

# Bootcamp Inteligencia Artificial

## Nivel Explorador

**Semana 1: Python Para Inteligencia Artificial**

**Variables y Asignaciones**

# Agenda

1. Introducción a Python
2. Entornos de desarrollo en línea
3. Aspectos generales de Python
4. Ejercicios
5. Características de los lenguajes de programación

# 1.1 ¿Qué es Python?

Python es un popular lenguaje de programación. Fue creado por Guido van Rossum y publicado en 1991.

Se utiliza para:

- El desarrollo web (del lado del servidor),
- Desarrollo de software,
- Matemáticas,
- Scripting de sistemas.

## 1.2 ¿Qué puede hacer Python?

- Python puede utilizarse en un servidor para crear aplicaciones web.
- Python se puede utilizar junto con el software para crear flujos de trabajo.
- Python puede conectarse a sistemas de bases de datos. También puede leer y modificar archivos.
- Python puede utilizarse para manejar big data y realizar matemáticas complejas.
- Python puede utilizarse para la creación rápida de prototipos o para el desarrollo de software listo para la producción.

## 1.3 ¿Por qué Python?

- Python funciona en diferentes plataformas (Windows, Mac, Linux, Raspberry Pi, etc).
- Python tiene una sintaxis sencilla similar a la del idioma inglés.
- Python tiene una sintaxis que permite a los desarrolladores escribir programas con menos líneas que otros lenguajes de programación.
- Python se ejecuta en un sistema de interpretación, lo que significa que el código puede ejecutarse tan pronto como se escribe. Esto significa que la creación de prototipos puede ser muy rápida.
- Python puede tratarse de forma procedimental, orientada a objetos o funcional.

## 1.4 Es bueno saber:

- La versión principal más reciente de Python es Python 3. Sin embargo, Python 2, a pesar de que no se actualiza más que las actualizaciones de seguridad, sigue siendo bastante popular.
- En este curso se escribirá Python en una plataforma on line (Colab de Google).
- Es posible escribir Python en un Entorno de Desarrollo Integrado, como Thonny, Pycharm, Netbeans o Eclipse, que son particularmente útiles cuando se manejan grandes colecciones de archivos Python.

## 1.5 Sintaxis

- Python fue diseñado para ser legible, y tiene algunas similitudes con el idioma inglés con influencia de las matemáticas.
- Python utiliza nuevas líneas para completar un comando, a diferencia de otros lenguajes de programación que suelen utilizar punto y coma o paréntesis.
- Python se basa en la sangría del texto (indentation), utilizando espacios en blanco, para definir el ámbito de aplicación, como el de los bucles, las funciones y las clases. Otros lenguajes de programación suelen utilizar corchetes para este fin.

```
print("Hello, World!")
```

## 2.1 Entornos de desarrollo en línea

Un entorno de desarrollo para Python es un ambiente que ofrece herramientas y funcionalidades para facilitar la escritura, depuración y ejecución de código Python. Estos entornos suelen incluir un editor de texto o código fuente, un intérprete de Python, herramientas de depuración, gestión de archivos y a menudo integración con sistemas de control de versiones.





