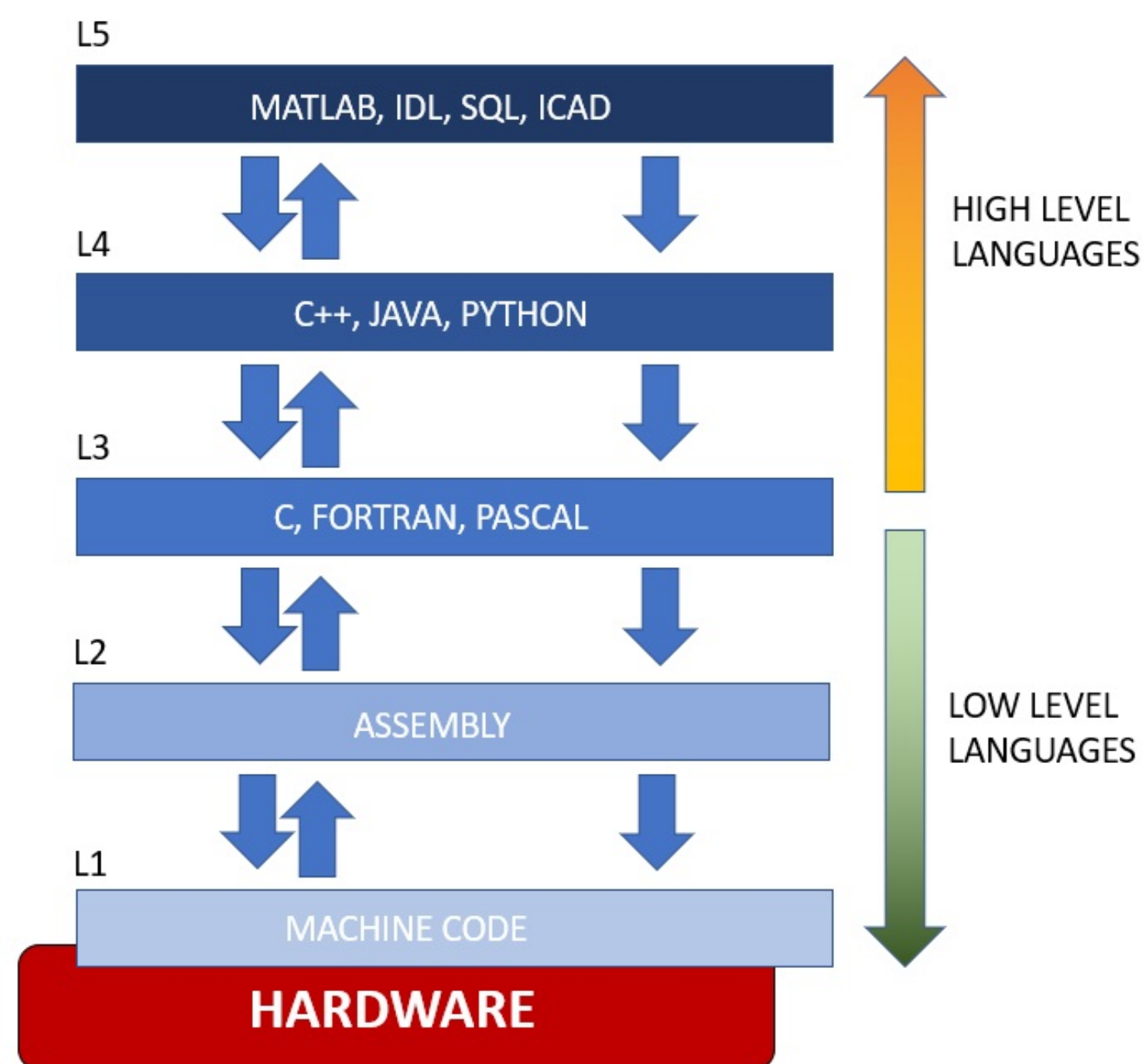


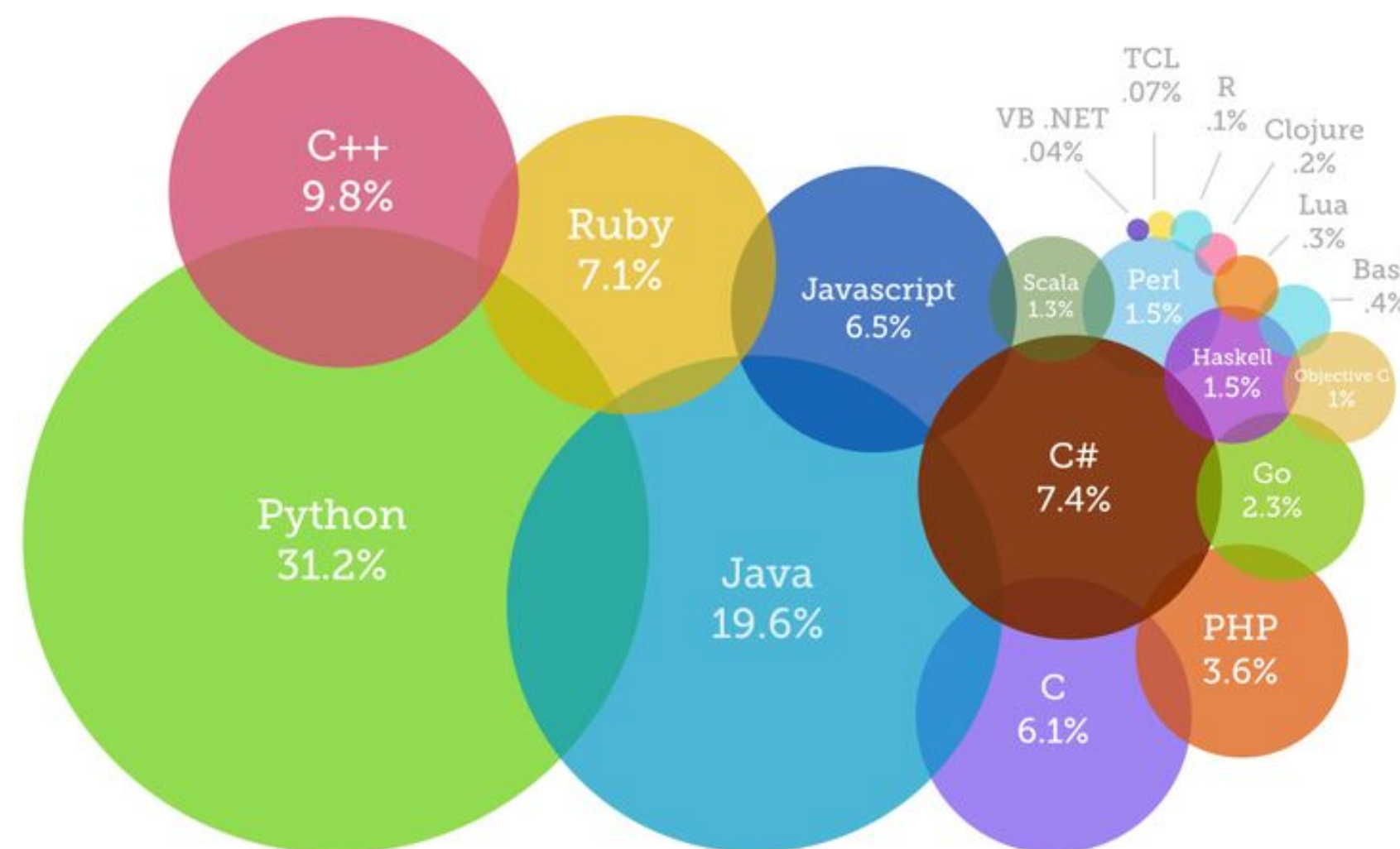
