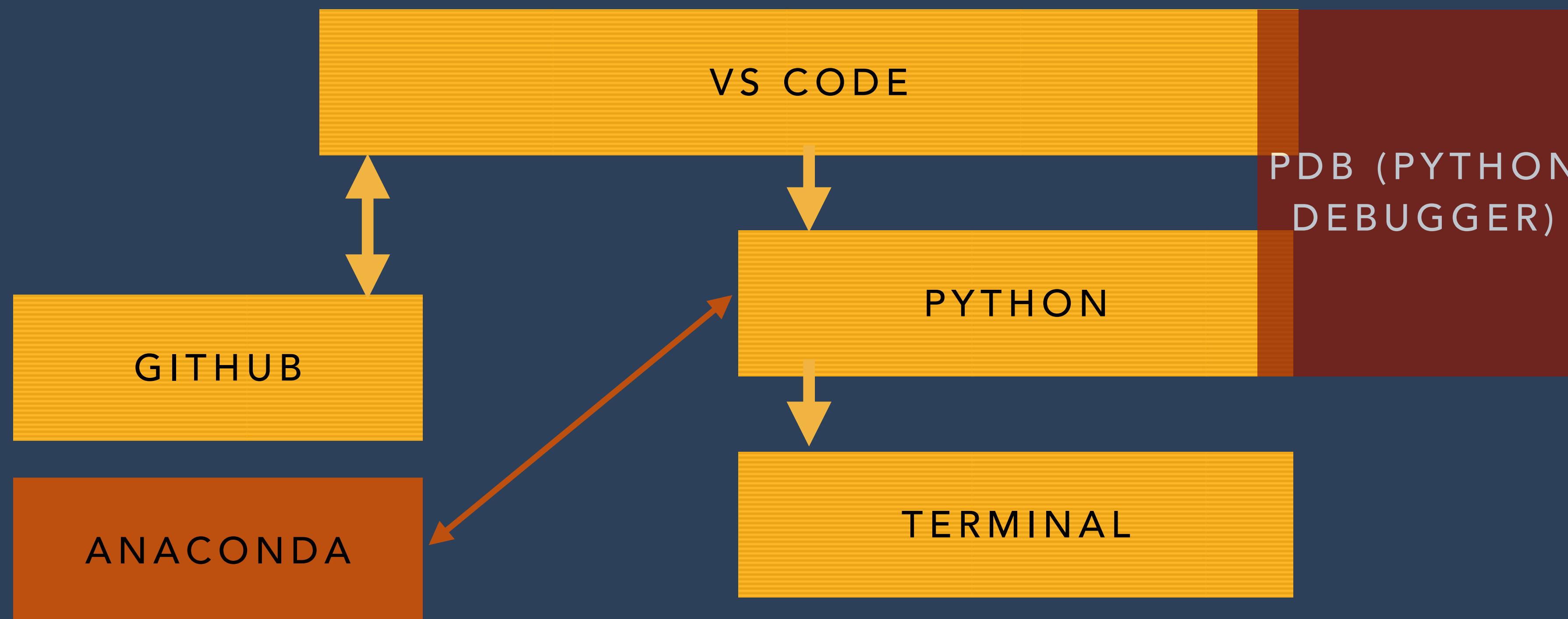
On the right, a detailed view of the 'everydaycomputing.py' file shows the code and the changes made in the last commit. The changes are highlighted in green, indicating additions. The code snippet includes imports, class definitions, and various methods for handling requests and rendering responses.

```
everydaycomputing.py
@@ -49,6 +49,20 @@ class MainPage(webapp2.RequestHandler):
    # Loop through the dictionary and print
    info (for debugging)
    self.response.write(structured_dictionary)

+ class GoalHandler(webapp2.RequestHandler):
+     def get(self):
+         """
+         self.response.headers['Content-Type']
+         query = LearningGoal.query().order(-
+             Article.timestamp.created)
+         goals = query.fetch()
+         for goal in goals:
+             self.response.write(goal.goal + "\n")
+
+         #self.response.write('Everyday computing')
+         #self.response.headers['Content-Type']
+         csv'
+
+     @@@ -66,9 +80,10 @@ APP = webapp2.WSGIApplication([
+         webapp2.Route('/article/goal/<task:(insert|update)', ArticleGoalHandler,
+             'user-project')
+         webapp2.Route('/article/goal/<article_key>/<learning_goal_key>', ArticleGoalHandler,
+             'user-project')
+         webapp2.Route('/article/edit/<category>/<article_id>', ArticleCategoryEditHandler,
+             'user-project')
+         webapp2.Route('/goals/', GoalHandler, 'user-project')
```

**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

```
1  
2 print("Hello World!")
```

```
file1.py file2.txt hello.py  
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teac  
s/mpcs50101/mpcs50101-2019-autumn/mpcs50101-2019  
[503 % python hello.py  
Hello World!  
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teac  
s/mpcs50101/mpcs50101-2019-autumn/mpcs50101-2019  
504 %
```

# THE END

MPCS 50101  
MODULE 1



THE UNIVERSITY OF  
CHICAGO

©TTAA .BB INKKOWSKI ,22019