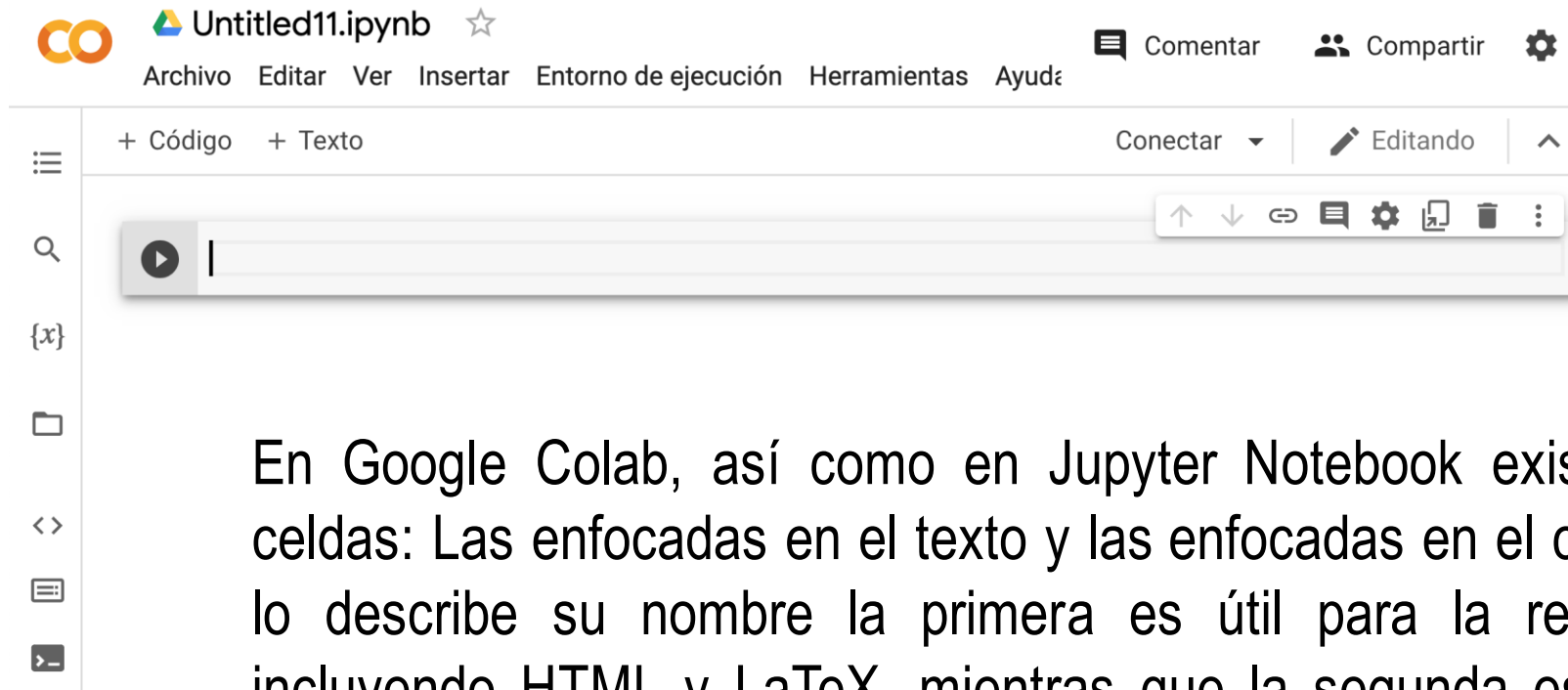
## 2.2 Entornos de desarrollo en línea

Hay varios entornos de desarrollo en línea para Python que permiten a los usuarios escribir, ejecutar y compartir código sin la necesidad de instalar software en su computadora. Algunos de ellos son:

- **Google Colab:** es una plataforma gratuita basada en Jupyter Notebooks que se ejecuta en la nube de Google. Permite escribir y ejecutar código Python, además de integrarse con servicios de almacenamiento de Google. (<https://colab.research.google.com/>)
- **Deepnote:** es un entorno de desarrollo en línea que se centra en la colaboración y la ciencia de datos. Proporciona un entorno de cuaderno similar a Jupyter que se ejecuta en la nube. Está diseñado para ser intuitivo y permite a los usuarios escribir código en Python (u otros lenguajes) en celdas individuales, visualizar datos, crear gráficos y compartir proyectos fácilmente. (<https://deepnote.com/>)
- **Repl.it:** ofrece un entorno en línea para escribir y ejecutar código en Python y otros lenguajes. Proporciona colaboración en tiempo real, integración con GitHub y permite ejecutar proyectos directamente desde el navegador. (<https://replit.com/>)