Python

A Soft Start

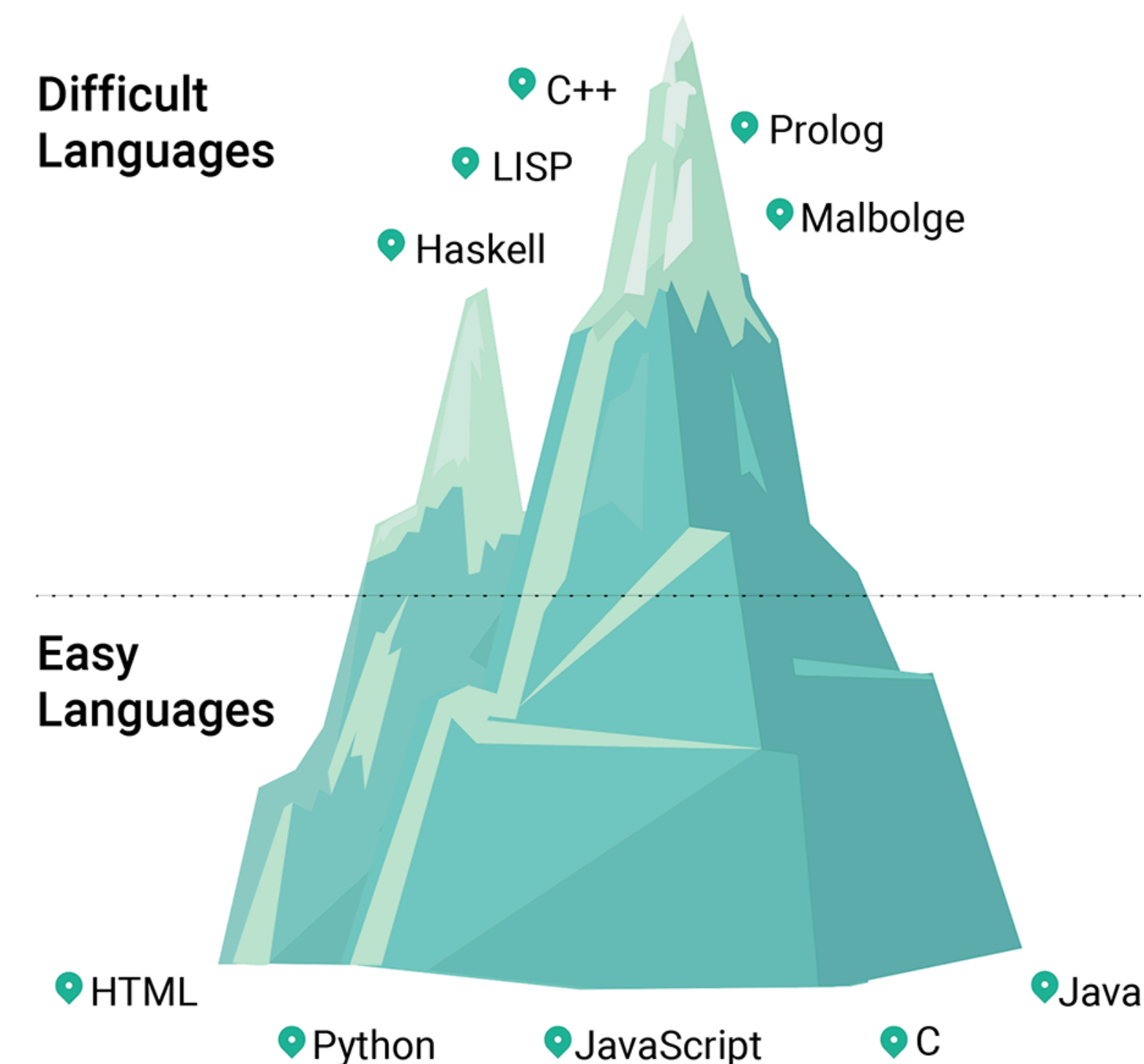
Everything We Said About