CS303E: Elements of Computers and Programming

A First Look at Python

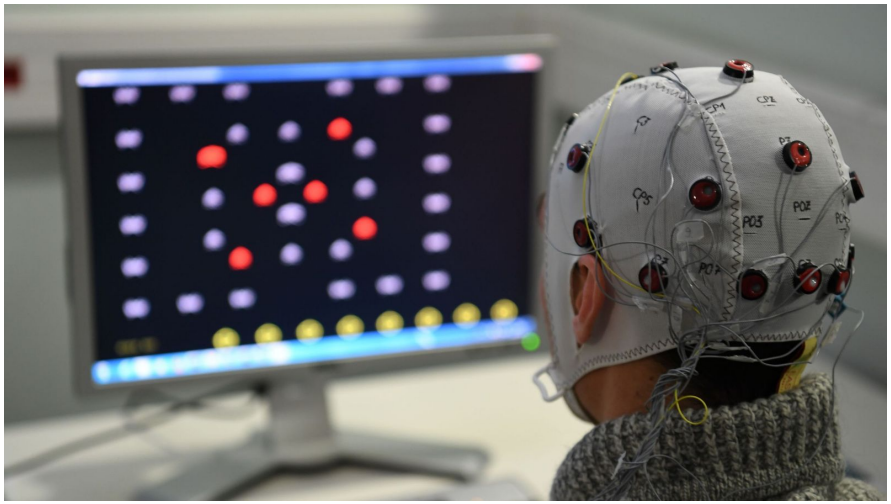
Dr. Bill Young
Department of Computer Science
University of Texas at Austin

Last updated: August 27, 2024 at 14:19

Some Thoughts about Programming

"The only way to learn a new programming language is by writing programs in it." –B. Kernighan and D. Ritchie, Developers of C

"Computers are good at following instructions, but not at reading your mind." –D. Knuth, Turing Award Winner

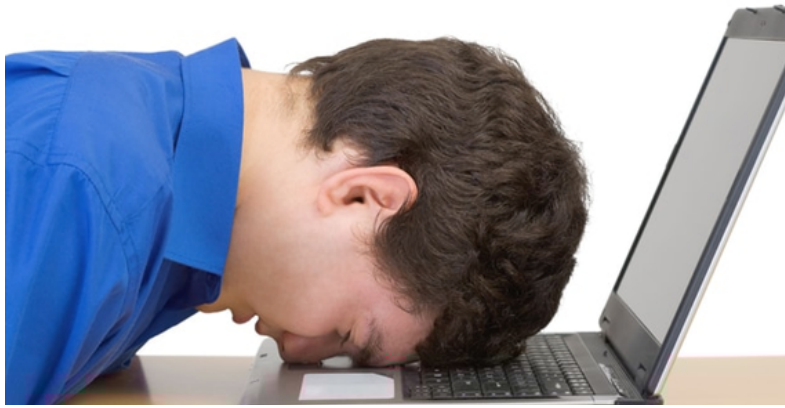


Some Thoughts about Programming

Program:

n. A magic spell cast over a computer allowing it to turn one's input into error messages.

tr. v. To engage in a pastime similar to banging one's head against a wall, but with fewer opportunities for reward.



"A computer lets you make mistakes faster than any invention in human history—with the possible exception of handguns and Tequila." –Mitch Ratcliffe, digital thinker

What is Python?

Python is a high-level programming language developed by Guido van Rossum in the Netherlands in the late 1980s, released in 1991.

As of 2021, most popular language, ahead of C and Java. Programming Language of the Year in 2007, 2010, 2018, 2020.

It's named after the British comedy troupe Monty Python.

Why Python?

Python is a simple but powerful language, with features that make it an excellent first programming language.

- Easy and intuitive mode of interacting with the system.

Why Python?

Python is a simple but powerful language, with features that make it an excellent first programming language.

- Easy and intuitive mode of interacting with the system.
- Clean syntax that is concise. You can say/do a lot with few words.

