

# Python

Inteligencia Artificial



**Marco Teran**

2021 - Bogotá

# Contenido

- 1 Introducción a Python
- 2 Características de Python
- 3 Tipos de datos
- 4 Listas
- 5 Tuplas
- 6 Diccionarios
- 7 Entorno de desarrollo Anaconda
- 8 Recursos

# Introducción a Python

# Historia



- Python fue creado por **Guido van Rossum** en 1999
  - Da este nombre al lenguaje inspirado por el popular grupo cómico británico Monty Python
  - Guido creó Python durante unas vacaciones de navidad en las que (al parecer) se estaba aburriendo

# Python's Benevolent Dictator For Life

*"Python is an experiment in how much freedom program-mers need. Too much freedom and nobody can read another's code; too little and expressive-ness is endangered."*

**Guido van Rossum**



# Hola Mundo en Python

```
#!/usr/bin/env python  
print "Hola Mundo" # "Hola Mundo"
```

# PYTHON



## WHICH VERSION OF PYTHON USE THE MOST?

47%

53%

Python 2

Python 3



## PYTHON DEVS INTERESTING FACTS

- ☐ Use Auto completion in editor more than 75%
- ☐ Use the debugger more or less 50%
- ☐ Use databases and SQL less than 60%



## 3 REASONS TO LEARN PYTHON



Python is used for web development



Want a higher salary?  
You Should Learn Python **Right Now**



Interested into Cyber Security?  
[Dive into Python](#)



## WHAT TECHNOLOGIES AND/OR FRAMEWORKS USE PYTHON PROGRAMMER?

67%

django

52%

ANACONDA

41%

Flask

32%

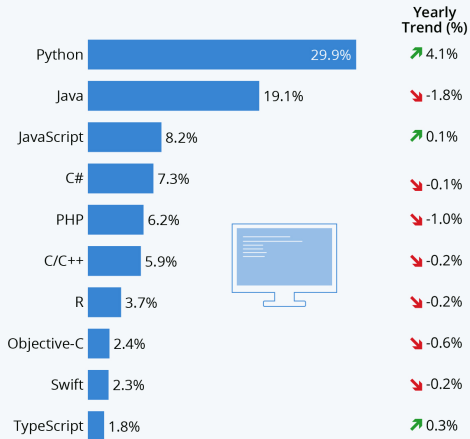
SQLAlchemy

**77%** of Python programmers develop Web Back-End applications.

**34%** do data analysis or develop Business Intelligence apps.

# Python Remains Most Popular Programming Language

Popularity of each programming language based on share of tutorial searches in Google



Yearly trend compares percent change from Feb 2019 to Feb 2020  
Sources: GitHub, Google Trends



# PYTHON FRAMEWORKS FOR DATA SCIENCE

## NUMPY

It is a Python library that handles most of the numerical computing done using Python. It provides support for multi-dimensional arrays and matrices and comes with an impressive collection of routines to operate the arrays.



## SCIPY

SciPy is a Python library that is commonly used in applications that call for scientific computing by scientists, engineers, and other technical fields. It has modules for signal processing, integration, solving ODEs, linear algebra, and more.



## TENSORFLOW

TensorFlow is a platform that was created by the Google Brain Team with the sole purpose of making it easy for you to build Machine Learning (ML) models.



## KERAS

Keras is a Python-based API that can run on TensorFlow, Theano, or CNTK. It is essentially a neural-network library that is designed to facilitate quick experiments on neural networks.



## MATPLOTLIB

Matplotlib is mainly used for data visualization through plotting. Matplotlib is analogous to MATLAB in terms of application with the advantage of allowing you to program using Python which also means that it is open-source and free.



## PANDAS

Pandas is a library that is used for data computation and analysis. It is extensively used for data wrangling which explains its popularity when any form of data analysis is involved.



# Características de Python

# Características de Python

- Muy legible y elegante
  - Imposible escribir código ofuscado
- Simple y poderoso
  - Minimalista: todo aquello innecesario no hay que escribirlo (;, {, }, ...)
  - Muy denso: poco código hace mucho
  - Soporta objetos y estructuras de datos de alto nivel: strings, listas, diccionarios, etc.
  - Múltiples niveles de organizar código: funciones, clases, módulos, y paquetes
  - Si hay áreas que son lentas se pueden reemplazar por *plugins* en C o C++, siguiendo la API para extender o empotrar Python en una aplicación, o a través de herramientas como SWIG, sip o Pyrex.