Python So Far



High level language

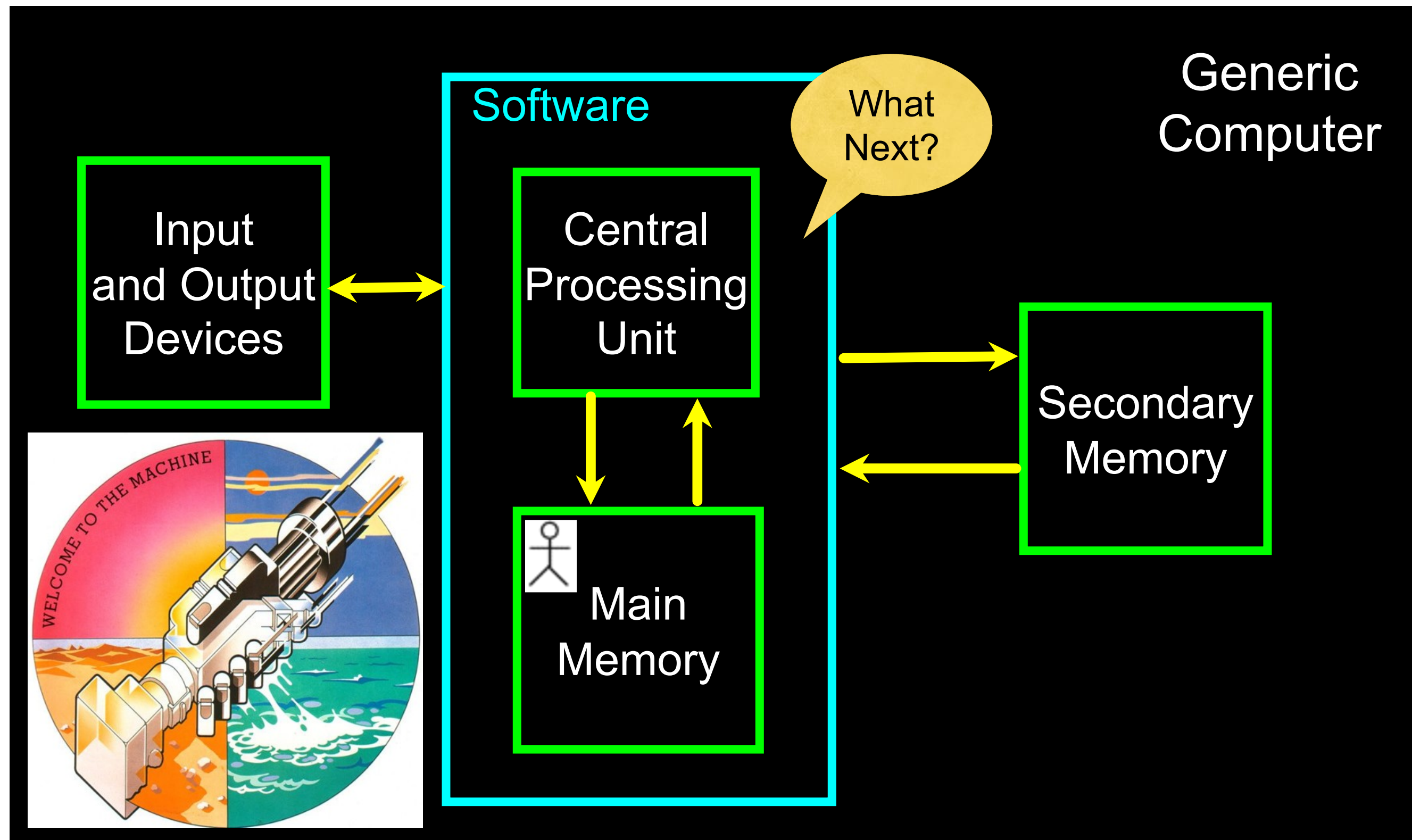


Popular language
[Google web search systems, Youtube, Pixar, Civilization IV, Netflix, iRobot]



