Python

Session 1 :
What’s going on there?

Abolfazl Madani

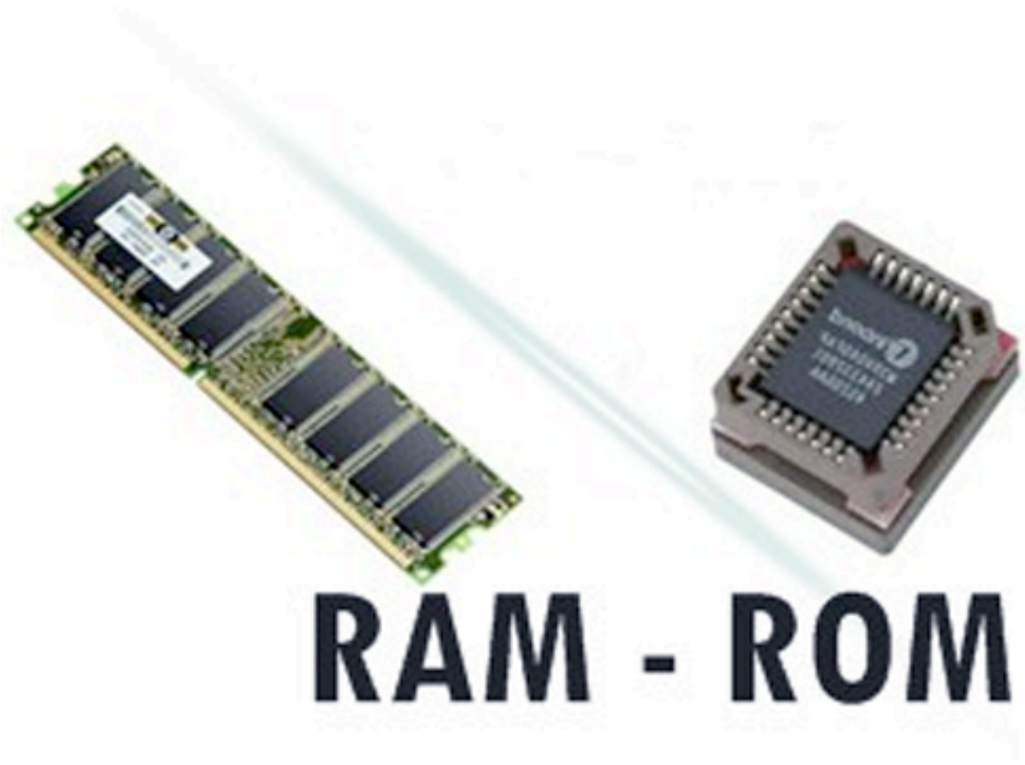
But, not really complicated!

- Follow some rules :
  - The result is not the only matter
  - Different steps need to look forward
  - **Syntax**

Case and You

Ram and ROM

- ROM (read-only memory)
- RAM (random-access memory)
    - chip: ROM can hold data without power and RAM cannot. Essentially, ROM is meant for permanent storage, and RAM is for temporary storage.



Welcome to 0 1 World

A computer’s memory is divided into tiny storage locations known as bytes

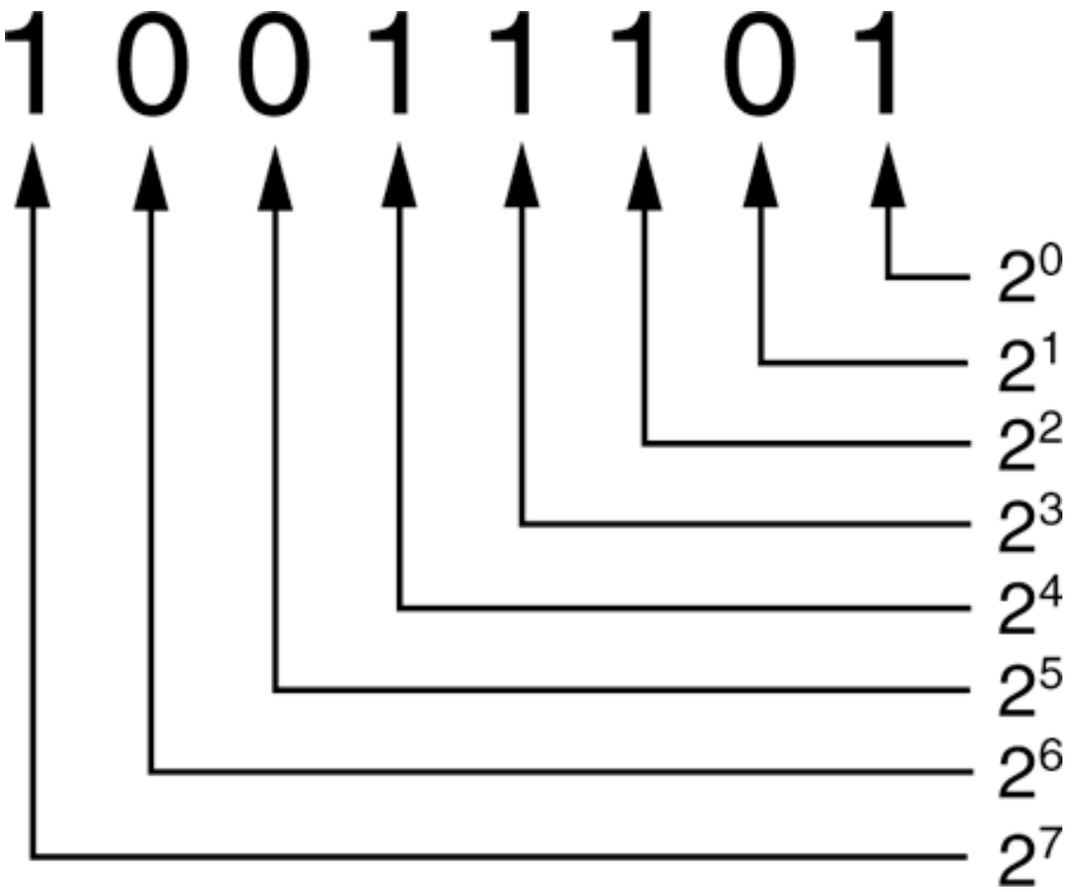
One byte represents one number

A byte is divided into eight smaller storage locations known as bits (binary digits)

Bits are tiny electrical components that can hold either a positive or a negative charge.

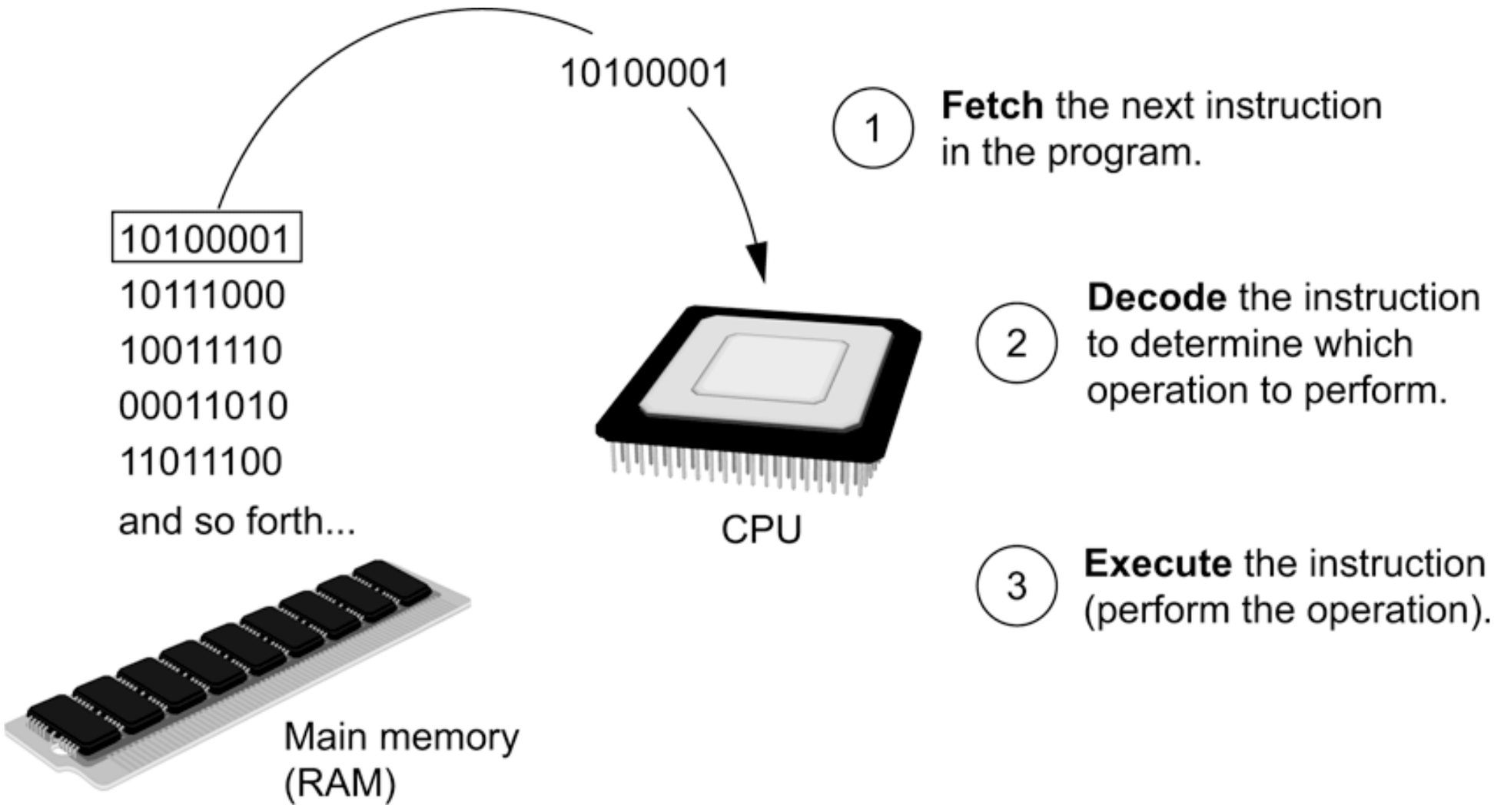
A positive charge is similar to a switch in the on position