Why Python?

Python is a simple but powerful language, with features that make it an excellent first programming language.

- Easy and intuitive mode of interacting with the system.
- Clean syntax that is concise. You can say/do a lot with few words.
- Design is compact. You can carry the most important language constructs in your head.

Why Python?

Python is a simple but powerful language, with features that make it an excellent first programming language.

- Easy and intuitive mode of interacting with the system.
- Clean syntax that is concise. You can say/do a lot with few words.
- Design is compact. You can carry the most important language constructs in your head.
- There is a very powerful library of useful functions available.

You can be productive quite quickly. You will be spending more time solving problems and writing code, and less time grappling with the idiosyncrasies of the language.

Hello, World!

By convention, often the first program one writes in a language is the **Hello, World** program, to print the string "Hello, World!"

Here's what that looks like in Java:

```
// Our first Java program
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Hello, World!

By convention, often the first program one writes in a language is the **Hello, World** program, to print the string "Hello, World!"

Here's what that looks like in Java:

```
// Our first Java program
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

And here it is in Python:

```
# Our first Python program.
print("Hello, World!")
```

Zen of Python

The core philosophy of Python is summarized in the document *The Zen of Python*, which includes items such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

BTW: Python will display the Zen principles if you type:

```
import this
```

What is Python?

Python is a **general purpose** programming language. That means you can use Python to write code for any programming tasks.

Python was used to write code for:

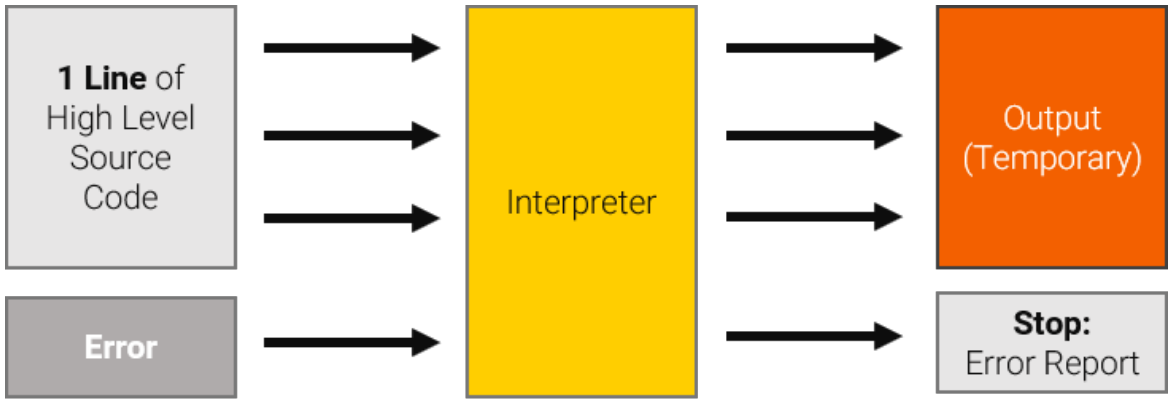
- the Google search engine
- mission critical projects at NASA
- financial transactions at the NY Stock Exchange
- the grading scripts for this class

What is Python?

Python is an **object-oriented** programming language.
Object-oriented programming is a powerful approach to developing reusable software.

Much more on that topic later!

What is Python?