Easy language

What We Said About Learning to Program



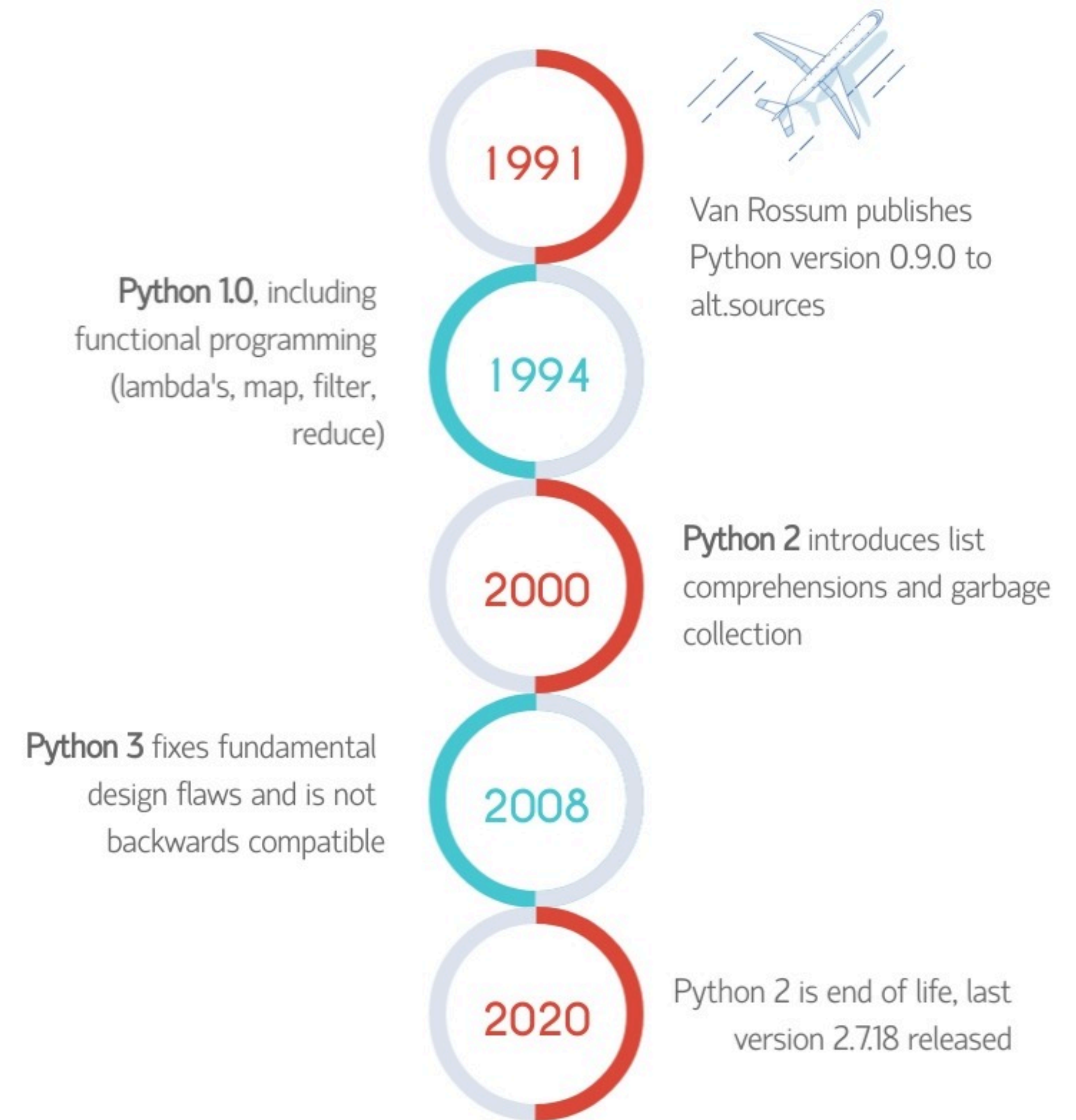
- Computers are built for one purpose: to **do things for us** (e.g., physics calculations)
- CPUs are dumb but fast
- Use **programming languages** to write the **sequence of stored instructions** (code/program/software) that are necessary for the computer do what you need it to do
- By programming, you place **pieces of your intelligence** in the machine

Learning languages requires you to try them out: you will be hitting failures and debugging over and over...
So practice, practice, practice!

