



PYTHON PROGRAMMING FOR EVERYONE

---

## 2.1.1 WHY PROGRAM?

# 1. WHAT IS A PROGRAM?

- ▶ It is the language that tells the computer what to do.
- ▶ It is a way to automate the boring stuff.
- ▶ It can be one line long, or more than a million lines long.
- ▶ It is the solution to a problem.
- ▶ Constructs *may* include: input, output, sequential execution, conditional execution, repeated execution, or reuse.
- ▶ It is a sequence of instructions that specifies how to perform a computation.

**IF I HAD 60 MINUTES TO SOLVE A PROBLEM, I'D SPEND 55 MINUTES DEFINING IT, AND 5 MINUTES SOLVING IT.**

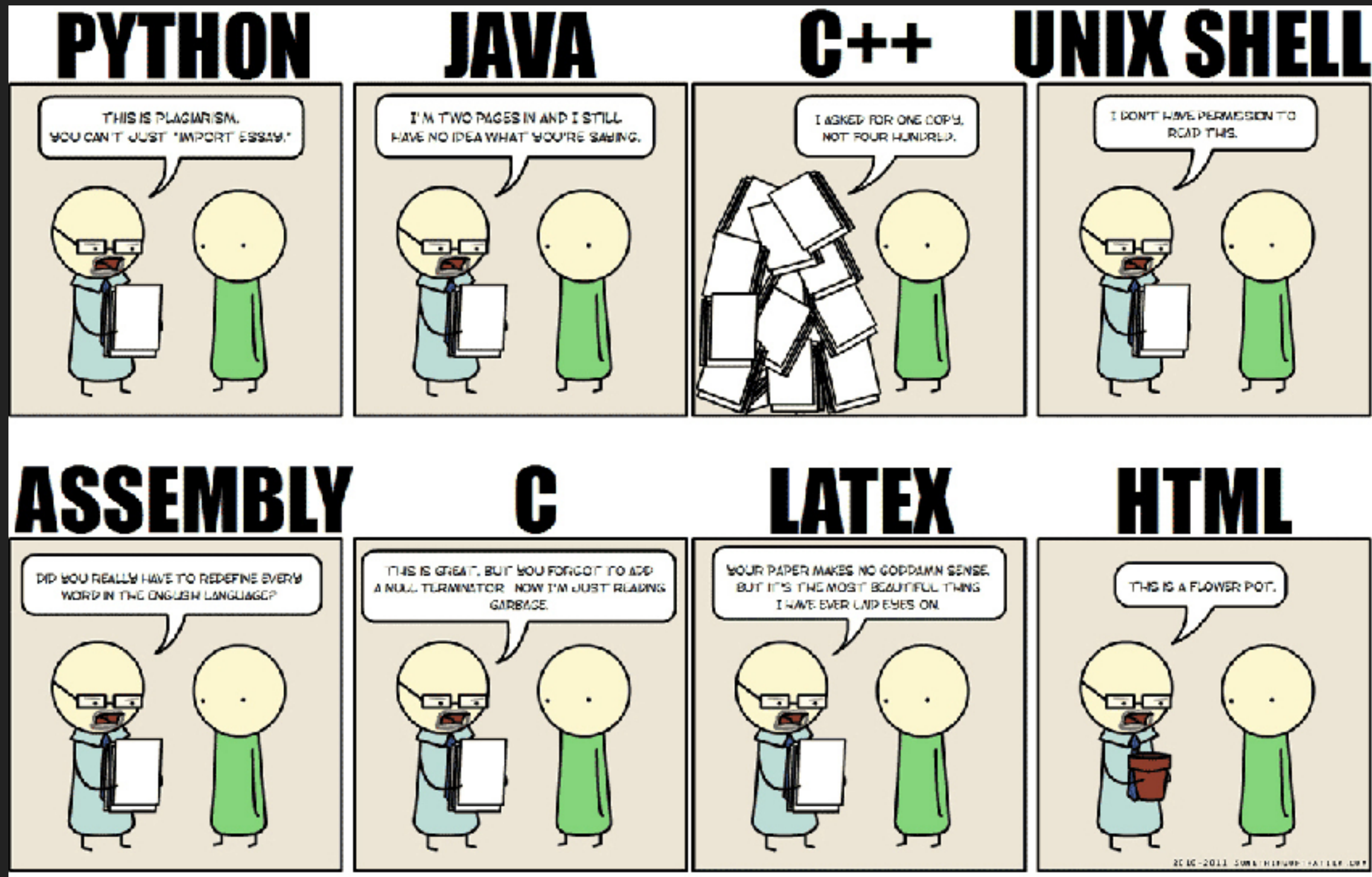
**Albert Einstein**

## 2. PROBLEM SOLVING

- ▶ The most important skill to master when programming – or *thinking like a computer scientist* – is problem solving.
- ▶ At the most basic level, problem solving is the process of formulating a problem, finding a solution, and expressing that solution.
- ▶ In reality, problem solving includes being able to:
  - ▶ Identify what the problem is and define it in detail.
  - ▶ Break the overall, large problem into smaller pieces.
  - ▶ Develop solutions for each of the small pieces. and express those solutions in-depth.
  - ▶ Take all the small solutions and put them back together to solve the larger, overall problem.