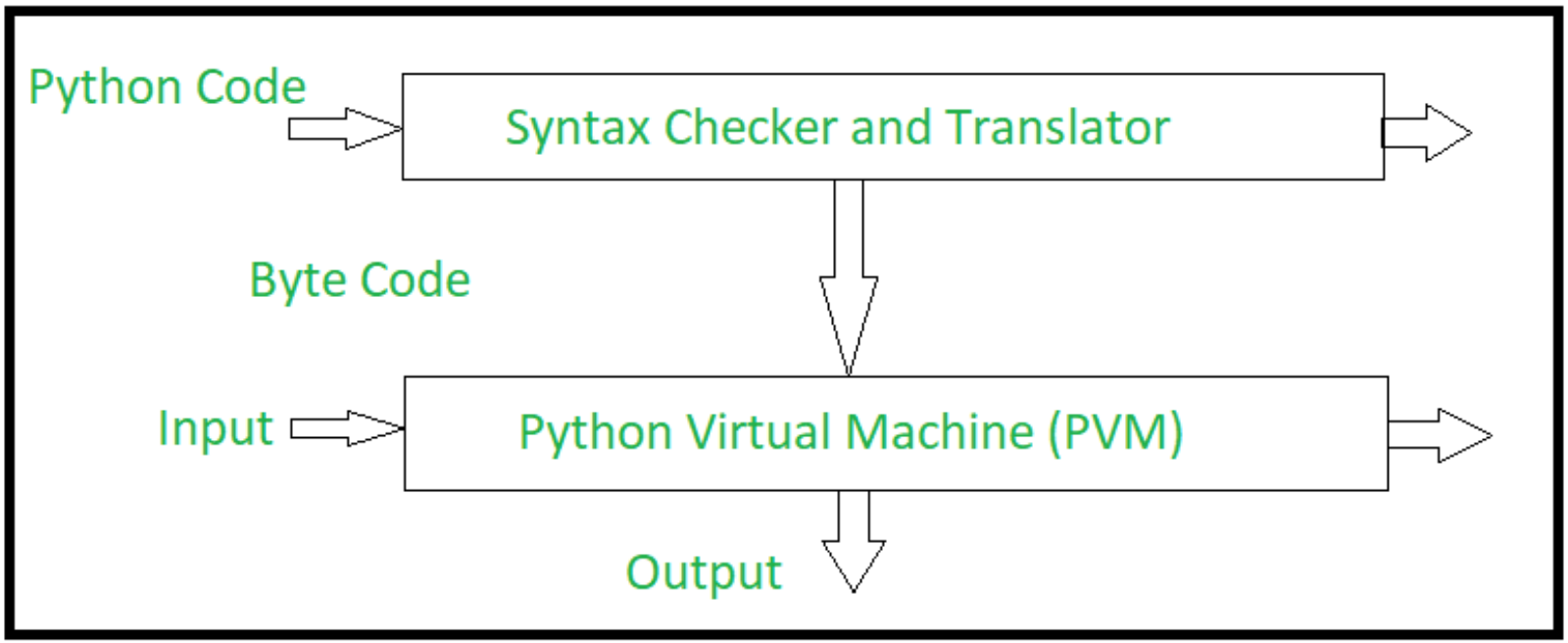
Python is **interpreted**, which means that Python code is translated and executed one statement at a time.

This is different from other languages such as C which are **compiled**.

The Interpreter

Actually, Python is always translated into **byte code**, a lower level representation.

The byte code is then interpreted by the Python Virtual Machine.



Getting Python

To install Python on your personal computer / laptop, you can download it for free at: `www.python.org/downloads`

- There are two major versions: Python 2 and Python 3. Python 3 is newer and *is not backward compatible with Python 2.* **Make sure you're running Python 3.**
- It's available for Windows, Mac OS, Linux.
- If you have a Mac, it *may* already be pre-installed.
- It should already be available on most computers on campus.
- It comes with an editor and user interface called IDLE.