History of Python



- Conceived by **Guido van Rossum** (Centrum voor Wiskunde en Informatica, Netherlands)
- Successor to **ABC** programming language, which was inspired by **SETL** (based on mathematical set theory)
- Van Rossum was sole responsible for the project, as lead developer, from 1989 to 12 July 2018, when he announced his “permanent vacation” from his responsibilities as Python's “Benevolent Dictator For Life”
- In January 2019, active Python core developers elected a five-member “Steering Council” to lead the project
- And yes, it is named after Monty Python (whose sketch is also at the origin of the modern usage of the term “spam”)



Philosophy of Python

The Zen of Python

The Zen of Python, by Tim Peters

```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *right* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!
```

```
$ python3  
Python 3.10.7 (main, Sep 14 2022, 22:38:32)  
[Clang 13.0.0 (clang-1300.0.29.30)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import this
```

Python as a Language

- **Parseltongue** was the language of serpents and those who can converse with them [<https://harrypotter.fandom.com/wiki/Parseltongue>]
 - An individual who could speak Parseltongue was known as a **Parselmouth**
 - It was a very uncommon skill, and was known to be an almost exclusively hereditary trait
 - Nearly all known Parselmouths were descended from **Salazar Slytherin**
- **Python** is the language of the Python Interpreter and those who can converse with it
 - An individual who can speak Python is known as a **Pythonista**
 - It is not that uncommon, and should and can be acquired
 - Nearly all known Pythonistas use software initially developed by **Guido van Rossum**

What is Python?

- General purpose language blending procedural, functional, and **object-oriented** programming paradigms
 - In Python all entities (even functions) are objects
- Very high-level language
 - Very different from C/C++, with a simpler syntax
 - Logical structure of program determined by indentation
- Interpreted language: **compilation at runtime**
 - No separation in header files, source file, libraries, executables

General Features

- Dynamic typing
 - No need for type and size declaration: very different from C/C++
 - Objects are tracked at runtime by Python
- Automatic memory management: no need for `new` and `delete`
- Built-in object types: lists, dictionaries, strings, files are **native objects of the language**
- Powerful tools for file management and manipulation: achieve shell scripting inside programs
- Large collection of tools and libraries: included by default or available online

General Features

- Dynamic typing
 - No need for type and size declaration: very different from C/C++
 - Objects are tracked at runtime by Python
- Automatic memory management: no need for `new` and `delete`
- Built-in object types: lists, dictionaries, strings, files are **native objects of the language**
- Powerful tools for file management and manipulation: achieve shell scripting inside programs
- Large collection of tools and libraries: included by default or available online
- External packages are easy to install and manage with `pip` or `conda`

Before We Start

- The days of Python2 are over (because of security flaws, among other things)
- Make sure you are using Python3
 - Ubuntu does not allow `python`, only `python3`

```
$ python --version
Python 2.7.16

$ python3 --version
Python 3.10.7
```

Python

Let's Actually Start

Talking to the Python Interpreter

```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

I am waiting for the next instruction

Talking to the Python Interpreter

```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world only requires one line!")
Hello world only requires one line!
```

Talking to the Python Interpreter


```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world only requires one line!")
Hello world only requires one line!
>>> a = 3.2
>>> b = 4.23
>>> c = 3
>>> x = a*(b/c)
>>> x
4.5120000000000005
```

Talking to the Python Interpreter

```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world only requires one line!")
Hello world only requires one line!
>>> a = 3.2
>>> b = 4.23
>>> c = 3
>>> x = a*(b/c)
>>> x
4.5120000000000005
>>> print(x)
4.5120000000000005
```

Talking to the Python Interpreter

```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world only requires one line!")
Hello world only requires one line!
>>> a = 3.2
>>> b = 4.23
>>> c = 3
>>> x = a*(b/c)
>>> x
4.5120000000000005
>>> print(x)
4.5120000000000005
>>> print("Wow! " + "What a life!")
Wow! What a life!
```



Talking to the Python Interpreter

```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world only requires one line!")
Hello world only requires one line!
>>> a = 3.2
>>> b = 4.23
>>> c = 3
>>> x = a*(b/c)
>>> x
4.5120000000000005
>>> print(x)
4.5120000000000005
>>> print("Wow! " + "What a life!")
Wow! What a life!
>>> print(a,b,c,x)
3.2 4.23 3 4.5120000000000005
>>> print("a = " + str(a))
a = 3.2
```

Strings are easy to handle

Print to screen is extremely flexible

Talking to the Python Interpreter