A negative charge is similar to a switch in the off position



Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII



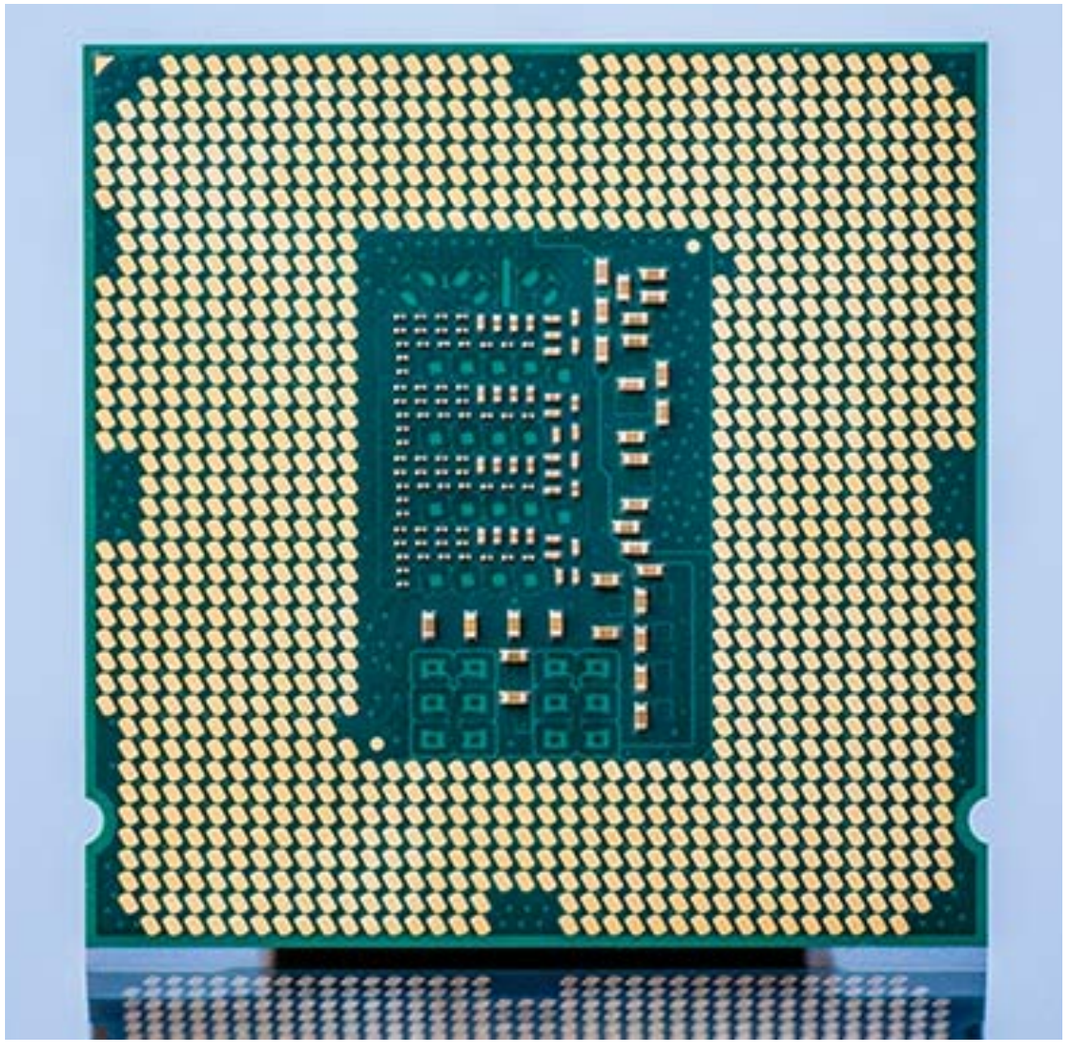
Upper Level

Fetch-decode-execute cycle is the term used when the CPU executes the instructions in a program.
The cycle consist of three steps:
Fetch
Decode

CPU

- The **CPU** (central processing unit) of a computer is often called the “brain” of a computer. It’s a collection of millions of transistors that can be manipulated to perform an awesome variety of calculations. A standard CPU has between one and four processing cores clocked anywhere from 1 to 4 GHz.

- A CPU is powerful because it can do everything. If a computer is capable of accomplishing a task, it’s because the CPU can do it.

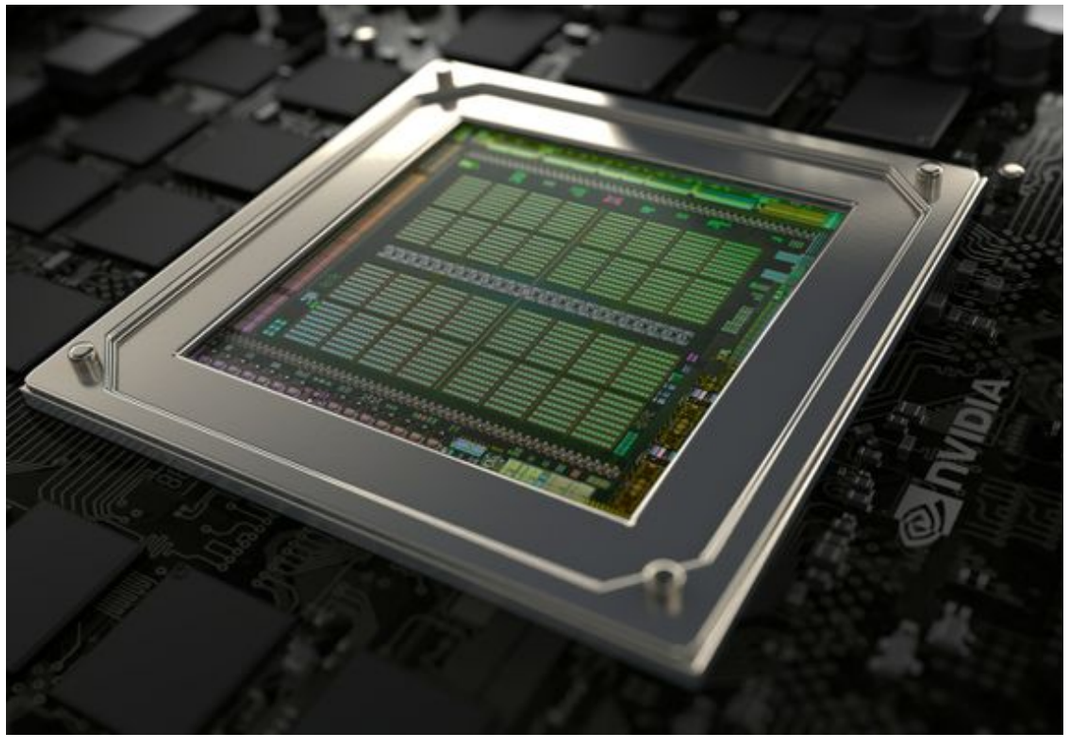


GPU

- A GPU (graphics processing unit) is a specialized type of microprocessor. It’s optimized to display graphics and do very specific computational tasks. It runs at a lower clock speed than a CPU but has many times the number of processing cores.

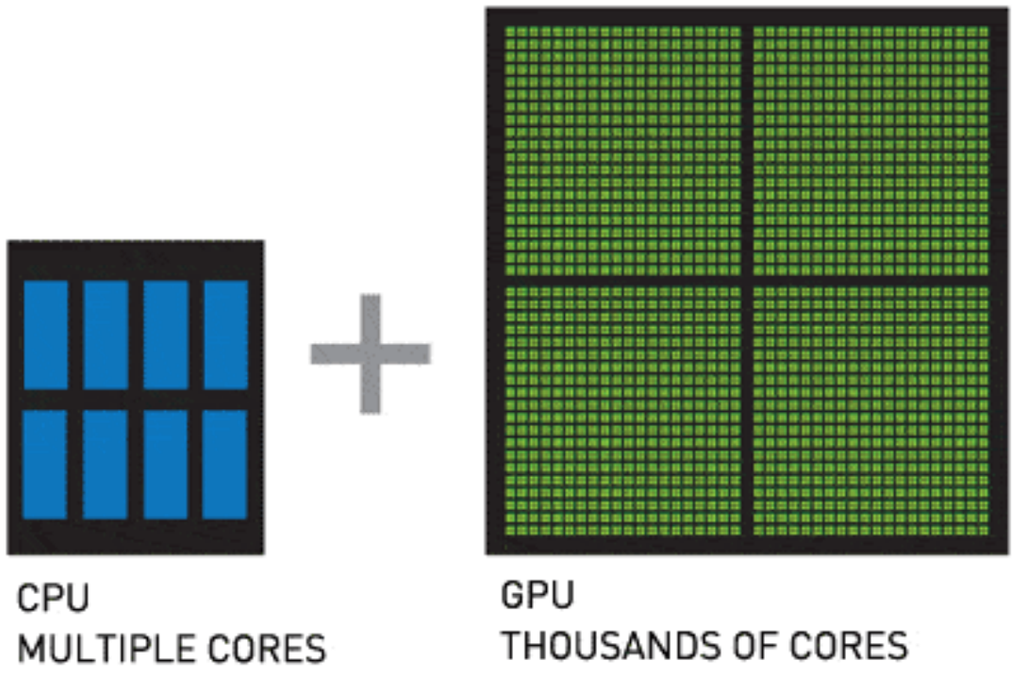
- A GPU as a specialized CPU that’s been built for a very specific purpose. Video rendering is all about doing simple mathematical operations over and over again, and that’s what a GPU is best at.