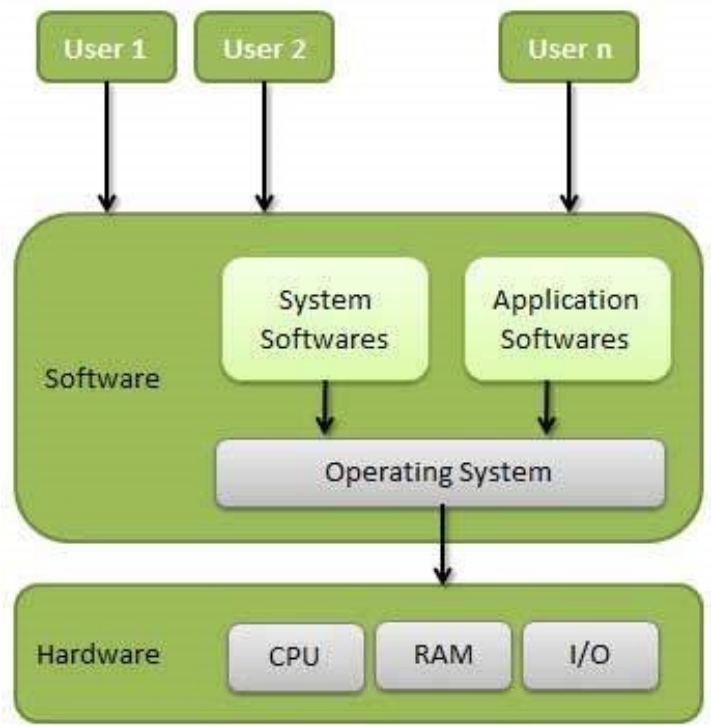
Break

Let's take a break here and resume in the next video.

Operating System

The *operating system* on your computer is the software that allows you to interact with the computer.

E.g., Linux, Windows, MacOS for computers; iOS and Android on mobile devices.



Aside: The Command Line Interpreter

You're probably used to using a *graphical user interface (GUI)*, e.g., clicking on or moving icons on your computer screen.

Your computer also has a more primitive interface that allows you to type commands that are then interpreted and executed by *the operating system*.

This interface may be called:

- the command line interpreter
- the shell
- the terminal

When you see the prompt ">" in these slides, that is the prompt of the command line interpreter. Your prompt may be different. *You can't type Python code there!*

The Python Interpreter: Interactive Mode

There are multiple ways to run a Python program.

Interactive mode: means to start the Python interpreter from the OS and type commands to the interpreter.

```
> python3
Python 3.6.9 (default, Mar 15 2022, 13:55:28)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more
    information.
>>> print("Hello, World")
Hello, World
>>> 2**50
1125899906842624
>>>
```

Notice the prompt ">" for the command line interpreter. We tell the OS to start *the Python interpreter* with the command `python3`. Then you see the Python prompt ">>>". You can now type in Python commands.

Invoking Python

Once the Python application is installed on your computer, it's there for you to run. How do you do that? *It depends on your OS, but we can probably help you figure it out.*

On my (Linux) machine, I just call the command `python3`. Your machine might be different.

In these slides I often use the command `python`. I told Linux to treat that as an "alias" for `python3`. *Be careful!* On your machine, the command `python` may call Python 2!

A Simple Python Program: Interactive Mode

```
> python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more
    information.
>>> print("Hello, World!")
Hello, World!
>>> print("Go Horns Go")
Go Horns Go
>>> print((10.5 + 2 * 3) / 45 - 3.5)
-3.1333333333333333
```

Here you see the prompt (>) for the OS/command loop, and the prompt (>>>) for the Python interpreter command loop. You don't type those; the system does. (Use Cntl-D to get out of the Python loop.)

The Python Interpreter: Batch Mode

Batch mode: means to run a file containing a complete Python program. This is also be called *script mode*.

```
> python3 YourProgramName.py
  <the output will appear here>
>
```

This assumes that:

- you have created a file named `YourProgramName.py` containing a complete Python program;
- that file is in the same directory where you're running Python.

You can use a text editor or an IDE (interactive development environment) to create the file.

A Simple Python Program: Batch Mode

Here's that "same" program as I'd be more likely to write it. Enter the following text using a text editor into a file called, say, `MyFirstProgram.py`.

In file `MyFirstProgram.py`:

```
# Display two messages:
print("Hello, World!")
print("Go Horns Go")

# Evaluate an arithmetic expression:
print((10.5 + 2 * 3) / 45 - 3.5)
```

The lines that begin with "#" are comments, ignored by Python. Blank lines are also ignored.