# Características de Python

- De scripting
  - No tienes que declarar constantes y variables antes de utilizarlas
  - No requiere paso de compilación/linkage
    - La primera vez que se ejecuta un script de Python se compila y genera bytecode que es luego interpretado
  - Alta velocidad de desarrollo y buen rendimiento
- Código interoperable (como en Java *"write once run everywhere"*)
  - Se puede utilizar en múltiples plataforma (más aún que Java)
  - Puedes incluso ejecutar Python dentro de una JVM (Jython)
- Open source
  - Razón por la cual la Python Library sigue creciendo
- De propósito general
  - Puedes hacer en Python todo lo que puedes hacer con *C#* o Java, o más

## Peculiaridades sintácticas

- Python usa tabulación (o espaciado) para mostrar estructura de bloques
  - Tabula una vez para indicar comienzo de bloque
  - Des-tabula para indicar el final del bloque

Código en C/Java	Código en Python
<pre>if (x) {     if (y) {         f1();     }     f2(); }</pre>	<pre>if x:     if y:         f1()     f2()</pre>

## ¿Para qué [no] es útil?

- Python no es el lenguaje perfecto, no es bueno para:
  - Programación de bajo nivel (system-programming), como programación de drivers y kernels
    - Python es de demasiado alto nivel, no hay control directo sobre memoria y otras tareas de bajo nivel
  - Aplicaciones que requieren alta capacidad de computo
    - No hay nada mejor para este tipo de aplicaciones que el viejo C
- Python es ideal:
  - Como lenguaje "pegamento" para combinar varios componentes juntos
  - Para llevar a cabo prototipos de sistema
  - Para la elaboración de aplicaciones cliente
  - Para desarrollo web y de sistemas distribuidos
  - Para el desarrollo de tareas científicas, en los que hay que simular y prototipar rápidamente

## Usando Python desde línea de comando

Para arrancar el intérprete (Python interactivo) ejecutar:

```
C:\>python
```

```
Python 2.4 (#60, Nov 30 2004, 11:49:19) [MSC v.1310 32 bit (Intel)] on win
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

Un comando simple:

```
>>> print "Hola Mundo"
```

```
Hola Mundo
```

```
>>>
```

Para salir del intérprete Ctrl-D (en Linux) o Ctrl-Z (en Windows)

# Tipos de datos



# Tipos de datos

**Numéricos** (integer, long integer, floating-point, and complex)

```
>>> x = 4
```

```
>>> int (x)
```

```
4
```

```
>>> long(x)
```

```
4L
```

```
>>> float(x)
```

```
4.0
```

```
>>> complex (4, .2)
```

```
(4+0.2j)
```

# Tipos de datos

**Strings** delimitados por un par de (' , " , "" "" )

- Dos string juntos sin delimitador se unen

```
>>> print("Hi" "there")  
Hithere
```

- Los códigos de escape se expresan a través de "\":

```
>>> print "\n"
```

## Tipos de datos

Algunos de los métodos que se pueden aplicar a un string son:

```
len('La vida es mucho mejor con Python.')
```

```
>>> 34
```

```
'La vida es mucho mejor con Python.'.upper()
```

```
>>> 'LA VIDA ES MUCHO MEJOR CON PYTHON'
```

```
"La vida es mucho mejor con Python".find("Python")
```

```
>>> 27
```

```
"La vida es mucho mejor con Python".find('Perl')
```

```
>>> -1
```

```
'La vida es mucho mejor con Python'.replace('Python', 'Jython')
```

```
>>> 'La vida es mucho mejor con Jython'
```

# Otros objetos de Python

**Listas** (conjuntos mutables de cadenas)

```
var = [] # crea una lista
var = ['uno', 2, 'tres', 'plátano']
```

**Tuplas** (conjuntos inmutables)

```
var = ('uno', 2, 'tres', 'plátano')
```

**Diccionarios** (arreglos asociativos o 'hashes')