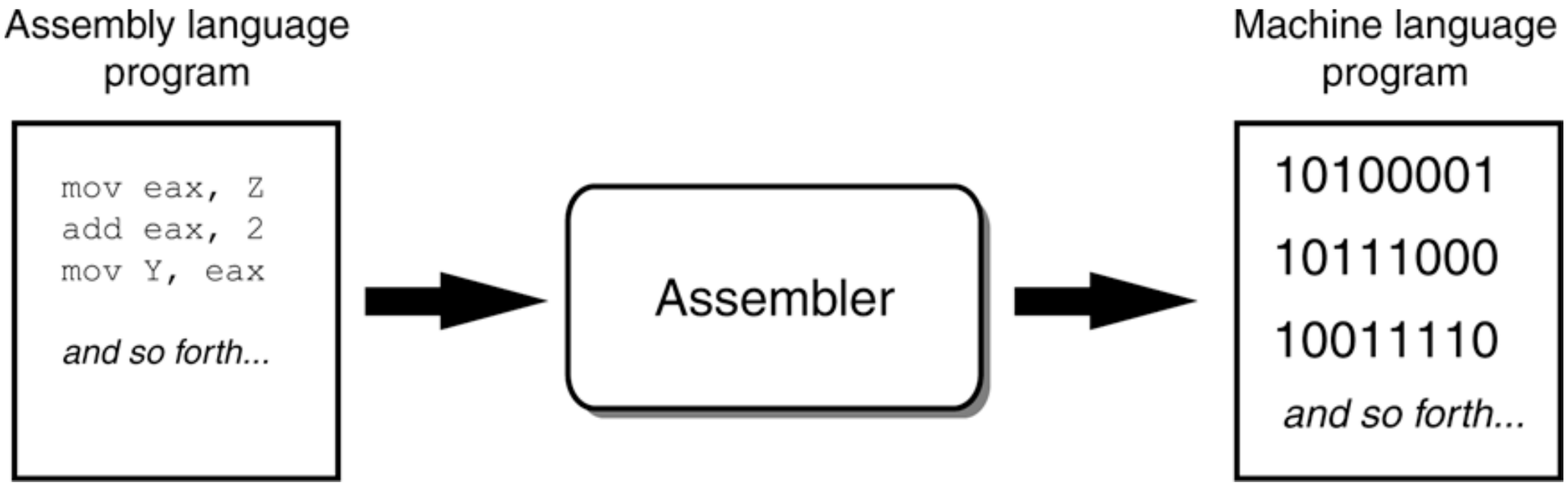
- This massive parallelism is what makes GPUs capable of rendering the complex 3D graphics required by modern games.



CPU vs GPU

- A GPU can only do a fraction of the many operations a CPU does, but it does so with incredible speed





What we want to do?

Programs Basic


Linux
MS DOS
ubuntu
BlackBerry
Mac OS
Apple
Microsoft
Windows
ANDROID
EssayCorp

Where we are?

Compiler & Interpreter

- Compiler:
  - Copy Program on Disk and let you run whenever that you want
  - Related to OS and Change the code for specific reason
  - High Speed
- Interpreter:
  - Run Line by Line
  - Need the program install on PC
  - Low Speed
  - Error find is more easily

Object Oriented Programming (OOP)

- It’s related to Object
- Class and Capasule
- Every Object :
    - Operation
    - Properties
- It could’t be change by anything out of their Obejct
- In OOP computer programs, they create objects and connect with them.
- programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.
- In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another.
- Your University Account
- It’s come from Brain

Basic OOP

- **Class** – A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

- **Class variable** – A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

- **Data member** – A class variable or instance variable that holds data associated with a class and its objects.

- **Function overloading** – The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.

- **Method** – A special kind of function that is defined in a class definition.

Basic OOP

- **Instance variable** – A variable that is defined inside a method and belongs only to the current instance of a class.

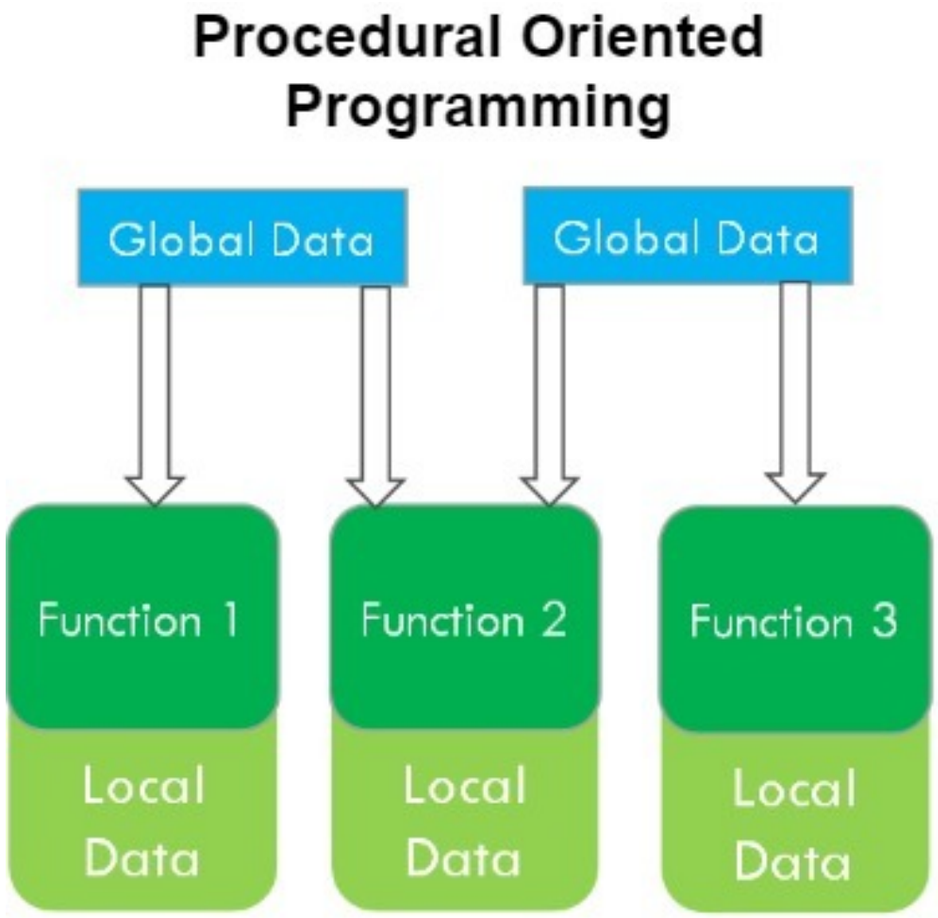
- **Inheritance** – The transfer of the characteristics of a class to other classes that are derived from it.

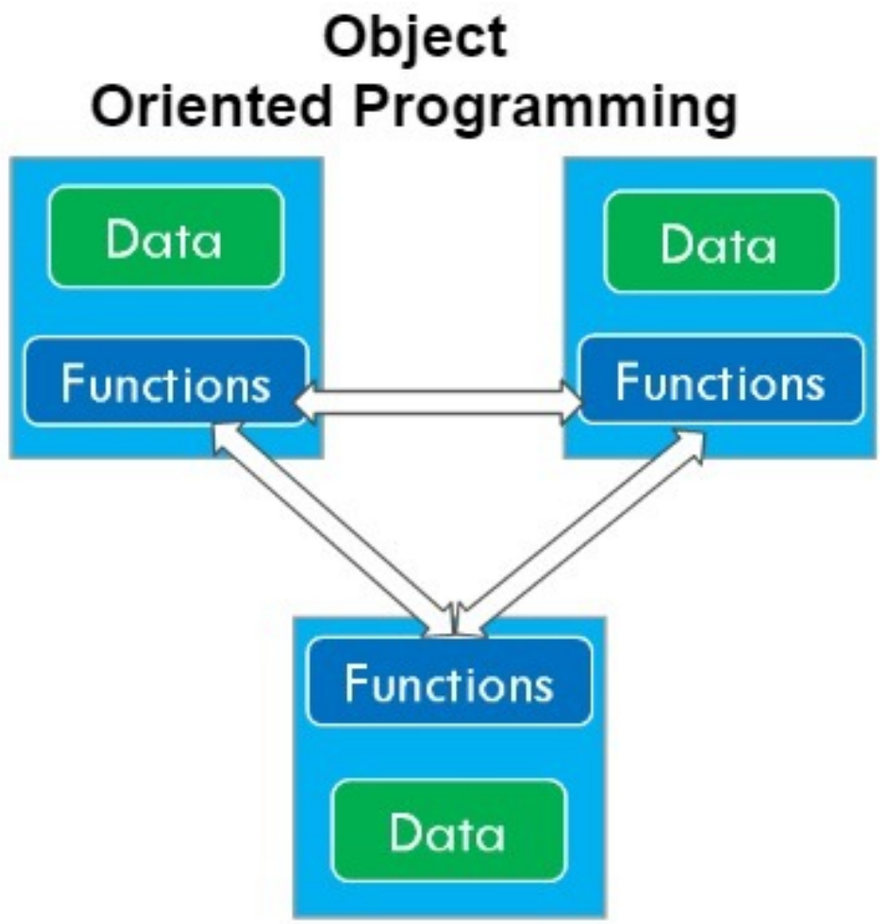
- **Instance** – An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.

- **Instantiation** – The creation of an instance of a class.

- **Object** – A unique instance of a data structure that's defined by it’s class. An object comprises both data members (class variables and instance variables) and methods.

- **Operator overloading** – The assignment of more than one function to a particular operator.