Running a Program in Batch Mode

After you have created a file containing your program you can run it from *the command line*, i.e., in batch mode.

```
> python3 MyFirstProgram.py
Hello, World!
Go Horns Go
-3.1333333333333333
>
```

This submits the program in file `MyFirstProgram.py` to the Python interpreter to execute.

This is more useful than interactive mode because you have a file containing your program and you can fix errors and resubmit without retyping a bunch of stuff.

Running from an IDE

You can also run your program from within an IDE (interactive development environment) like IDLE.

There are many different suitable IDEs for Python (VSCode, PyDev, PyCharm, Spyder, Thonny, Eclipse, many others).

Use any IDE of your choice, but VSCode is used in CS313E.

Running from an IDE

I personally don't use an IDE.

I develop my Python programs using the Emacs text editor and then run them at the command line.

- It's possible that neither I nor the TAs will be able to help you with IDE-specific issues.
- Some IDEs do things like add code to your program that may mess you up.

I *strongly suggest* that you learn how to run your code from the command line, i.e., without having to rely on any IDE. Then you'll have a backup if your IDE isn't working.

A Simple Python Program: Using a Function

Here's the "same" program as you'll see it written here and in the book. Again it's run in *batch mode*, but using a function `main()`.

In file `MyFirstProgram2.py`:

```
def main():
    # Display two messages:
    print("Hello, World!")
    print("Go Horns Go")

    # Evaluate an arithmetic expression:
    print((10.5 + 2 * 3) / 45 - 3.5)

# Call the function main()
main()
```

This is probably the best approach to get started. Notice that the "body" of the function must be indented, and you have to call the function.

Running that One

What you see when you run the program won't be any different if you put your code in a function:

```
> python3 MyFirstProgram2.py
Hello, World!
Go Horns Go
-3.1333333333333333
>
```

A Simple Python Program: One Last Way

Finally, if you have your Python code in a file, you can access it in interactive mode as follows:

```
>>> import MyFirstProgram
Welcome to Python!
Go Horns Go
-3.1333333333333333
```

The `import` command submits the contents of file `MyFirstProgram.py` to the interpreter and executes any commands found there.

Notice: `MyFirstProgram.py` is a file. From Python's perspective this defines a *module* called `MyFirstProgram` (no .py extension). It's the module you import, not the file.

Aside: About Print

If you do a computation and want to display the result use the `print` function. You can print multiple values with one print statement:

```
>>> print("The value is: ", 2 * 10 )
The value is:  20
>>> print( 3 + 7, 3 - 10 )
10 -7
>>> 3 + 7
10
>>> 3 - 10
-7
>>> 3 + 7, 3 - 10
(10, -7)
```

Notice that if you're computing an expression in interactive mode, *it will display the value without an explicit* `print`. But not in batch mode!

Python will figure out the type of the value and print it appropriately.

Printing and Batch Mode

In batch mode, you need explicit `print` statements to see any results!

In file `flaky.py`:

```
def main():
    3 + 7
    3 - 10
    3 + 7, 3 - 10
    print("What happened to the previous lines?")

main()
```

```
> python3 flaky.py
What happened to the previous lines?
>
```

Break

Let's take a break here and resume in the next video.

The Framework of a Simple Python Program

Define your program in file `Filename.py`:

```
def main ():

    Python statement
    Python statement
    Python statement
        ...
    Python statement
    Python statement
    Python statement

main ()
```

Defining a function called main.

These are the instructions that make up your program. *Indent all of them the same amount (usually 4 spaces).*

This says to execute the function `main`.

To run it:

```
> python Filename.py
```

This submits your program in `YourFilename.py` to the Python interpreter.

Program Documentation

Documentation refers to comments included within a source code file that explain what the code does.