```
var = {} # crear diccionario
var = {'lat': 40.20547, 'lon': -74.76322}
var['lat'] = 40.2054
```

Cada uno tiene su propio conjunto de métodos

# Listas

# Listas

- Piensa en una lista como una pila de tarjetas, en las que está escrita tu información
- La información permanece en el orden en que la colocas hasta que modificas ese orden
- Los métodos devuelven una cadena o un subconjunto de la lista o modifican la lista para añadir o eliminar componentes
- Escrito como `var[index]`, el índice se refiere al orden dentro del conjunto (piense en el número de tarjeta, empezando por el 0)
- Puede recorrer las listas como parte de un bucle

## Otros objetos de Python

### ■ Añadir a la lista

`var[n] = objeto` *# sustituye n por el objeto*

`var.append(objeto)` *#añade el objeto al final de la lista*

### ■ Eliminar de la lista

`var[n] = []` *# vacía el contenido de la tarjeta, pero conserva el orden*

`var.remove(n)` *# elimina la tarjeta en n*

`var.pop(n)` *# elimina n y devuelve su valor*

# Tuplas



# Tuplas

- Al igual que una lista, las tuplas son matrices iterables de objetos
- Las tuplas son inmutables: una vez creadas, no se pueden modificar
- Para añadir o eliminar elementos, hay que volver a declararlos

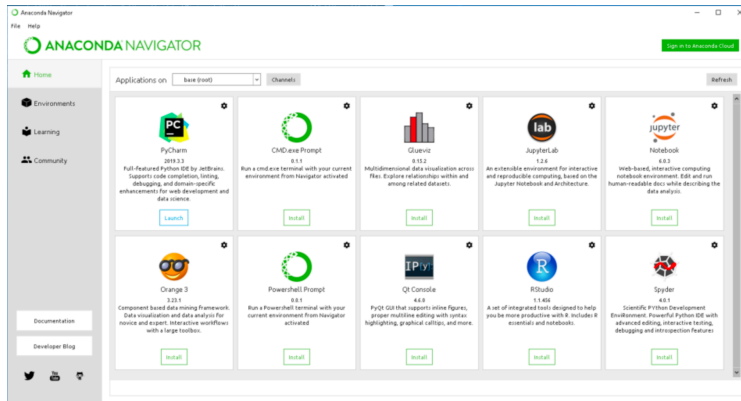
# Diccionarios

# Diccionarios

- Los diccionarios son conjuntos de pares de claves y valores
- Permite identificar los valores por un nombre descriptivo en lugar de ordenarlos en una lista
- Las claves no están ordenadas a menos que se ordenen explícitamente
- Las claves son únicas:
  - `var['item'] = "manzana"`
  - `var['item'] = "plátano"`
  - `print var['item']` imprime sólo plátano

# Entorno de desarrollo Anaconda

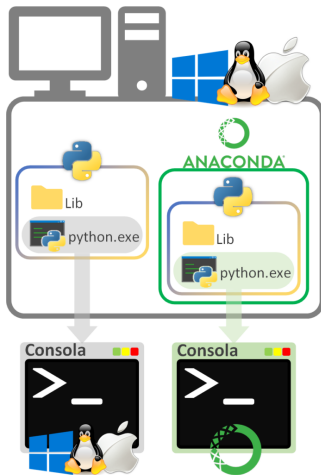
# Entorno de desarrollo Anaconda



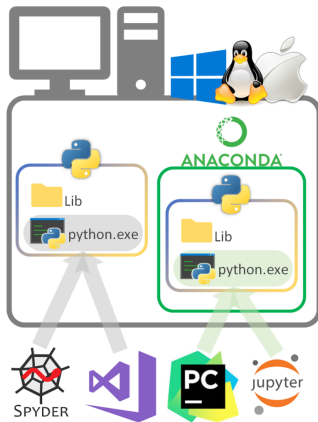
## Contienen:

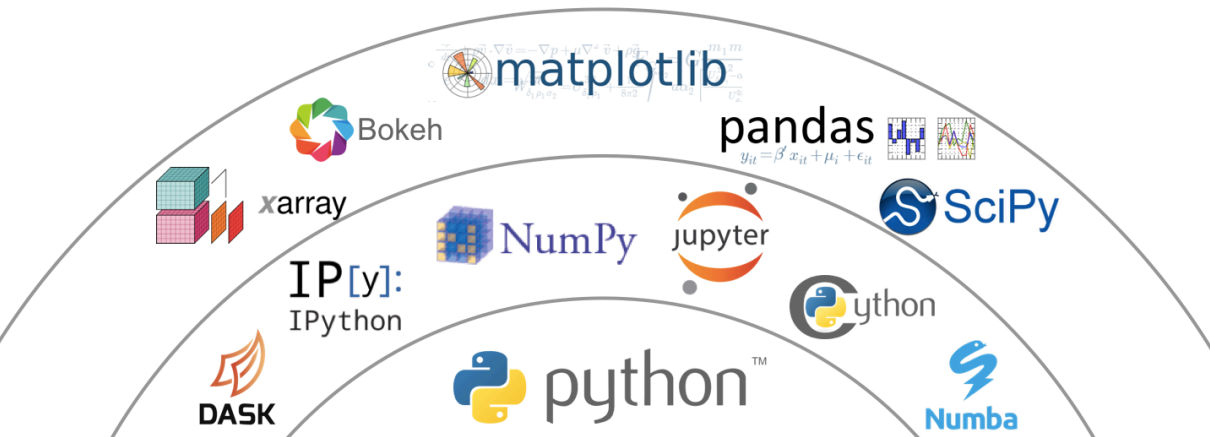
- Código fuente (e.g. python)
- Elementos de texto enriquecido (párrafos, ecuaciones, figuras, enlaces, etc.)

## Entorno de desarrollo Anaconda



# Entorno de desarrollo Anaconda







# Recursos

# Recursos

- Summer 2011 CSC108: Introduction to Computer Programming
- Python 3.9.1 documentation
- Python For Beginners (oficial)

# Muchas gracias por su atención

*¿Preguntas?*



**Contacto:** Marco Teran  
**webpage:** [marcoteran.github.io/](https://marcoteran.github.io/)