Procedural Oriented Programming
Global Data
Global Data
Function 1
Function 2
Function 3
Local Data
Local Data
Local Data


Object Oriented Programming
Data
Functions
Data
Functions
Functions
Data

OOP vs POP

History

- Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language

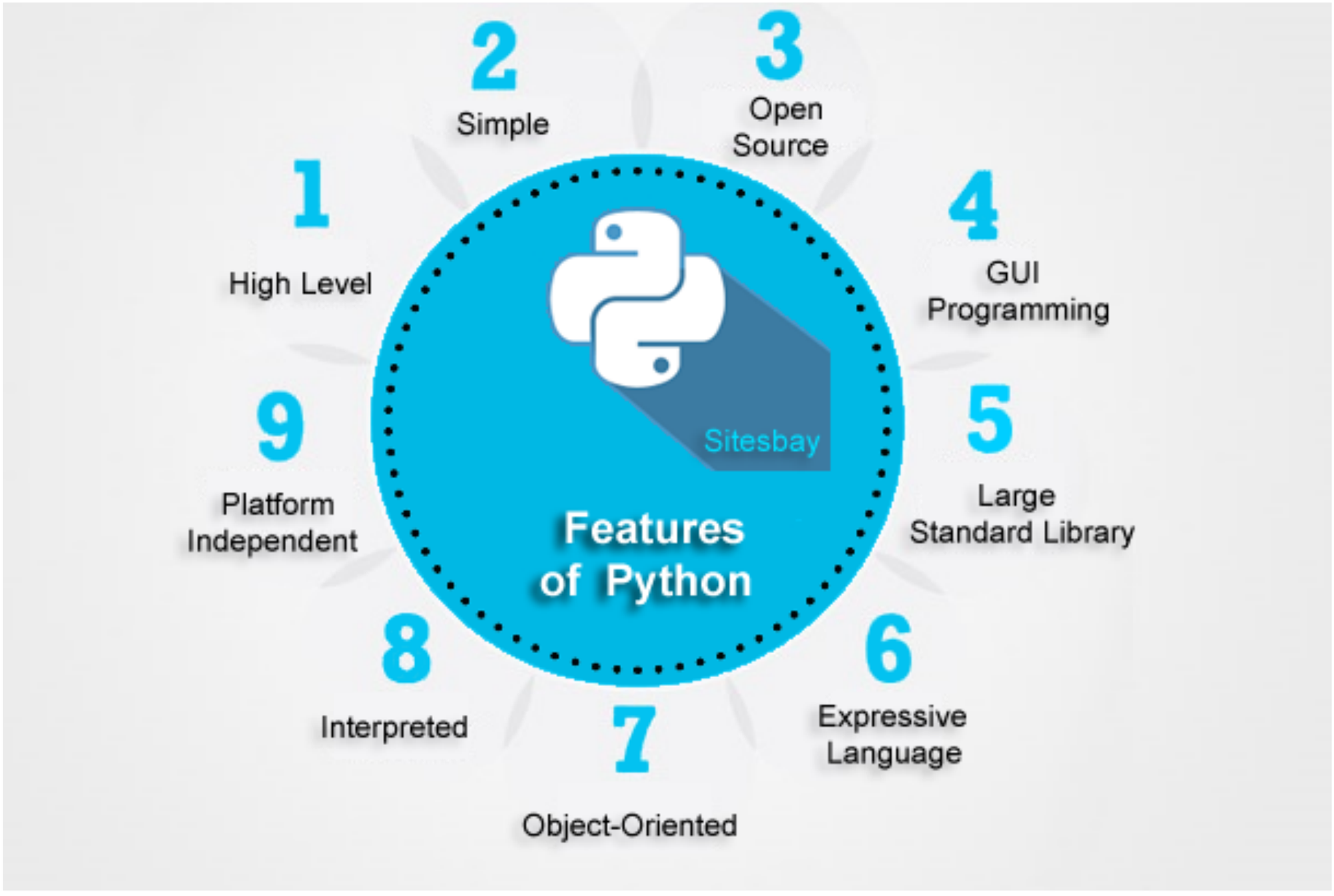
- Python 2.7's end of lifedate was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.

Why Python?

Comedy series from the 1970s

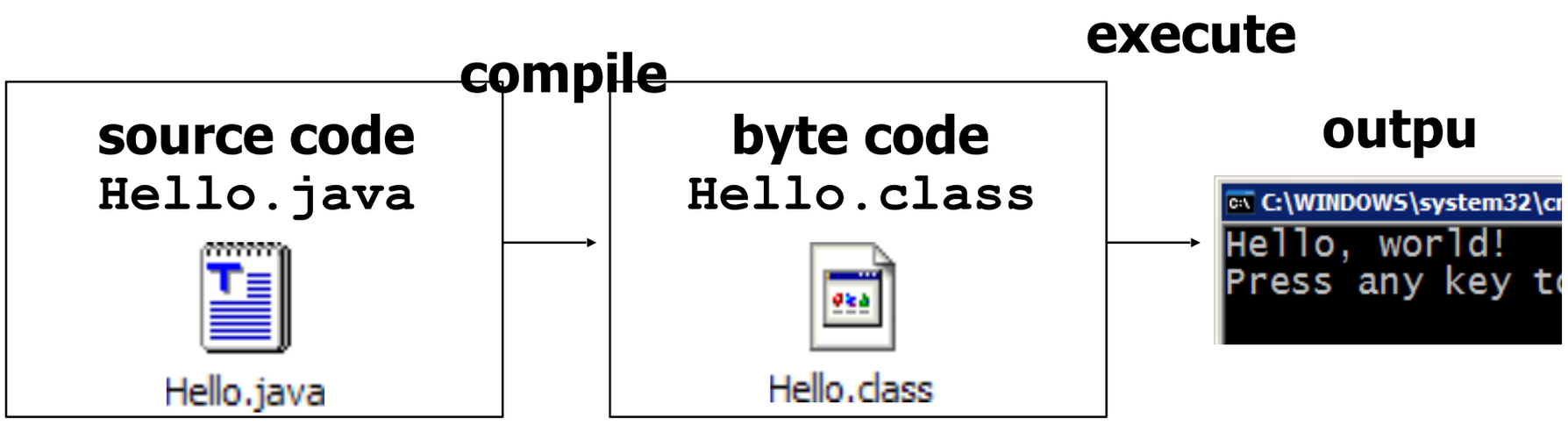
KEEP CALM AND CODE PYTHON

Why Python?

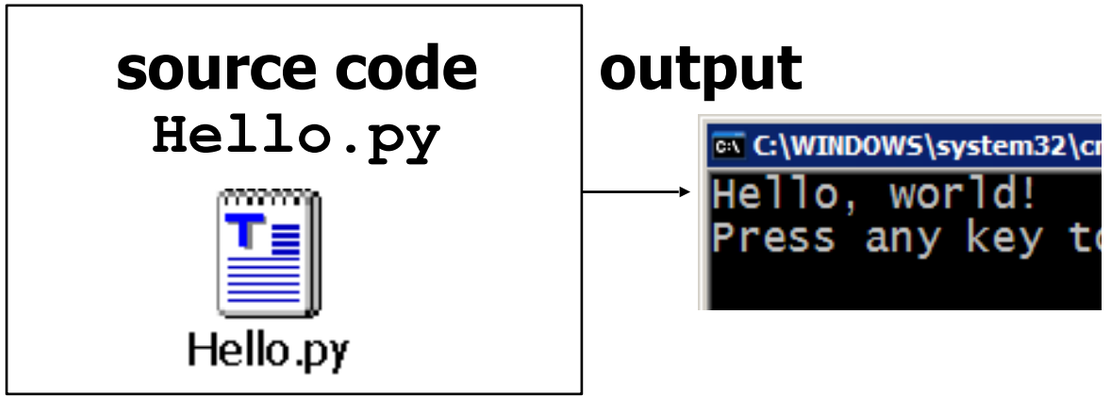

1
High Level
2
Simple
3
Open
Source
4
GUI
Programming
5
Large
Standard Library
6
Expressive
Language
7
Object-Oriented
8
Interpreted
9
Platform
Independent
Features
of Python
Sitesbay

Python vs Java

- Many languages require you to *compile* (translate) your program into a form that the machine understands.



- Python is instead directly *interpreted* into machine instructions.
-



HelloWorld.java

```java
package com.srccodes.example;

public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Program.cs*

HelloWorld.Program

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWorld {
    class Program {
        static void Main(string[] args) {
            Console.WriteLine("Hello World");
            Console.ReadKey();
        }
    }
}
```

Toolbox

HelloWorld.cpp*

HelloWorld (Global Scope)

