Unidad 1

Introducción a Python

1. Introducción

Python es un lenguaje de programación de alto nivel (cercano al lenguaje humano) que ha ganado mucha popularidad desde su creación. Fue desarrollado por Guido van Rossum a finales de los años 80 y lanzado oficialmente en 1991. Van Rossum buscaba un lenguaje que fuera fácil de aprender, legible y versátil, combinando sencillez y claridad con potencia y flexibilidad. Python se diseñó con una sintaxis clara y sencilla, lo que facilita el proceso de aprendizaje y desarrollo.

El nombre “Python” proviene del programa de comedia británico "Monty Python’s Flying Circus", Van Rossum era un fan del programa y quería que el lenguaje fuese divertido de usar, así que optó por un nombre que reflejara esta filosofía de accesibilidad y facilidad.

1.1 Principales características de Python

1. **Simplicidad y legibilidad**: Python es conocido por su sintaxis simple y fácil de leer, lo cual permite a los desarrolladores escribir código más comprensible y rápido de desarrollar. Esto hace que el lenguaje sea ideal tanto para principiantes como para programadores experimentados.
2. **Multiparadigma**: Python soporta varios paradigmas de programación, incluyendo la programación orientada a objetos, la programación funcional y la programación imperativa. Esta flexibilidad permite a los desarrolladores elegir el estilo que mejor se adapte a sus necesidades.
3. **Amplia biblioteca estándar**: Python cuenta con una biblioteca estándar extensa, conocida como “Baterías incluidas” (*Batteries included*), que proporciona módulos y paquetes para realizar tareas comunes como trabajar con archivos, hacer conexiones de red, manejar datos en formatos JSON y XML, realizar cálculos matemáticos avanzados, entre otros.
4. **Gran comunidad y soporte**: Al ser uno de los lenguajes más populares, Python cuenta con una comunidad activa que contribuye a su desarrollo y mantiene una gran cantidad de documentación, tutoriales y herramientas. Además, su código es de código abierto (*open-source*), lo que permite a los usuarios contribuir y mejorar el lenguaje.
5. **Portabilidad y compatibilidad multiplataforma**: Python es compatible con diversos sistemas operativos como Windows, macOS y Linux. Además, gracias a herramientas como virtualenv, es posible crear entornos virtuales que permiten aislar dependencias específicas de cada proyecto, facilitando el desarrollo y la implementación en diferentes entornos.
6. **Interpretado**: Python es un lenguaje interpretado, lo que significa que el código se ejecuta línea por línea sin necesidad de compilación previa. Esto facilita el

desarrollo rápido y permite a los desarrolladores probar y depurar código con facilidad. Esta característica es una de las grandes diferencias para la elección del uso de Python frente a otros posibles lenguajes compilados. Al no necesitar una compilación tiene un ciclo de desarrollo más rápido, por su ejecución directa. Esto sin embargo también trae desventajas, como la necesidad de un intérprete para la ejecución, o el rendimiento más lento respecto a otros lenguajes. A continuación podemos ver una tabla con las principales diferencias:

Resumen de las diferencias clave

Característica	Lenguajes Compilados	Lenguajes Interpretados
Ejecución	Se compilan a código máquina antes de ejecutar	Se interpretan línea por línea en tiempo real
Rendimiento	Generalmente más rápido	Generalmente más lento
Flexibilidad	Menos flexibles en tiempo de ejecución	Más flexibles en tiempo de ejecución
Ciclo de desarrollo	Más lento debido a la compilación	Más rápido por la ejecución directa
Dependencia	No requiere un intérprete en ejecución	Requiere un intérprete en ejecución

1.2 Evolución y versiones de Python

Desde su lanzamiento, Python ha pasado por varias versiones importantes. La versión 2 (Python 2.0) introdujo características fundamentales como el recolector de basura y la compatibilidad Unicode (Estándar de condificación de caracteres). Sin embargo, en 2008 se lanzó Python 3, una versión mayormente incompatible con Python 2 pero diseñada para mejorar el rendimiento, la simplicidad y la eficiencia del lenguaje. La transición fue significativa y duró varios años, pero hoy en día Python 3 es el estándar.