## 3. VOCABULARY & LANGUAGES

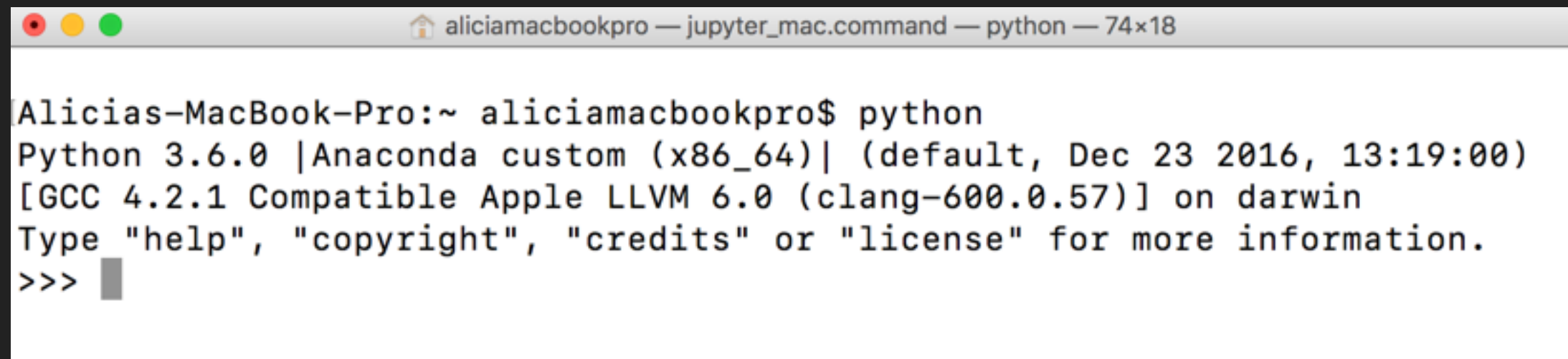
- ▶ Natural Languages = those that people speak, such as English.
- ▶ Formal Languages = those designed BY people for specific purposes, like Python.
- ▶ Formal Languages, like Python, have a strict set of syntax rules that cannot be ambiguous, and must be literal.
  - ▶ A computer doesn't have the ability to interpret more than one meaning from a single statement in Python.
- ▶ Formal Languages have 'reserved words' which mean something very specific to the computer. E.g., while, and, not, or, if, else, etc.



## 4. PYTHON ON THE COMMAND LINE

Step 1: Open your Terminal or Command Line tool.