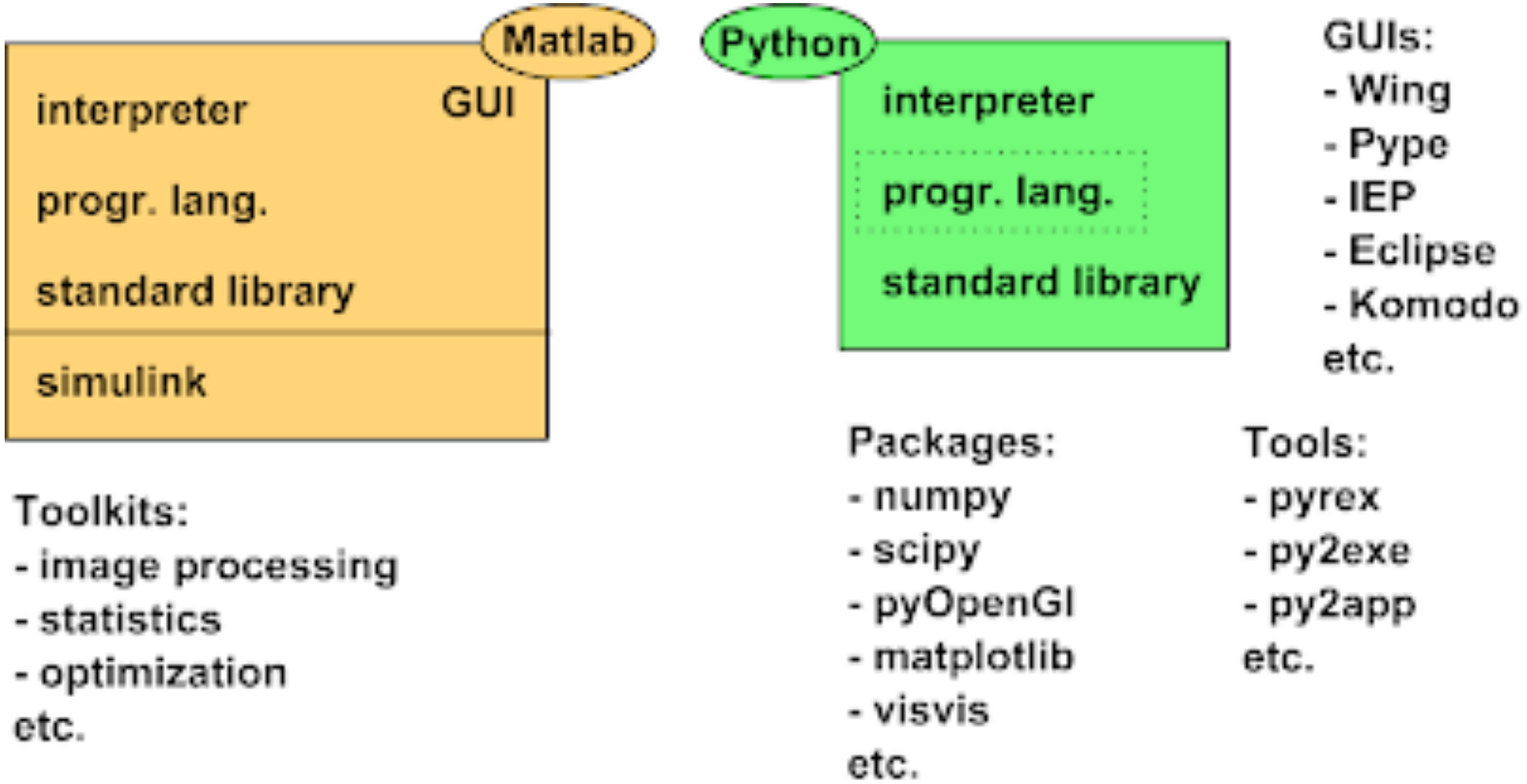
```cpp
// HelloWorld.cpp : Defines the entry point for the console a
//

#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World! ";
    return 0;
}
```

print(“hello world”)

Python Vs ~~Matlab~~ 1

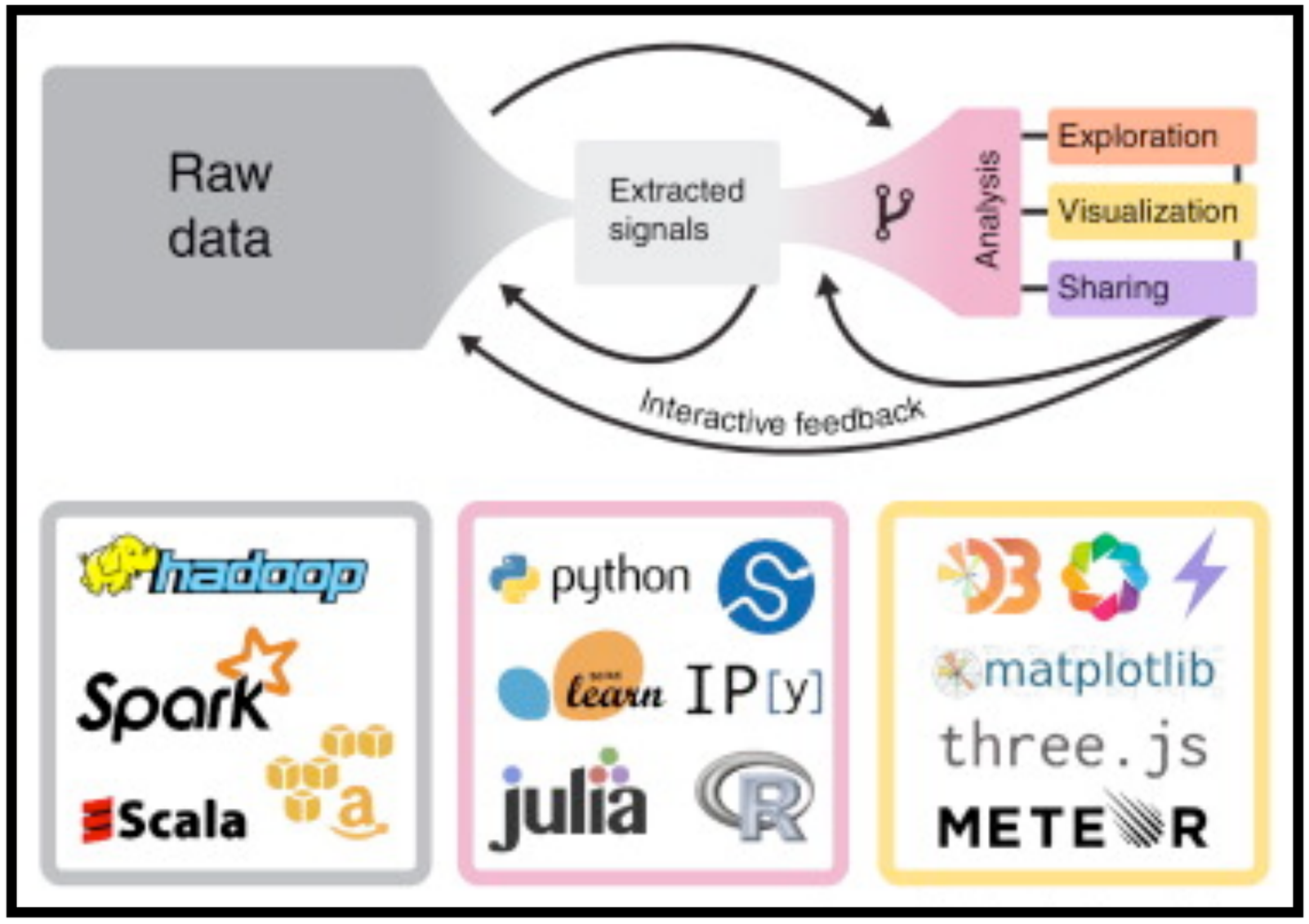
- MATLAB becomes increasingly useless as you get farther away from matrices. Python is equally good at everything.
    - Someday maybe I'll want to turn something into a stand-alone program with a GUI
    - pull data out of a pdf
    - interface with hardware and instruments
    - draw a 3D picture
    - write a script to reorganize files on my computer

vays has a professional-quality module for it!



Python vs ~~Matlab~~ 2

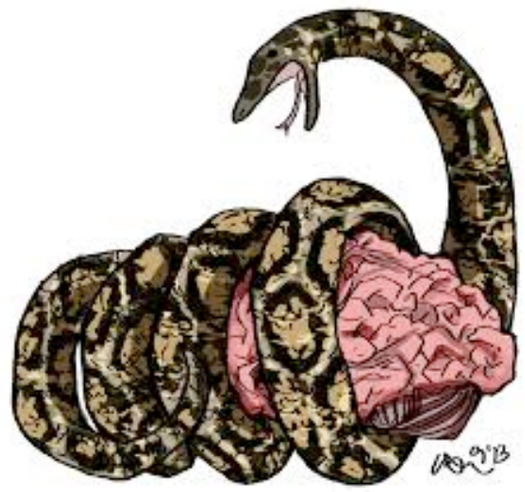
It’s just Comedy PROGRAMM



Python

New way of being Expert

Python



- Python is a general-purpose programming language

- Python "**package**"/ "**module**" / "**library**" / "**framework**" is what you download (or write) to get additional functions and definitions. Examples:

  - NumPy for fast array computations and manipulations and linear algebra.

  - SciPy for optimization, image-processing, interpolation, numerical integration, etc.

  - Matplotlib for making graphs.

module

- **Module** is a file which contains python functions , global variables etc. It is nothing but .py file which has python executable code / statement.For example: Let’s create a file Module.py:

```
    def welcome_message(user_name):
        return "Welcome " + name
```

- statement.For example: Let’s create a file Module.py:

```
    import user
    print user.welcome_message("Module")
```

package

- **Package** is namespace which contains multiple package/modules. It is a directory which contains a special file **__init__.py**

    - Let's create a directory **first.** Now this package contains multiple packages/modules to handle user related requests.

```
first/       # top level package
    __init__.py

    get/     # first subpackage
        __init__.py
        basic.py
        features.py
        preprocessing.py

    create/   # second subpackage
        __init__.py
        gui.py
        mac.py
```

