CS303E: Elements of Computers and Programming

Python

Mike Scott
Department of Computer Science
University of Texas at Austin

Adapted from Dr. Bill Young's Slides

Last updated: May 23, 2023

Some Thoughts about Programming

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

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

"Programming is not a spectator sport." - Bill Young

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.

What is Python?

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

Python has twice received recognition as the language with the largest growth in popularity for the year (2007, 2010).

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

What is Python?

Python is a simple but powerful ***scripting*** language. It has 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.

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
  - programs for exchanging financial transactions at the NY Stock Exchange
  - the grading scripts for this class

What is Python?

Python can be an **object-oriented** programming language. Object-oriented programming is a powerful approach to developing reusable software. More on that later!

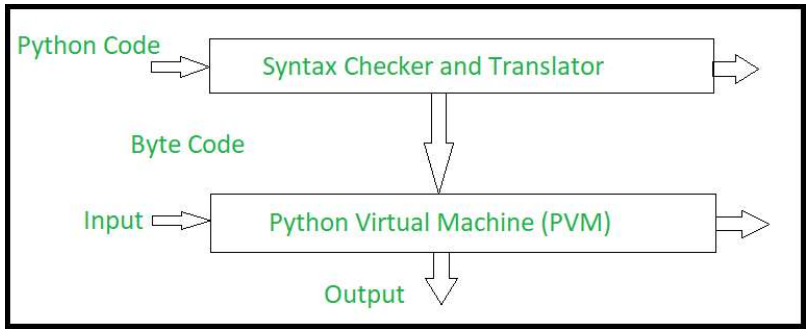
Python is **interpreted**, which means that Python code is translated and executed one statement at a time.

This is different from other languages such as C which are ***compiled*, the code is converted to machine code and then the program can be run after the compilation is finished.**

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 3.8.
- 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.
- I strongly recommend downloading and installing the PyCharm, Educational version, IDE.

A Simple Python Program: Interactive Mode

This illustrates using Python in **interactive mode** from the command line. *Your command to start Python may be different.*

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:0
D64)] on win32
Type "help", "copyright", "credits" or "license()" f
>>> print('Hello World!')
Hello World!
>>> print('Hook \'em Horns!')
Hook 'em Horns!
>>> print((10.5   +   2   *   3)   /   45   -   3.5)
-3.1333333333333333
>>>
```

Here you see the prompt for the OS/command loop for the Python interpreter read, eval, print loop.

A Simple Python Program: Script Mode

Here's the "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. This is called *script mode*.

In file my_first_program.py:

```
def main():
    # Display two messages.
    print('Hello World!')
    print('Hook \'em Horns!')

