



# Introducción a Python

Conceptos básicos

Por: Hugo Alberto Rivera Díaz



XpectreLabs

# ¿Qué es Python?

Python es un lenguaje de programación popular. Fue creado por Guido van Rossum y lanzado en 199

Desarrollo Web

Desarrollo de Software

Data Science

Configuración de  
Sistemas

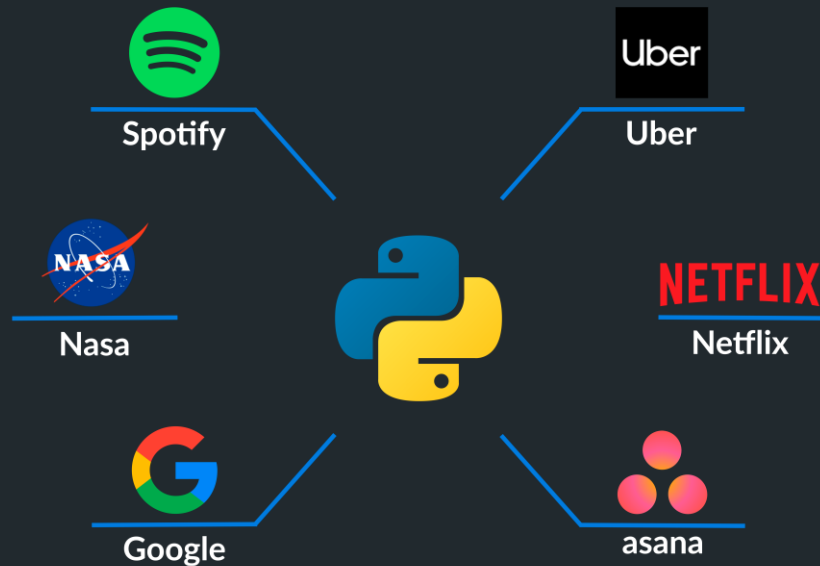
## ¿Qué puede hacer Python?

- Crear aplicaciones web en un servidor
- Ser usado para crear flujos de trabajo
- Se puede conectar a sistemas de bases de datos.
- Manejo de big data y matemáticas complejas
- Prototipado rápido hasta un software listo para producción



# ¿Por qué Python?

- Multiplataforma
- Sintaxis simple similar al idioma inglés
- Sintaxis que permite a los desarrolladores reducir líneas de código
- Sistema Interprete = Prototipado rápido
- Puede usarse de manera orientada a objetos, funcional o procedural



# Conoce tu versión de Python

- Abre una nueva consola en tu computadora.
- Escribe **python** en Windows o **python3** en Mac OS/Linux y pulsa **enter**.

```
hug0@DESKTOP-B2395G1:~$ python3
Python 3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Puedes imprimir tu primer **"Hola Mundo"** en **python** escribiendo consola

```
>>> print("Hola Mundo")
Hola Mundo
>>>
```

# Variables en Python

- **Python**, es **fuertemente tipado** aunque no lo parezca, ya que nosotros definimos variables sin especificar el tipo de dato

- **A = 25**
- **B = "NodeSchool"**
- **C = True**
- **D = [1, 2, 3]**

## PALABRAS RESERVADAS PYTHON

- No pueden usarse como nombres de variables

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		



# Basic Stuff

- Como en la mayoría de los lenguajes, empezaremos familiarizándonos con **python**, realizando en la consola algunas **operaciones básicas**
  - **2 + 3**
  - **4 \* 5**
  - **5 - 1**
  - **40 / 2**
  - **50 - 20 \* 2**
  - **(20 - 10) \* 6**
  - **10 // 3** (cociente)
  - **1 + 1j** (números complejos)
  - **10 % 3** (residuo)
  - **2 \*\* 3** (potencia)



# Funciones matemáticas

- **Python** cuenta con una extensa variedad de **funciones matemáticas** para el manejo y ejecución de cálculos de una manera más sencilla

- `sum([1, 2, 3])`
- `min([1, 2, 3])`
- `max([1, 2, 3])`
- `abs(-7)`
- `round(3.942, 2)`
- `pow(2, 3)`


*Importando el módulo **math***

- `math.factorial(5)`
- `math.floor(29.75)`
- `math.trunc(77.29)`
- `math`
- `print(math.pi, math.e)`
- `math.sqrt(81)`

# Cadenas de caracteres

- **Python** interpreta lo que existe entre comillas simples o dobles como una cadena de caracteres.
- **len("Hola")** (Encontrar el tamaño de una palabra)
- **"Hola" + "Mundo"** (Concatenación)
- **"Hola Mundo".find("Mundo")** (Encontrar secuencia)
- **"Hola Mundo".strip()** (Eliminar espacios)
- **"Hola Mundo".upper()** (Convertir Mayúsculas)
- **"Hola Mundo".lower()** (Convertir minúsculas)
- **"El mensaje es {}".format("Hola Mundo")** (Formato en cadenas)





# Una cadena es una lista pero una lista no siempre es una cadena