- Now you can import it in following way

```
from user.get import basic # imports info module from get package
from user.create import gui #imports api module from create package
```

library

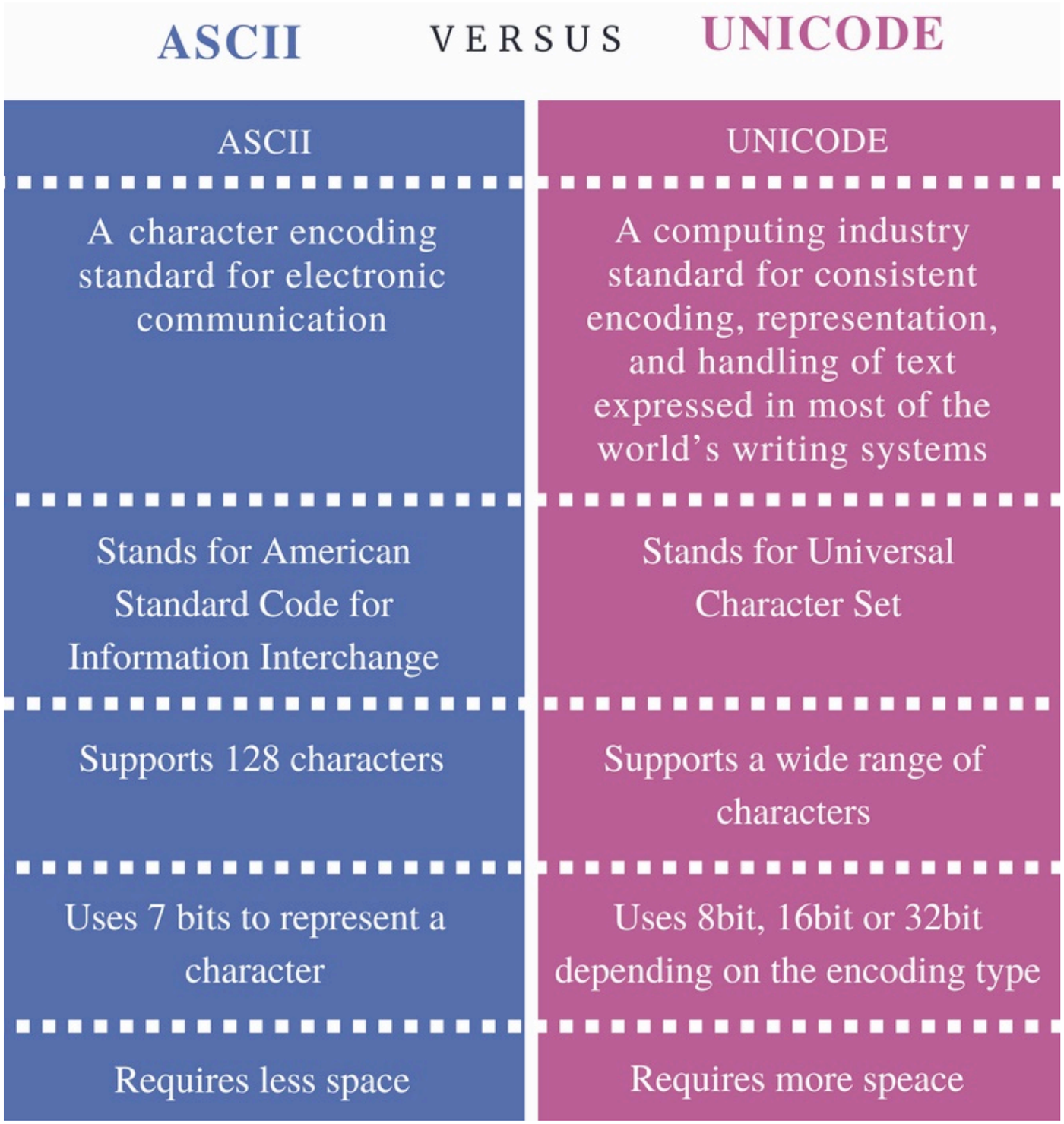
- It is collection of various packages. There is no difference between package and python library conceptually.

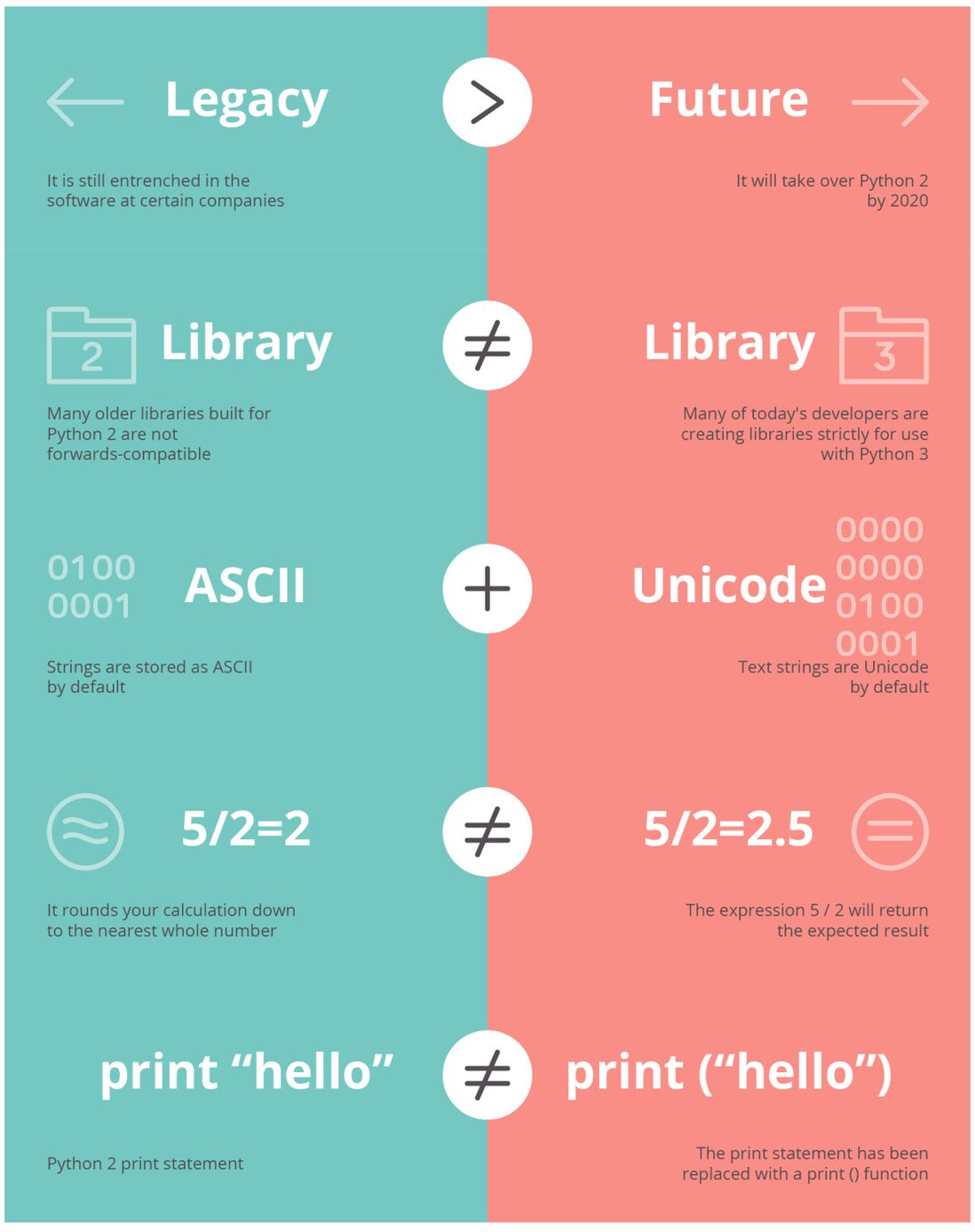
framework

- It is a collection of various libraries which architects the code flow.
  - Numpy
  - Matplotlib
  - pandas

ASCII vs UNICODE

What we should do?
Just Accept it!

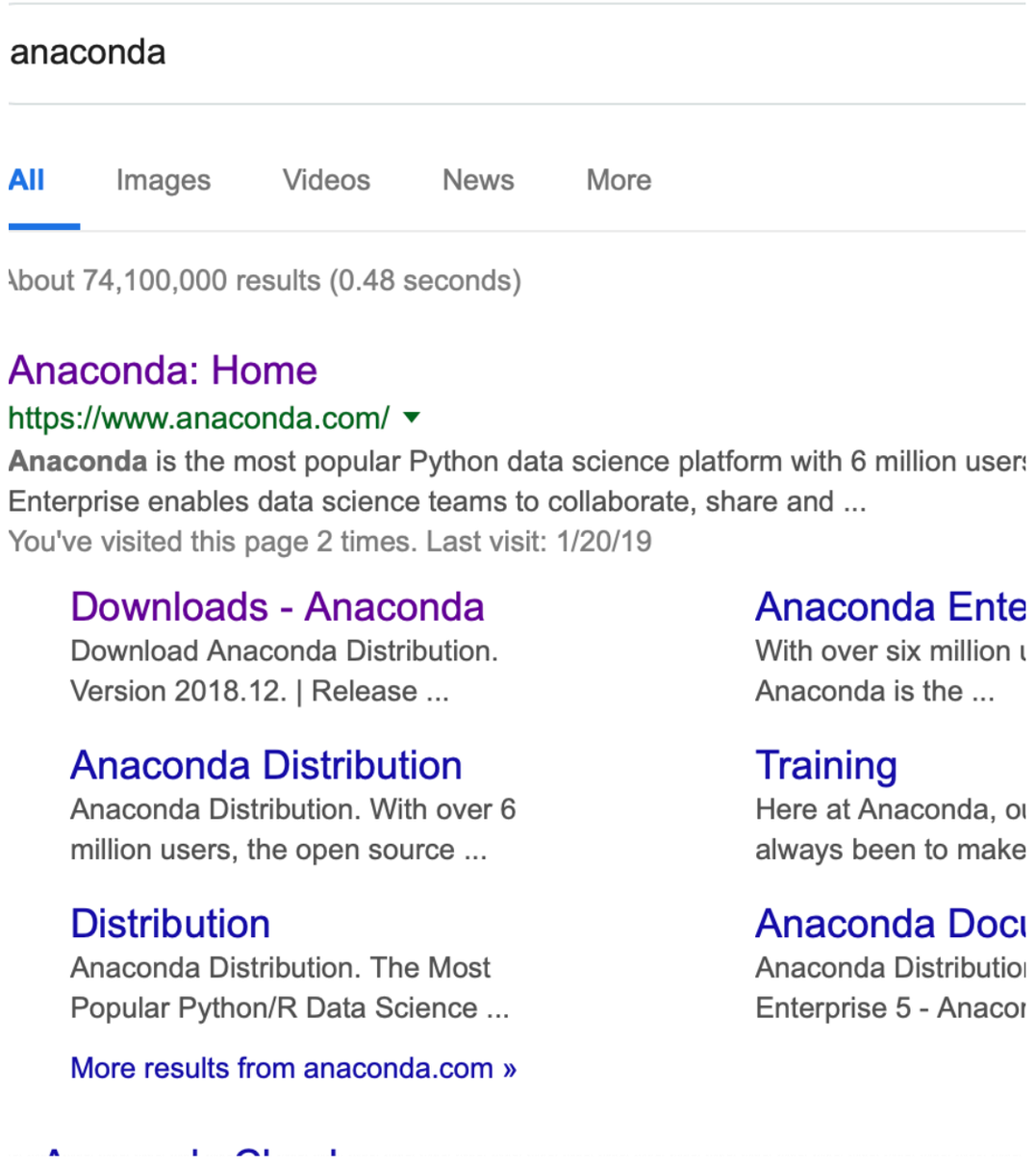




Python 2 vs Python 3

Anaconda (Python)