- Include a **file header**: a summary at the beginning of each file explaining what the file contains, what the code does, and what key feature or techniques appear.
- You should always include your name, date, and a brief description of the program.

```
# Joe Student
# CS303E Assignment 1
# August 24, 2024
#
# This program solves the halting problem,
# cures cancer and ensures world peace.
```

Program Documentation

- Comments should also be interspersed in your code:
  - At the start of each function or class definition (i.e., program subdivision);
  - Before each major code block that performs a significant task;
  - Before or next to any line of code that may be hard to understand.

```
sum = 0
# sum the integers [start ... end]
for i in range( start, end + 1):
    sum += i
```

Don't Over Comment

Comments are useful so that you and others can understand your code. Useless comments just clutter things up:

```
x = 1       # assign 1 to x
y = 2       # assign 2 to y
```

Don't do this!

Programming Style

Every language has its own unique *style*. This is a C program.

Good programmers follow certain *conventions* to make programs clear and easy to read, understand, debug, and maintain.

```c
#include <stdio.h>

/* print table of Fahrenheit to Celsius
   [C = 5/9(F-32)] for fahr = 0, 20, ...,
       300 */

main()
{
  int fahr, celsius;
  int lower, upper, step;

  lower = 0;     /* low limit of table */
  upper = 300;   /* high limit of table */
  step = 20;     /* step size */
  fahr = lower;
  while (fahr <= upper) {
    celsius = 5 * (fahr-32) / 9;
    printf("%d\t%d\n", fahr, celsius);
    fahr = fahr + step;
  }
}
```

Programming Style

Some Python programming conventions:

- Follow variable naming conventions.
- Use meaningful variable/function names.
- Document your code.
- Each level indented the same (typically 4 spaces).
- Use blank lines to separate segments of code.

We'll learn more elements of style as we go.

Programming Style

Here's our previous Fahrenheit to Celsius program in Python.

```
def printFahrToCelsius():
    """ Print a table of Fahrenheit to Celsius values:
       [C = 5/9(F-32)] for fahr = 0, 20, ..., 300. """

    lower = 0                    # lower limit of table
    upper = 300                  # upper limit
    step  = 20                   # step size

    # print table header:
    print("Fahr\tCelsius")
    # Compute and print table values.
    for fahr in range( lower, upper + 1, step ):
        celsius = 5 * (fahr - 32) / 9.0
        print( format( fahr, "3d"), "\t", /
               format( celsius, "6.2f") )

printFahrToCelsius()
```

Errors: Syntax

Remember: "Program: *n.* A magic spell cast over a computer allowing it to turn one's input into error messages."

You may encounter three types of *errors* when developing your Python program.

syntax errors: these are ill-formed Python and caught by the interpreter prior to executing your code.

```
>>> 3 = x
  File "<stdin>", line 1
SyntaxError: can't assign to literal
```

These are usually the easiest to find and fix.

Errors: Runtime

runtime errors: you try something illegal while your code is executing

```
>>> x = 0
>>> y = 3
>>> y / x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

The system will detect the error, but only when the program runs.

Errors: Logic

logic errors: your program runs but returns an incorrect result.

```
>>> lst = [1, 2, 3, 4, 5]
>>> prod = 0
>>> for x in lst:
...     prod = prod * x
>>> print (prod)
0
```

This program is syntactically fine and runs without error. But it probably doesn't do what the programmer intended; it always returns 0 no matter what's in `lst`. How would you fix it?

Logic errors are often the hardest errors to find and fix.

Almost Certainly It's Your Fault!

At some point you'll be tempted to say: "My program is obviously right. The interpreter / operating system / world must be incorrect / flaky / hate me."

This software has been used for *millions* of programs before yours.

Try It!

"The only way to learn a new programming language is by writing programs in it." –B. Kernighan and D. Ritchie

Python is wonderfully accessible. If you wonder whether something works or is legal, just try it out.

Programming is not a spectator sport! Write programs!

Next stop: Simple Python.