Step 2: Type “python” on the command line.



```
aliciamacbookpro — jupyter_mac.command — python — 74x18
Alicias-MacBook-Pro:~ aliciamacbookpro$ python
Python 3.6.0 |Anaconda custom (x86_64)| (default, Dec 23 2016, 13:19:00)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Your command line has gone from **Computer-Name:~ Command-Line-Name\$** to **>>>**.

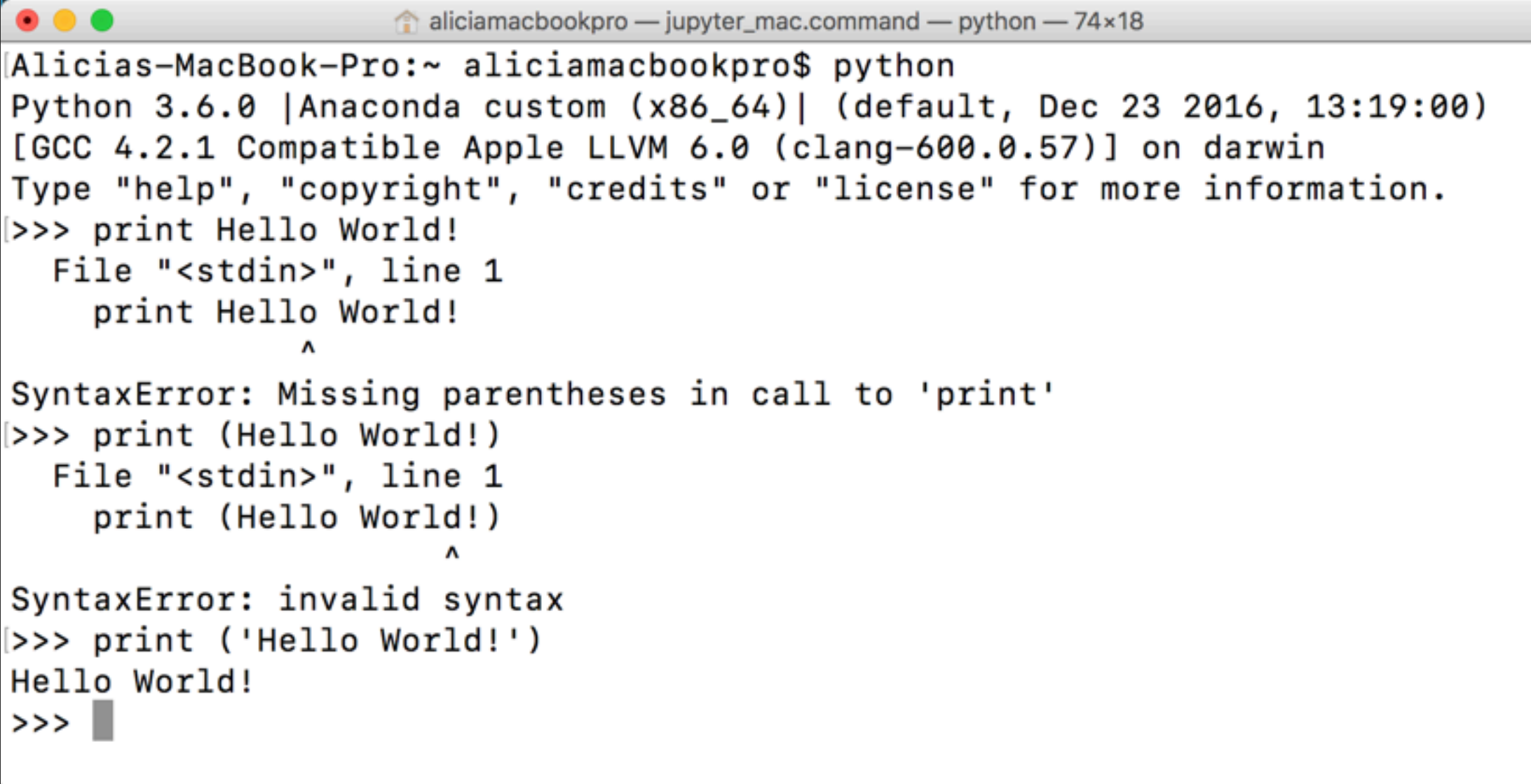
You are now inside the **Python Interpreter**, which is interactive. You can type Python instructions (aka. code) into the interpreter and it will automatically translate those instructions to machine language, send it to the CPU to be executed, then translate the result back to “English,” and display the result on the screen.



## 5. HELLO WORLD!

Step 3: Enter an instruction for Python to execute.

Here's where syntax comes into play. Python is ONLY going to understand what to do if you use the exact syntax. Any other syntax will result in an error message – a 'syntax' error message.

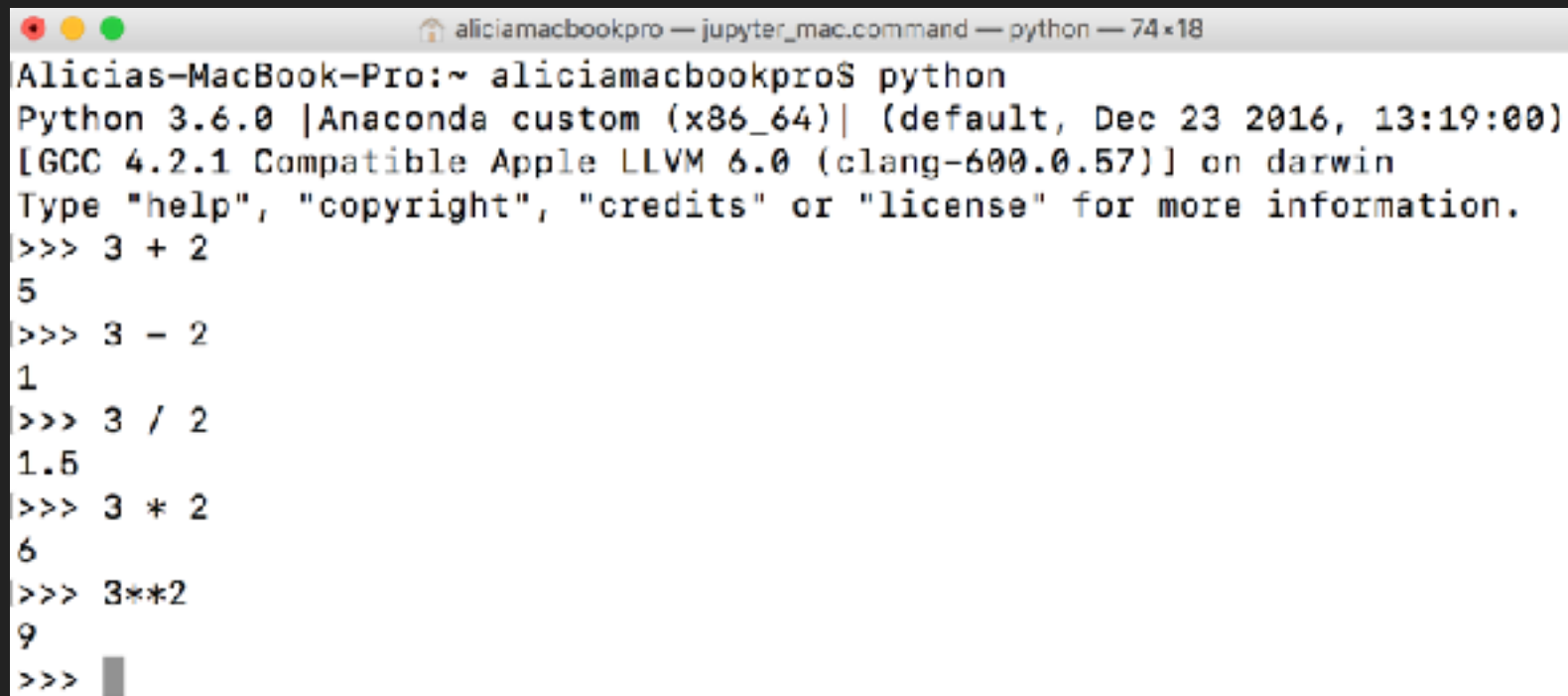