- The greatest Platform for ***Python***
- Download here:
  - https://www.anaconda.com/download/



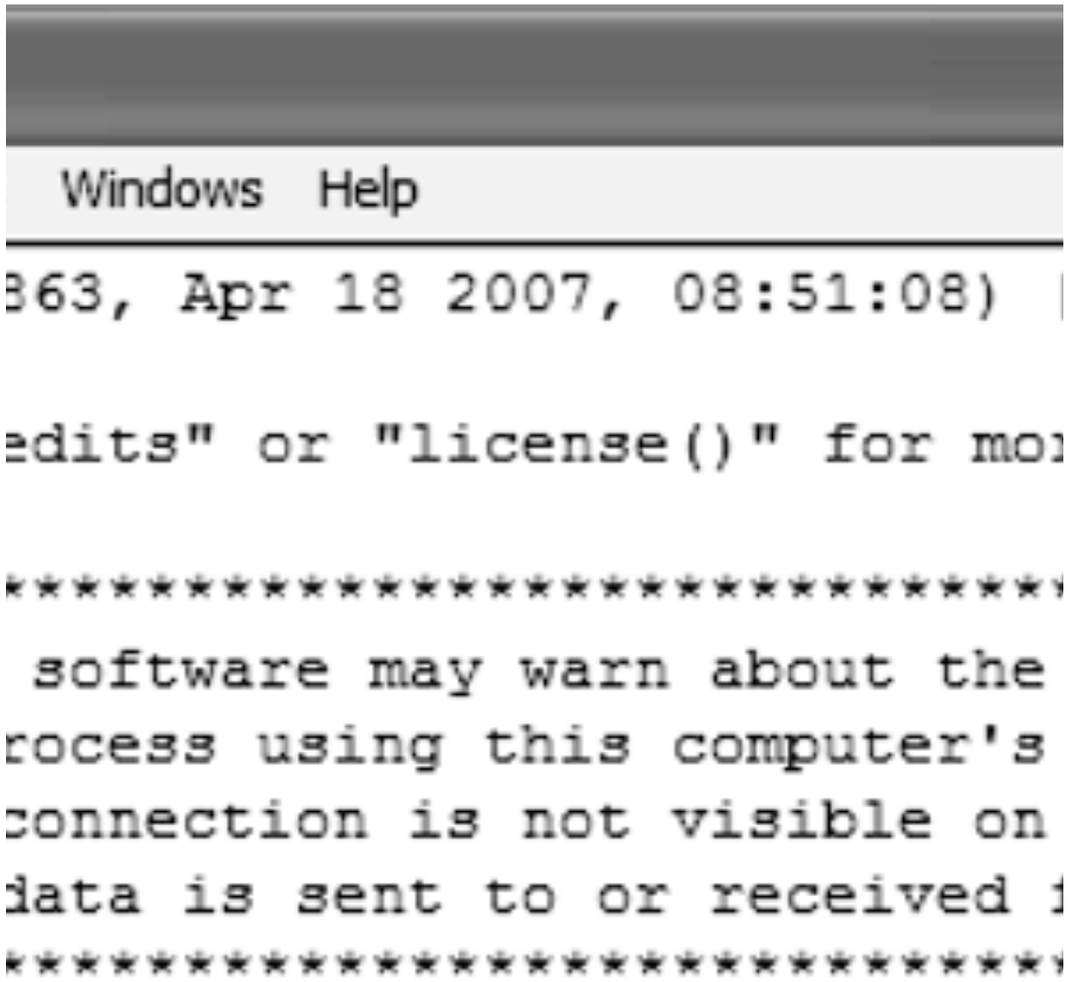
Conda Prompt (Win)

- Managing your python version:

  `conda --version`

  - `conda update conda`
  - Create your own env:
    - `conda create --name python=3.5`
    - List of Packages:
    - `conda list`

IDLE (Basic)

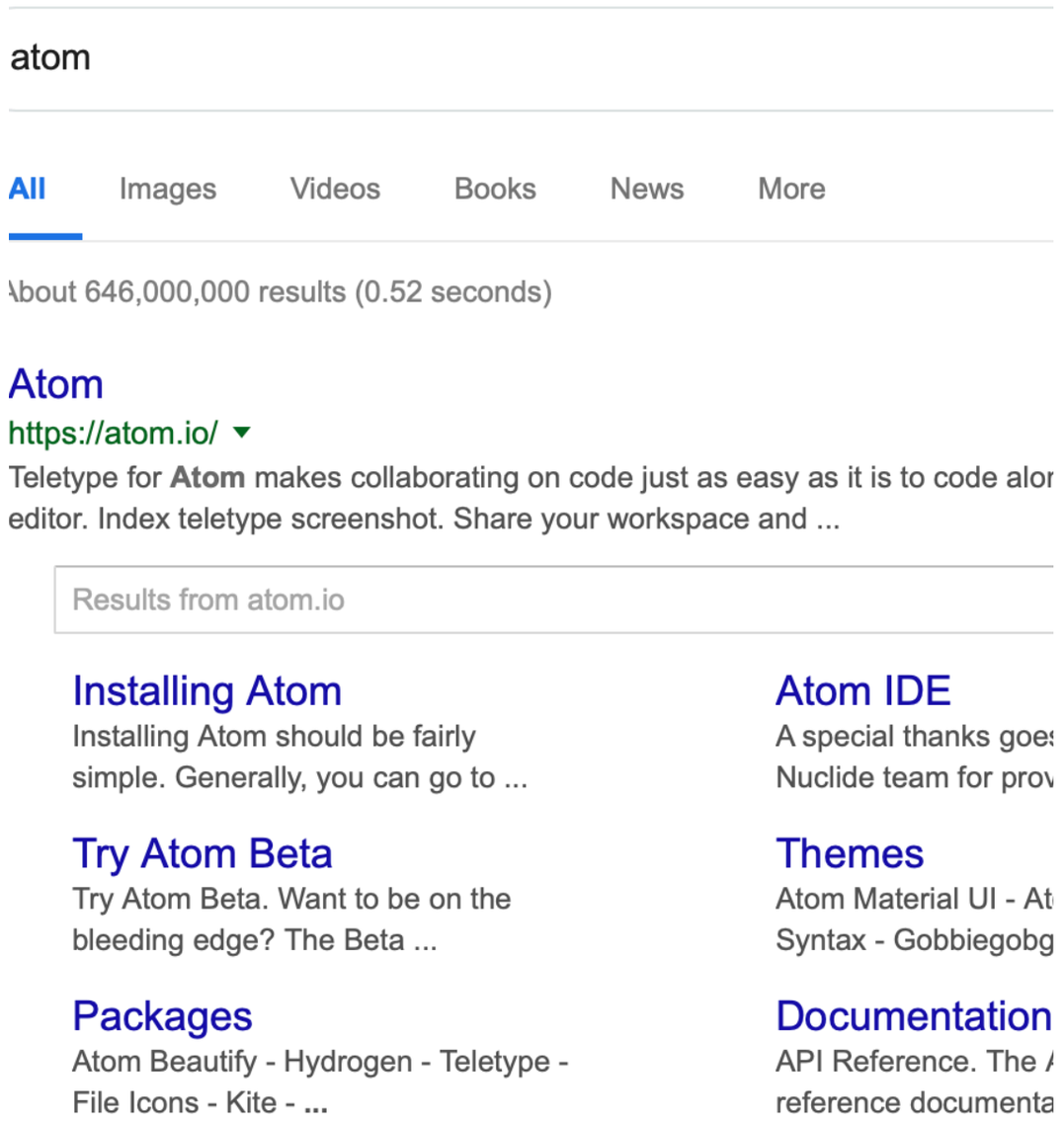


Different way of coding on python:

1. PyDev with Eclipse
2. Komodo
3. Emacs
4. Vim
5. TextMate
6. Gedit
7. Idle
8. PIDA (Linux)(VIM Based)
9. NotePad++ (Windows)
10. BlueFish (Linux)

Atom

- Atom :
  - One of the most important **IDLE** for python and it's great for connecting to **Github**
  - **IDLE : I**ntegrated **D**eve**L**opment **E**nvironment (IDLE)