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


main()
```


A Simple Python Program

```
Hello World!
Hook 'em Horns!
-3.1333333333333333


Process finished with exit code 0
```

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

This is better, because you have a file containing your program and you can fix errors and resubmit without retyping a bunch of stuff.

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.

Python will figure out the type of the value and print it appropriately. This is very handy when learning the basics of computations in Python.

Another aside: Binary Numbers, Base 2 Numbers

- The vast majority of computer systems use digital storage
- Some physical phenomena that is interpreted to be a 0 or 1
  - abstraction, pretending something is different, simpler, than it really is
- also known as binary representations
- 1 bit -> 1 binary digit, a 0 or a 1
- 1 byte -> 8 bits
- binary numbers, base 2 numbers

Base 2 Numbers

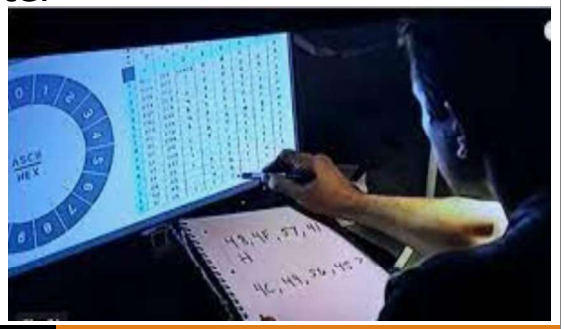
- 5372_{10}
- = (5 * 1,000) + (3 * 100) + (7 * 10) + (2 * 1)
- = $(5 * 10^3) + (3 * 10^2) + (7 * 10^1) + (2 * 10^0)$
- Why do we use base 10? 10 fingers?
- Choice of base is somewhat arbitrary
- In computing we also use base 2, base 8, and base 16 depending on the situation
- In base 10, 10 digits, 0 - 9
- In base 2, 2 digits, 0 and 1

Base 2 Numbers

- 1011011_2
- = (1 * 64) + (0 * 32) + (1 * 16) + (1 * 8) + (0 * 4) + (1 * 2) + (1 * 1) = 91
- = $(1 * 2^6) + (0 * 2^5) + (1 * 2^4) + (1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (1 * 2^0) = 91$
- Negative numbers and real numbers are typically stored in a non-obvious way
- If the computer systems only stores 0s and 1s how do we get digital images, characters, colors, sound, …
- Encoding

Encoding

- Encoding is a system or standard that dictates what "thing" is representing by what number
- Example ASCII or UTF-8
- This number represents this character
- First 128 numbers of ASCII and UTF-8 same
- 32 -> space character
- 65 -> capital A
- 97 -> lower case a
- 48 -> digit 0



Computer Memory

- Recall, 1 bit -> a single 0 or 1
- 1 byte = 8 bits
- A typical laptop or desktop circa 2023
- … has 4 to 32 Gigabytes of RAM, also known as main memory.
  - 1 Gigabyte -> 1 billion bytes
- The programs that are running store their instructions and data (typically) in the RAM
- … have 100s of Gigabytes up to several Terabytes (trillions of bytes) in secondary storage. Long term storage of data, files
- Typically spinning disks or solid state drives.

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 file_name.py
```

This submits your program in file_name.py to the Python interpreter.

Aside: Running Python From a File

Typically, if your program is in file hello.py, you can run your program by typing at the command line:

```
> python hello.py
```

You can also create a *stand alone script*. On a Unix / Linux machine you can create a *script* called hello.py containing the first line below (assuming that's where your Python implementation lives):

```
#!/usr/bin/python3
#   The line above may vary based on your system
print('Hello World!')
```


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 shall always include your name, email, grader, and a brief description of the program.

```
# File: <NAME OF FILE>
# Description: <A DESCRIPTION OF YOUR PROGRAM>
# Assignment Number: <Assignment Number, 1 - 13>
#
# Name: <YOUR NAME>
# EID:  <YOUR EID>
# Email: <YOUR EMAIL>
# Grader: <YOUR GRADER'S NAME Carolyn OR Emma or Ahmad>
#
# On my honor, <YOUR NAME>, this programming assignment is my own work
# and I have not provided this code to any other student.
```

Program Documentation

- Comments shall also be interspersed in your code:
  - Before 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
```


Programming Style

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

Good programmers follow certain *conventions* to make programs clear and easy to read, understand, debug, and maintain. We have conventions in 303e. Check the assignment page.

```
#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 **important** Python programming conventions:

- Follow variable and function naming conventions.
- Use meaningful variable/function names.
- Document your code **effectively**.
- Each level indented the same (4 spaces).
- Use blank lines to separate segments of code inside functions.
- 2 blank lines before the first line of function (the function header) and after the last line of code of the function

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

Check the assignments page for more details.

Errors:

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

We will encounter three types of *errors* when developing our 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 typically 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
```


Almost Certainly It's Our Fault!

At some point we all say: "My program is obviously right. The interpreter / Python must be incorrect / flaky / and it hates me."

"As soon as we started programming, we found out to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."

-Sir Maurice V Wilkes

Errors: Logic

logic errors: Calculate 6! (6 * 5 * 4 * 3 * 2 * 1)
your program runs but returns an incorrect result.

```
>>> prod = 0
>>> for x in range(1, 6):
...     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 the values in range. How would you fix it?

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

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! Do exercises!