Introduction to Programming in Python

Introduction to Python

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

Last updated: June 4, 2021 at 11:04

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

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 simple but powerful. 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 quickly.

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?

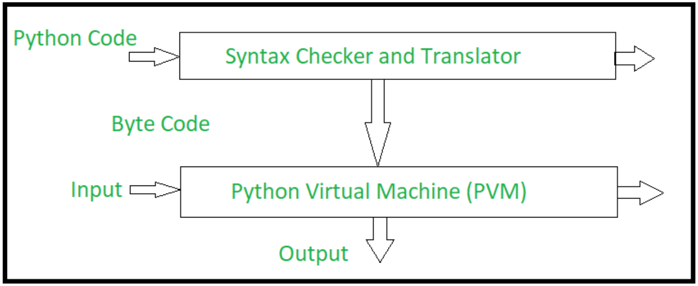
Python is an **object-oriented** programming language.
Object-oriented programming is a powerful approach to developing reusable software. More on that later!

The Interpreter

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

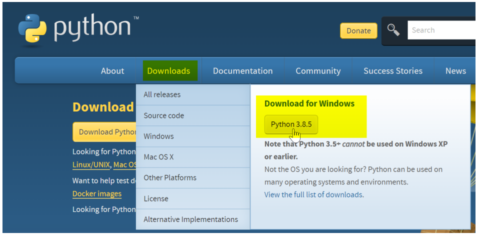
Actually, Python is always translated into **byte code**, a lower level representation.

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`



Note that Python 3.5+ *cannot* be used on Windows XP or earlier.
Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of 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 comes with an editor and user interface called IDLE.

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.*

```
> python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help" 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 for the Python interpreter command 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 `MyFirstProgram.py`:

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

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

You can use IDLE as your text editor. There are other IDEs that are more powerful, e.g., PyCharm.

A Simple Python Program

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

A Simple Python Program: Another Way

Here's the "same" program as you might see it written, again in *script mode*, but using a function. **This is the preferred mode!**

```
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()
```

You'll see this style in many examples here.

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`. In Script mode it won't; you'll have to call `print`.

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

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
# Texas Summer Discovery: Python
# August 24, 2020
#
# This program solves the halting problem,
# cures cancer and ensures world peace.
```

Program Documentation

- Also include comments 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:

Don't do this!

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

Programming Style

Every language has its own unique *style*. Good programmers follow certain *conventions* to make programs clear and easy to read, understand, debug, and maintain.

Some Python programming 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.

Errors: Syntax

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

syntax errors: these are illegal Python and caught by the interpreter before 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
```

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.

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!

Drawing: Homework 1

One way to draw with Python: draw with your text editor.

```
    ^___^
   ( 0,0 )
   /)____)
    ,, ,,
```

Then just put print statements around each line:

In file `Owl.py`:

```
def main():
    print("    ^___^  ")
    print("   ( 0,0 ) ")
    print("   /)____) ")
    print("    ,, ,, ")

main()
```

```
> python Owl.py
    ^___^
   ( 0,0 )
   /)____)
    ,, ,,
>
```