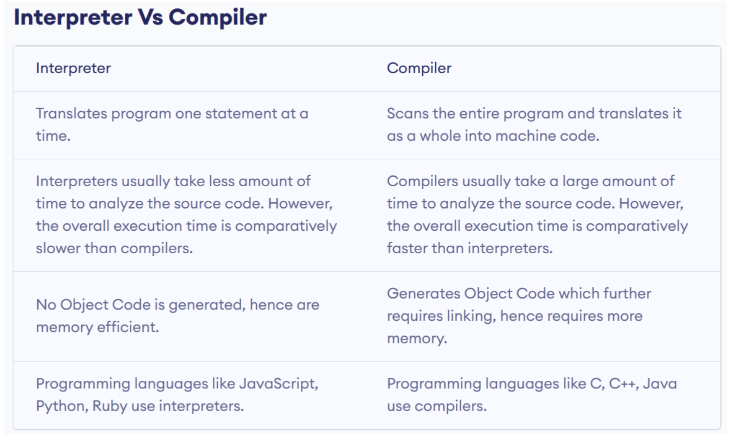

Interpreter Vs Compiler

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers.	Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters.
No Object Code is generated, hence are memory efficient.	Generates Object Code which further requires linking, hence requires more memory.
Programming languages like JavaScript, Python, Ruby use interpreters.	Programming languages like C, C++, Java use compilers.

Python Tutorial

Peiyan (Vince) Gong, Abhinav Kumar
9/12/18, updated 9/3/23

Installing Python:
If you use Ubuntu 20.04, you already have the version of Python (**3.8**) that we will be using.
[Download Python | Python.org]
This will work across different operating systems, but Linux is preferable.

Python是一种解释型语言，这意味着你可以在不需要单独的编译步骤的情况下编写和执行Python代码。Python解释器会在运行时逐行处理代码，按照遇到的顺序执行每一行。这与JavaScript相似，它也是一种解释型语言。

What is Python?

1. Python is **Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to JavaScript.
   a. Difference between interpreter and compiler: https://www.programiz.com/article/difference-compiler-interpreter
2. Python is **Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
3. Python is **Object-Oriented** − Python supports Object-Oriented programming that encapsulates code within objects.
4. Python is a **Beginner's Language** − Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

相比之下，编译型语言（如C或C++）需要单独的编译步骤，其中源代码在执行之前被翻译成机器代码。这可以带来更快的执行速度，但在管理和分发编译后的二进制文件方面可能需要更多的工作。

Basics:
These are some basics before we get into the deeper tutorial.
Indentation: 缩进
Python makes heavy use of indents to organize code. Code that is not properly indented will not run.

Right way:
```
if True:
   print "True"
else:
   print "False"
```

Wrong way:

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Comments:
# This is a comment

Multi-line comments:

```
"""
Triple quotes can be used to write comments across multiple lines.
They can also be used to write documentation in code.
"""

def function(argument):
        """Function that prints the argument"
        print(argument)
```

Tutorial:

CS 231n is a computer science course taught at Stanford. They have published their material online, including a Python tutorial.

https://cs231n.github.io/python-numpy-tutorial/

The tutorial includes a Colab notebook, which is a way to run Python code in your browser. We will be using Python 3.8 in this course, but the differences between Python 3.6.9 (the version in the tutorial) and Python 3.8 are minimal.

Installing packages:

Python includes a package manager called `pip` that is used to install 3rd party libraries. Some popular packages we will be using include numpy and matplotlib. To install packages, run:

```
pip install <package>
```

Or

```
pip3 install <package>
```

Which command you use will depend on your operating system.
You can install multiple packages at once:

```
pip install numpy matplotlib
```

Tips for Debugging and Programming:

1. Think carefully and plan things out before coding. (flow charts, pseudo code, simple graphs)
2. Use comments to track what you are doing: When writing a function, write comments about what you want the function to do. When writing an equation, write comments on what you want the equation to solve. When naming a parameter, write comments on what the parameter represents.
3. Try to learn a good coding style: Make your code readable: Name your variables and functions with reason. Use width, height, length, weight, etc instead of x,y,z,a,b,c. Good code should have a short and clear main function with just a few lines.
4. The devil is in the details! Be careful with typos, mis-sized matrices. Check everything and print everything.
5. If your code is not running or doing something weird, just google it.
6. No one can get their code working in one shot, be patient.
7. **Start early! Start early! Start early!**