## 2.3 Aspectos básicos de un notebook en Colab

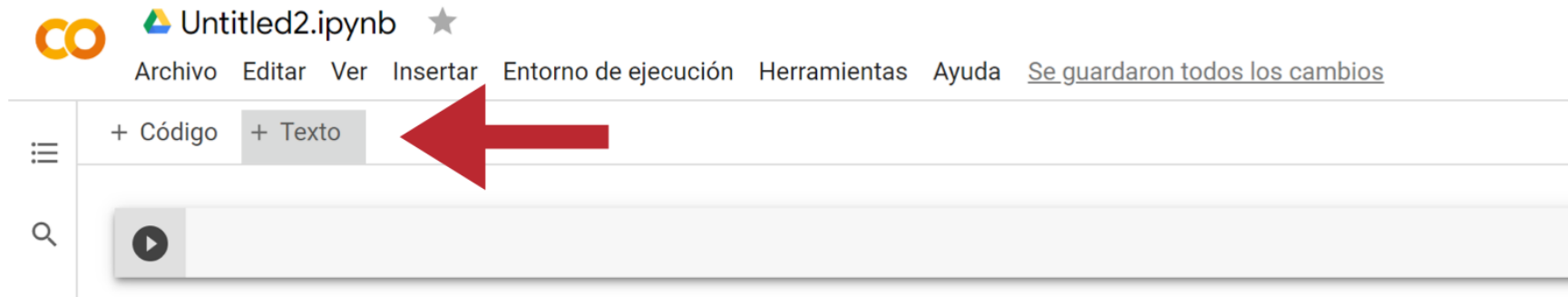
Cuando se inicia un notebook obtendremos una vista como la siguiente:



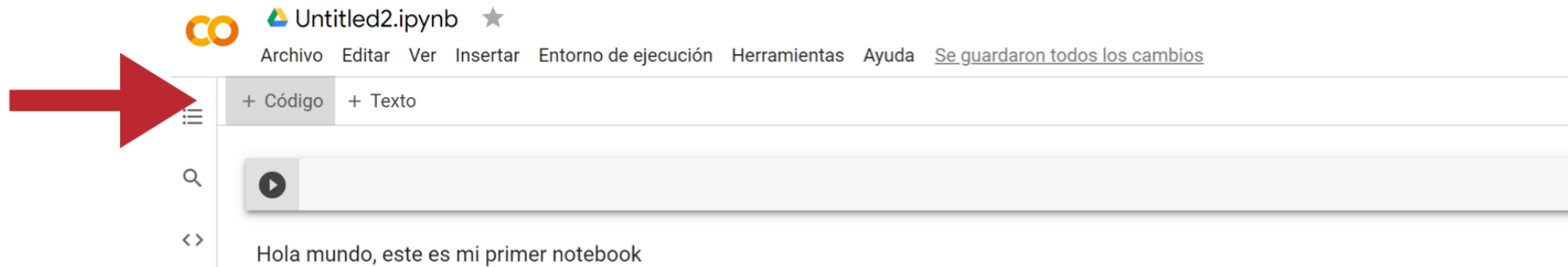
En Google Colab, así como en Jupyter Notebook existen dos tipos de celdas: Las enfocadas en el texto y las enfocadas en el código. Como bien lo describe su nombre la primera es útil para la redacción de texto incluyendo HTML y LaTeX, mientras que la segunda está enfocada a la ejecución de código en el lenguaje de programación Python.

## 2.5 Aspectos básicos de un notebook en Colab

Para agregar una celda de texto será necesario dar clic en el siguiente botón:



Para agregar una celda de código será necesario dar clic en el botón siguiente:



## 2.6 Aspectos básicos de un notebook en Colab

La acción anterior insertará una celda en la cual podrás escribir y ejecutar código, de la siguiente manera:



+ Código + Texto

✓ RAM [ ]  
Disco [ ]

↑ ↓ 🔗

✓ 0 s **▶** `print('Hola Talento Tech 2024!!!')`

Hola Talento Tech 2024!!

## 3.1 Sangría (Indentation) en Python

- La sangría se refiere a los espacios al principio de una línea de código.
- Mientras que en otros lenguajes de programación la sangría en el código es sólo para la legibilidad, la sangría en Python es muy importante.
- Python utiliza la sangría para indicar un bloque de código.



```
if 5 > 2:  
    print("Cinco es mayor a 2!")
```

- Python te dará un error si te saltas la sangría:



```
if 5 > 2:  
print("Cinco es mayor a 2!")
```

```
File "<ipython-input-5-1cf66c84ec30>", line 2  
print("Cinco es mayor a 2!")  
    ^
```

```
IndentationError: expected an indented block
```

## 3.2 Variables en Python

**Una variable puede tener un nombre corto (como x e y) o un nombre más descriptivo (edad, nombre del coche, volumen\_total).**

Reglas para las variables de Python:

- Un nombre de variable debe comenzar con una letra o el caracter de subrayado
- Un nombre de variable no puede empezar por un número
- Un nombre de variable sólo puede contener caracteres alfanuméricos y guiones bajos (A-z, 0-9 y \_ )
- Los nombres de las variables distinguen entre mayúsculas y minúsculas (edad, Age y AGE son tres variables diferentes)

## 3.3 Variables en Python

Muchos valores a múltiples variables

Python permite asignar valores a múltiples variables en una sola línea:



```
x, y, z = "Azul", "Rojo", "Verde"  
print(x)  
print(y)  
print(z)
```

```
Azul  
Rojo  
Verde
```

## 3.4 Variables en Python

Salida de variables

La función `print()` de Python se utiliza a menudo para dar salida a las variables.



```
x = "Python es un lenguaje"  
print(x)
```

Python es un lenguaje



```
x = "Python "  
y = "es un"  
z = " lenguaje"  
print(x+y+z)
```

Python es un lenguaje



```
x = "Python"  
y = "es un"  
z = "lenguaje"  
print(x, y, z)
```

Python es un lenguaje



## 3.5 Tipos de datos en Python

- En programación, el tipo de datos es un concepto importante.
- Las variables pueden almacenar datos de diferentes tipos, y diferentes tipos pueden hacer diferentes cosas.
- Python tiene los siguientes tipos de datos incorporados por defecto, en estas categorías:

Tipos de texto

Tipos numéricos

Tipos de secuencia

Tipo de mapeo

Tipos de conjuntos

Tipo booleano

Tipos binarios

`str`

`int`, `float`, `complex`

`list`, `tuple`, `range`

`dict`

`set`, `frozenset`

`bool`

`bytes`, `bytearray`, `memoryview`

## 3.6 Tipos de datos en Python

A continuación, se describen algunos tipos de datos que se manejan en Python:

```
#Tipos de datos  
a=1  
b=1.2  
c='Hola'  
d=True  
e=[1,2,3,4,5]
```

```
print(a, type(a))  
print(b, type(b))  
print(c, type(c))  
print(d, type(d))  
print(e, type(e))
```

```
1 <class 'int'>  
1.2 <class 'float'>  
Hola <class 'str'>  
True <class 'bool'>  
[1, 2, 3, 4, 5] <class 'list'>
```

## 3.7 Números en Python

A continuación, se describen algunos tipos de datos que se manejan en Python:

```
x = 1      # int
y = 2.8    # float
z = 1j     # complex
print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```



```
x = 35e3
y = 12E4
z = -87.7e100
```

```
print(type(x))
print(type(y))
print(type(z))
```



```
<class 'float'>
<class 'float'>
<class 'float'>
```

```
x = 3+5j
y = 5j
z = -5j
print(type(x))
print(type(y))
print(type(z))
```

```
<class 'complex'>
<class 'complex'>
<class 'complex'>
```

## 3.8 Números en Python

### Conversión de tipo

```

▶ x = 1      # int
  y = 2.8    # float
  z = 1j     # complex
  #convertir a float:
  a = float(x)
  #convertir de float a int:
  b = int(y)
  #convertir de int a complejo:
  c = complex(x)

```

```

▶ print(a)
  print(b)
  print(c)
  print(type(a))
  print(type(b))
  print(type(c))

```

```

☞ 1.0
   2
   (1+0j)
   <class 'float'>
   <class 'int'>
   <class 'complex'>

```

## 3.9 Booleanos en Python

- En programación a menudo necesitas saber si una expresión es Verdadera o Falsa.
- Puedes evaluar cualquier expresión en Python, y obtener una de las dos respuestas, Verdadero o Falso.
- Cuando comparas dos valores, la expresión se evalúa y Python devuelve la respuesta booleana:

```
▶ print(10 > 9)  
print(10 == 9)  
print(10 < 9)
```

```
True  
False  
False
```

```
▶ print(bool("Hola"))  
print(bool(15))
```

```
True  
True
```

```
▶ x = "Hello"  
y = 15  
  
print(bool(x))  
print(bool(y))
```

```
☐➔ True  
True
```

## 4.0 Ejercicios

- Crear en Colab y evaluar los tipos de datos que tiene Python.
- Crear en Deep Note un programa que imprima por pantalla mensajes que utilicen todos los símbolos de Python.
- Crear en Repl. it un programa que evalúe las compuertas AND y OR y NOT.
- Crear en Colab un programa que sume dos números complejos.

## 5.1 Características lenguajes de programación

Todo lenguaje de programación está compuesto por un alfabeto, un vocabulario y una gramática.

- Alfabeto o conjunto de caracteres: es el conjunto de elementos estructurales del lenguaje como:
  - Caracteres alfabéticos (letras minúsculas y mayúsculas)
  - Caracteres numéricos (dígitos del 0 al 9)
  - Caracteres especiales (símbolos especiales tales como [.), [#], [\$], [/] y muchos otros

## 5.2 Características lenguajes de programación

- Vocabulario o léxico: Es el conjunto de palabras válidas o reservadas en un lenguaje. Por ejemplo, *float*, *int*, *else*, *char*, *while*, cada una de ellas tiene un significado predeterminado en el lenguaje, es decir, son palabras reservadas.
- Gramática: es el conjunto de lineamientos que se deben seguir para construir las instrucciones.



## 5.3 Características lenguajes de programación

- La lista de instrucciones es, de hecho, el alfabeto de un lenguaje máquina.
- Necesitamos un lenguaje en el que los humanos puedan escribir sus programas y un lenguaje que las computadoras puedan usar para ejecutar los programas, tales lenguajes son a menudo llamados lenguajes de programación de alto nivel.
- Un programa escrito en un lenguaje de programación de alto nivel se llama código fuente.

## 5.4 Características lenguajes de programación




Un programa bien hecho debe tener ciertas características que le permitan operar correctamente como:

- Operatividad
- Legibilidad
- Transportabilidad
- Claridad
- Modularidad

## 5.5 Características lenguajes de programación

Los programas deben ser correctos en amplios sentidos:

- 
- Alfabéticamente
  - Léxico
  - Sintácticamente
  - Semánticamente

Cuando escribimos un programa, en el lenguaje que estemos utilizando, la máquina se encargará de convertir ese programa a lenguaje de máquina, de forma fácil y rápida. Existen dos formas de hacerlo: **la compilación y la interpretación.**

## 5.5 Compilación

En la compilación, el programa que nosotros hacemos, debe ser traducido una vez a lenguaje de máquina y ese trabajo lo realiza el compilador. El proceso de compilación es el siguiente:

❖ Una vez que tenemos diseñado el programa, debe ser introducido mediante el proceso de edición, para lo cual se utiliza un editor que nos permiten crear un archivo en el cual introducimos el programa, creándose un programa fuente con las instrucciones que nosotros elaboramos en el lenguaje que estemos utilizando en ese momento. El programa fuente es sometido al proceso compilación.

## 5.6 Interpretación

Aquí se traduce el programa fuente cada vez que se ejecuta; el programa que realiza este tipo de transformación se denomina intérprete, Cuando se escribe un programa, en realidad lo que tenemos es un archivo de texto; este texto, no debe incluir formatos, porque es necesario invocar al intérprete y dejar que lea el archivo fuente.

## 5.7 Comparación

	Compilación	Interpretación
<b>Ventajas</b>	<p>La ejecución del código traducido suele ser más rápida.</p> <p>Sólo el usuario debe tener el compilador; el usuario final puede usar el código sin él.</p> <p>El código traducido se almacena en lenguaje máquina, ya que es muy difícil de entender, es probable que tus propios inventos y trucos de programación sigan siendo secreto.</p>	<p>Puede ejecutar el código en cuanto lo complete; no hay fases adicionales de traducción.</p> <p>El código se almacena utilizando el lenguaje de programación, no el de la máquina; esto significa que puede ejecutarse en computadoras que utilizan diferentes lenguajes máquina; no compila el código por separado para cada arquitectura diferente.</p>

## 5.8 Identificadores

- Un identificador es una secuencia de símbolos que se utilizan como nombres de variables, funciones, arreglos, clases y otras estructuras de los lenguajes de programación.
- Un identificar válido debe cumplir con la condición adicional de que no pertenezca a las palabras reservadas para el lenguaje

## 5.9 Palabras reservadas

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



# Preguntas