```
cadena = "Hola mundo"  
lista = [2, 4, 6, 8]
```

```
print(len(a))
```

```
print(len(b))
```

```
print(a[1])
```

```
print(b[1])
```

# Operadores lógicos

- Comparar cosas - en Python puedes comparar cosas haciendo uso de `>`, `>=`, `==`, `<=`, `<` y de los operadores **and**, **or** y **not**

**and**

El operador and evalúa si el valor del lado izquierdo y el lado derecho *se cumple*.

```
>>> True and False
False
```

**not**

El operador not devuelve el valor opuesto la valor booleano.

```
>>> not True
False
```

**or**

El operador or evalúa si el valor del lado izquierdo o el lado derecho se cumple.

```
>>> True or False
True
```

- Boolean** - un tipo de objeto que sólo puede tener uno de dos valores: **True** o **False**

```
>>> 5 > 2
```

```
True
```

```
>>> 3 < 1
```

```
False
```

```
>>> 5 > 2 * 2
```

```
True
```

```
>>> 1 == 1
```

```
True
```

```
>>> 5 != 2
```

```
True
```

```
>>> 6 > 2 and 2 < 3
```

```
True
```

```
>>> 3 > 2 and 2 < 1
```

```
False
```

```
>>> 3 > 2 or 2 < 1
```

```
True
```

# Listas

- Una **lista** es una **estructura de datos** y un tipo de dato en **Python** con características especiales. Nos permiten almacenar cualquier tipo de valor como enteros, cadenas y hasta otras funciones.

```
lista_ejemplo = [2, 3, 6, 8, 10, 12]
```

len

sort

reverse

append

pop

remove

extend

count

len

index

# If, else, elif....for

- Un montón de cosas en el código sólo deberían ser ejecutadas cuando se cumplan ciertas condiciones. Por eso **Python** tiene algo llamado **sentencias if**.
- En ocasiones, tenemos que repetir varias veces una determinada tarea hasta conseguir nuestro objetivo. En **Python** esto se realiza con el comando **for**.

```
if expression1:
    statement(s)
elif expression2:
    statement(s)
elif expression3:
    statement(s)
else: statement(s)
```

```
for letter in 'Python':      # First Example
    print 'Current Letter :', letter

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:        # Second Example
    print 'Current fruit :', fruit

print "Good bye!"
```

# Diccionarios

- Un **diccionario** es similar a una lista, pero accedes a valores usando una **llave** en vez de un **índice**. Una llave puede ser cualquier cadena o número.



```
diccionario = {'nombre' : 'Carlos', 'edad' : 22, 'cursos': ['Python','Django','JavaScript'] }

print diccionario['nombre'] #Carlos
print diccionario['edad'] #22
print diccionario['cursos'] #['Python','Django','JavaScript']

print diccionario['cursos'][0]#Python
print diccionario['cursos'][1]#Django
print diccionario['cursos'][2]#JavaScript

for key in diccionario:
    print key, ":", diccionario[key]
|
```