Anaconda + Atom

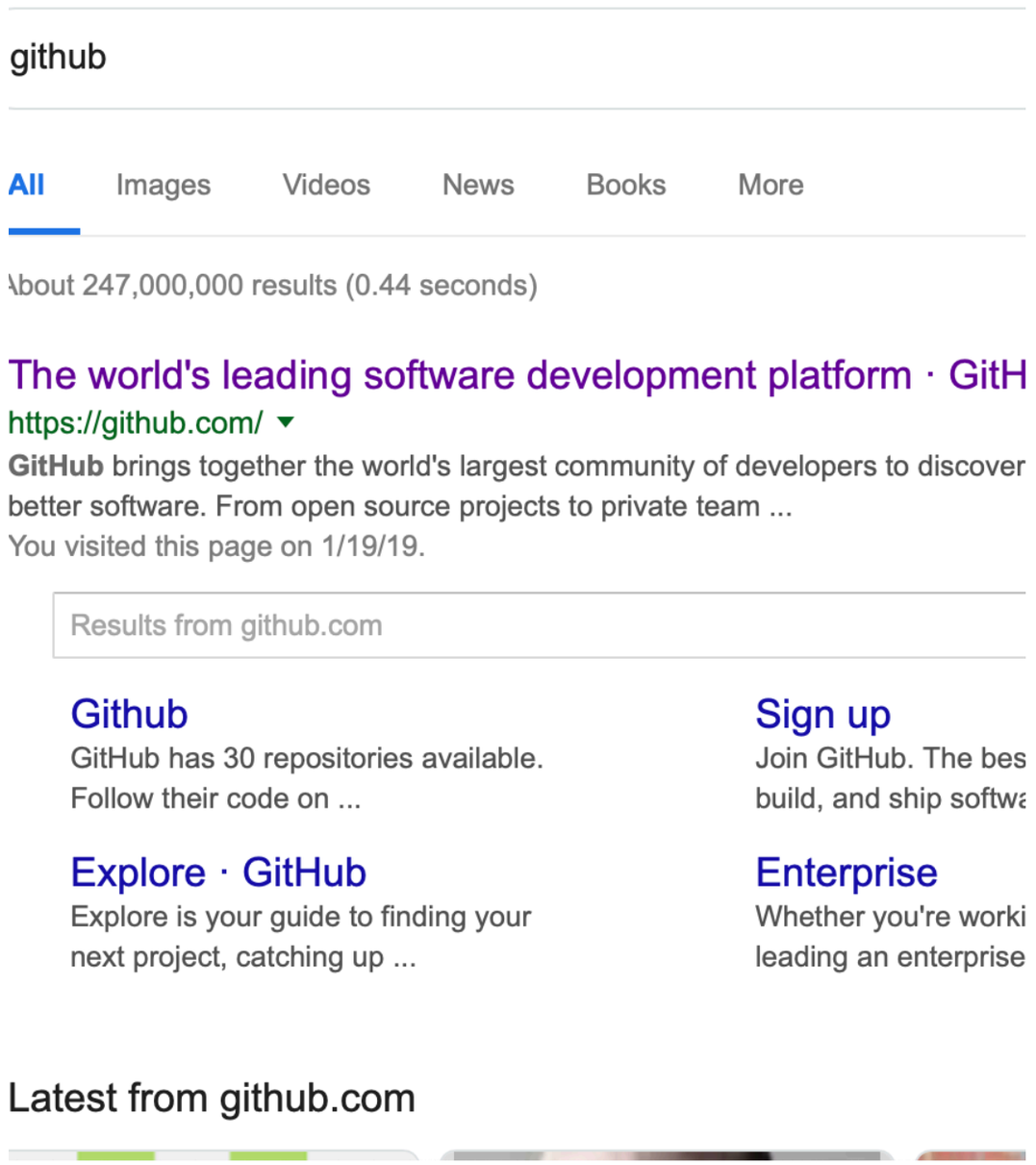
Please follow the rules: (Install All of them)

atom>settings>packages:

- file icons
- project manager
- hydrogen (apm install hydrogen)
- autocomplete-python

Github

- Why github?
- Being professional
- Other one can see you in work
- abmadani



sublime text

All Images Videos News Books More Settings Tools

About 188.000.000 results (0,49 seconds)

Sublime Text - A sophisticated text editor for code, markup and prose

https://www.sublimetext.com/

Sublime Text is a sophisticated text editor for code, markup and prose. You'll love the slick user interface, extraordinary features and amazing performance.

Download

Linux repos - Sublime Text 3.0 - Dev Builds - Sublime Text 2 - **...**

Sublime Text 2

Sublime Text 2 may be downloaded and evaluated for ...

Install for Linux

... of the Linux ecosystem, packages and package ...

Sublime Text - Download

Download. Sublime Text 3 may be downloaded from the Sublime ...

Buy

Buy Sublime Text. Sublime Text may be downloaded and ...

Sublime Forum

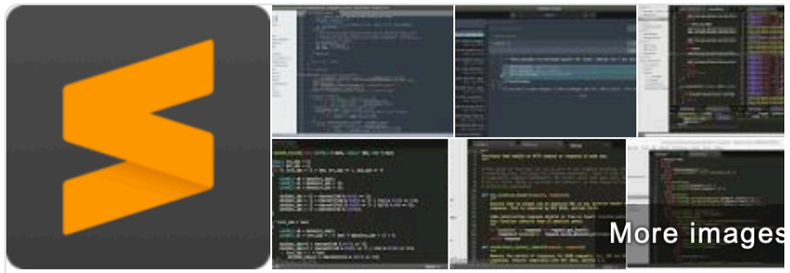
The official Sublime HQ forum.

More results from sublimetext.com »

People also ask

Is Sublime Text Safe?

Is there a free version of Sublime Text?



Sublime Text

Software

Sublime Text is a proprietary cross-platform source code editor with a Python application programming interface. It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses. Wikipedia

License: Proprietary software, Nagware

Initial release: January 18, 2008; 11 years ago

Developer(s): Jon Skinner, Will Bond

Stable release: 3.2.1 Build 3207 / April 6, 2019; 17 days ago

Size: 8.7 MB (Windows); 12.9 MB (OSX)

Sublime Text

Arithmetic Operators

Operator	Description	Example
=	Assignment	num **=** 7
+	Addition	num = 2 **+** 2
-	Subtraction	num = 6 **-** 4
*	Multiplication	num = 5 * 4
/	Division	num = 25 **/** 5
%	Modulo	num = 8 **%** 3
**	Exponent	num = 9 ** 2

Basic

Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1***`, `***value2***`)`	larger of two values
`min(`***value1***`, `***value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

from math import *

Math

This is especially important for division, as integer division produces a different result from floating point division:

```
10 // 3 produces 3

10 / 3 produces 3.3333

10.0 / 3.0 produces 3.3333333


int(3.3) produces 3

str(3.3) produces “3.3”

float(3) produces 3.0

float(“3.5”) produces 3.5

int(“7”) produces 7
```

Variables

variable: A named piece of memory that can store a value**.**

Format:

```
<name of variable> = <Information to be stored in the variable>
```

- Usage:
  - Compute an expression's result,
  - store that result into a variable,
  - and use that variable later in the program.

- assignment statement: Stores a value into a variable**.**
  - Syntax:

```
Integer (e.g., num1 = 10)
Floating point (e.g., num2 = 10.0)
Strings (e.g., name = "james")
```

  - Examples:

```
x = 5
```

Notes

- `Y` and enter
- You can assign to multiple names at the same time

```
x, y = 2, 3
x
y
```

- This makes it easy to swap values

```
x, y = y, x
```

- Assignments can be chained

```
a = b = x = 2
```

Avoid variables name

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Because they have their own functions

Constant Names

- They are similar to variables: a memory location that's been given a name.
- Unlike variables their contents *shouldn't* change.
- The naming conventions for choosing variable names generally apply to constants but the name of constants should be all UPPER CASE. (You can separate multiple words with an underscore).
- Example PI = 3.14
- They are capitalized so the reader of the program can distinguish them from variables.

★ Tip: Good for understanding better

Constants Vs Literals

- Named constant: given an explicit name

```
TAX_RATE = 0.2
afterTax = income - (income * TAX_RATE)
```

 # for single line

“ ” ”

Multiple line

“ ” ”

- Literal/unnamed constant/magic number: not given a name, the value that you see is literally the value that you have.

```
afterTax = 100000 - (100000 * 0.2)
```

Sequences Types

- **Tuple:** **`('john', 32, [CMSC])`**

  A simple *Immutable* ordered sequence of items
  Items can be of mixed types, including collection types

- **Strings:** **`"John Smith"`**

  Immutable
  Conceptually very much like a tuple

- **List:** **`[1, 2, 'john', ('up', 'down')]`**

  Mutable ordered sequence of items of mixed types

  Define tuples using parentheses and commas:

```
li = ['abc', 23, 4.34, 23]
li[1] = 45
    t = (23, 'abc', 4.56, (2,3), 'def')
     t[2] = 3.14

tu = (23, 'abc', 4.56, (2,3), 'def')
```

List

- All list is string value
- It’s Changeable
- How to index in list :
  - `print(list_name[?])`
- If our list is limited and we call out of range :
  - `IndexError: list index out of range`
- Each element or value in the list is name : **item**

Tuple

- Unchangeable data type

- Mixed of datas

Examples

```
t = (23, 'abc', 4.56, (2,3), 'def')
```

- Positive index: count from the left, starting with 0
  `t[1] :`
- Negative index: count from right, starting with –1
  `t[-3]`
- Define lists are using square brackets and commas
  `li = [“abc”, 34, 4.34, 23]`
  `li[1]`
  `li[1:4]`

Negative indices count froz end

`t[1:-1]`

Omit first index to make copy starting from beginning of the container

`t[:2]`

Omit second index to make copy starting at first index and going to end

`t[2:]`

Boolean

- Boolean test whether a value is inside a container:

```
t = [1, 2, 4, 5]
3 in t
False
4 in t
True
4 not in t
False
```

- For strings, tests for substrings

```
a = 'abcde'
'c' in a
True
'cd' in a
True
'ac' in a
False
```

+

The + operator produces a *new* tuple, list, or string whose value is the concatenation of its arguments.

```
(1, 2, 3) + (4, 5, 6)
 (1, 2, 3, 4, 5, 6)

 [1, 2, 3] + [4, 5, 6]
 [1, 2, 3, 4, 5, 6]

“Hello” + “ ” + “World”
 ‘Hello World’
```