```
aliciamacbookpro — jupyter_mac.command — python — 74x18
[Alicias-MacBook-Pro:~ aliciamacbookpro$ python
Python 3.6.0 |Anaconda custom (x86_64)| (default, Dec 23 2016, 13:19:00)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print Hello World!
  File "<stdin>", line 1
    print Hello World!
        ^
SyntaxError: Missing parentheses in call to 'print'
[>>> print (Hello World!)
  File "<stdin>", line 1
    print (Hello World!)
        ^
SyntaxError: invalid syntax
[>>> print ('Hello World!')
Hello World!
>>> █
```



## 6. ARITHMETIC

- ▶ Python works a lot like a calculator as well.
- ▶ You can use **operators** to do arithmetic in Python.
- ▶ Operators include: + (addition), - (subtraction), \* (multiplication), / (division), and \*\* (exponentiation).

A screenshot of a terminal window on a Mac. The title bar shows 'aliciamacbookpro — jupyter\_mac.command — python — 74x18'. The terminal text shows the user running 'python' in the shell, which starts the Python 3.6.0 interpreter. The interpreter displays version and environment information. The user then enters several arithmetic expressions at the prompt '>>>' and receives the results: '3 + 2' returns 5, '3 - 2' returns 1, '3 / 2' returns 1.5, '3 \* 2' returns 6, and '3\*\*2' returns 9. The prompt '>>>' is shown again at the bottom.

```
Alicias-MacBook-Pro:~ aliciamacbookpro$ python
Python 3.6.0 |Anaconda custom (x86_64)| (default, Dec 23 2016, 13:19:00)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 3 + 2
5
>>> 3 - 2
1
>>> 3 / 2
1.5
>>> 3 * 2
6
>>> 3**2
9
>>> 
```

## 7. ERRORS & DEBUGGING

- ▶ You will ALWAYS make a mistake at some point. Sometimes that mistake – or **bug** – is easy to fix, other times it might take awhile. Regardless, don't take it personally!
- ▶ There's some interesting explanations as to why a bug is called a bug – check it out on [Wikipedia](#)!
- ▶ Fixing bugs in a program is called **debugging**.
- ▶ Bugs, or errors, come in three flavours: Syntax, Logic/Runtime, or Semantic.
- ▶ **TIP:** When you're first starting out, make mistakes on purpose. See what error messages are created. Learn from these mistakes.