```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world only requires one line!")
Hello world only requires one line!
>>> a = 3.2
>>> b = 4.23
>>> c = 3
>>> x = a*(b/c)
>>> x
4.5120000000000005
>>> print(x)
4.5120000000000005
>>> print("Wow! " + "What a life!")
Wow! What a life!
>>> print(a,b,c,x)
3.2 4.23 3 4.5120000000000005
>>> print("a = " + str(a))
a = 3.2
>>> print('a = {2}, b = {0}, c = {1}'.format(a, b, c))
a = 3, b = 3.2, c = 4.23
>>>
```

Strings are easy to handle

Print to screen is extremely flexible

Calling the method format on a string: the number refers to position within object passed

Talking to the Python Interpreter

```
$ python3
Python 3.10.7 (main, Sep 14 2022, 22:38:32) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world only requires one line!")
Hello world only requires one line!
>>> a = 3.2
>>> b = 4.23
>>> c = 3
>>> x = a*(b/c)
>>> x
4.5120000000000005
>>> print(x)
4.5120000000000005
>>> print("Wow! " + "What a life!")
Wow! What a life!
>>> print(a,b,c,x)
3.2 4.23 3 4.5120000000000005
>>> print("a = " + str(a))
a = 3.2
>>> print('a = {2}, b = {0}, c = {1}'.format(a, b, c))
a = 3, b = 3.2, c = 4.23
>>>
```

Strings are easy to handle

Print to screen is extremely flexible

Calling the method `format` on a string: the number refers to position within object passed

To exit an interactive session, type `quit()` or press `Ctrl-D`

Interactive vs Script

- Interactive
 - Type directly to Python one line at a time and it responds
 - Good for experiments and programs of 3-4 lines long
 - Most programs are much longer...
- **Script**
 - Enter a sequence of statements (lines) into a file using a text editor and tell Python to execute the statements in the file
 - In a sense, we are “giving the Python interpreter a script to digest”
 - Convention to add “.py” as the suffix on the end of these files

Running a Python Script

Based on `examples/Python/example01.py`

```
# A first Python program: no preamble, no main, no types
```

```
a = 2.3
```

```
b = 4.5
```

```
c = a/b
```

Use # for comments

```
# Plain print
```

```
print(a,b,c)
```

```
# Plain print of more variables
```

```
print('c = a/b = ', c)
```

```
# printf style works, but Python has the powerful
```

```
# String format() method!
```

```
print('a = %.3f, b = %.3g, c = %2.4g' % (a,b,c))
```

```
# Print using ""
```

```
print("a = {0}, b = {1}, c = {2}".format(a, b, c))
```

```
# Print using ''
```

```
print('a = {0}, b = {1}, c = {2}'.format(a, b, c))
```

```
# Python style formatted output: {position:format}
```

```
# format syntax: minimum width . significant digits (for g), digits after the point (for f)
```

```
print('a = {1:.3f}, b = {2:.3g}, c = {0:2.4g}'.format(a, b, c))
```

```
# Woah... internal variable names on the fly without separate declarations
```

```
print('value = {v:.3f}, error = {err:.3g}, # measurements = {N:3d}'.format(N=1000, v=-1.23454335, err=0.1))
```

```
$ python3 example01.py
```

```
2.3 4.5 0.5111111111111111
```

```
c = a/b = 0.5111111111111111
```

```
a = 2.300, b = 4.5, c = 0.5111
```

```
a = 2.3, b = 4.5, c = 0.5111111111111111
```

```
a = 2.3, b = 4.5, c = 0.5111111111111111
```

```
a = 4.500, b = 0.511, c = 2.3
```

```
value = -1.235, error = 0.1, # measurements = 1000
```

Interactive Python with Jupyter

<https://jupyter-notebook.readthedocs.io/en/stable/notebook.html>

- We will also use Jupyter notebooks for interactive python during lectures and for projects
 - .ipynb extension
- Running offline
 - The virtual box has Jupyter installed
 - For MacOS or linux distributions follow <https://jupyter.org/install.html>
- Running online: <https://colab.research.google.com>
 - Classroom takes you to Colab directly if you click on a notebook
 - Otherwise, download it and then run

```
$ jupyter notebook examples/Python/example02.ipynb
```