A continuación podemos ver algunas de las principales diferencias, que se entenderán más adelante cuando demos los diferentes tipos de datos, funciones, etc

Resumen de Diferencias Clave

Característica	Python 2	Python 3
Soporte	Sin soporte desde 2020	Activo y actualizado
Print	Declaración sin paréntesis	Función con paréntesis
División de enteros	Redondeo hacia abajo (entero)	División verdadera (punto flotante)
Cadenas de texto	str son bytes, uso de u"" para Unicode	str son Unicode por defecto
Funciones de iterador	range() devuelve lista	range() devuelve un iterador

Característica	Python 2	Python 3
Módulo de entrada/salida	raw_input() para texto	input() solo retorna texto
Manejo de excepciones	except Exception, e:	except Exception as e:
Comprensiones	Limitadas	Comprensiones de conjunto y diccionario
Mejoras y módulos adicionales	Limitados	Nuevas características y módulos

1.3 Usos y aplicaciones de Python

La versatilidad de Python lo ha convertido en el lenguaje preferido para una gran variedad de aplicaciones, incluyendo:

- **Desarrollo web**: Con frameworks como Django y Flask, Python permite construir aplicaciones web robustas y seguras de manera eficiente.
- **Ciencia de datos y aprendizaje automático**: Librerías como Pandas, NumPy, Scikit-learn y TensorFlow han consolidado a Python como el lenguaje principal en el campo de la ciencia de datos y la inteligencia artificial.
- **Automatización y scripts**: Python es una excelente opción para escribir scripts de automatización, ya que permite realizar tareas repetitivas y gestionar procesos de forma sencilla.
- **Desarrollo de videojuegos**: Herramientas como Pygame permiten desarrollar videojuegos sencillos.
- **Aplicaciones de escritorio**: Python se utiliza también para el desarrollo de aplicaciones de escritorio gracias a herramientas como Tkinter y PyQt.

En conclusión, Python ha evolucionado desde un lenguaje pequeño a ser uno de los lenguajes de programación más influyentes y utilizados a nivel mundial, gracias a su simplicidad, flexibilidad y un vasto ecosistema de librerías y herramientas.

2. Primeros pasos

Tras la instalación de Python (URL en el curso) podemos comenzar a probar nuestras primeras instrucciones, ya que cuenta con un IDLE (Entorno de desarrollo integrado). Es una ventana de shell (consola).

Desde una ventana de comandos podemos comenzar con el comando “python”.

```
C:\Users\      >python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

A partir de aquí, las líneas de comandos comenzarán con >>>.

>>> print(“Hola Mundo”)

Con el comando print() imprimiremos por pantalla aquello que le pasemos, en este caso la *string* Hola Mundo.

En python NO es necesario acabar la línea con “;” como sí lo es en otros lenguajes. Esto no quiere decir que no se pueda usar, ya que lo podemos utilizar como recurso para escribir varias instrucciones en la misma línea

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

Pero se desaconseja su uso, ya que dificulta la legibilidad del código, que es un aspecto fundamental en el trabajo de un buen programador. Es preferible usar múltiples líneas. También es posible dividir una instrucción en varias líneas con el uso de “\”

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

Pero al igual que lo anterior, se desaconseja su uso.

Python es un lenguaje indentado (tabulado). Esto quiere decir que las tabulaciones (espacios en blanco) forman parte de su sintaxis. Es muy importante para los bloques de códigos.

```
[1]: for i in range (5):
         print(i)
     print("Fin")
```

```
0
1
2
3
4
Fin
```

Como se puede ver, el código no utiliza llaves { } ni puntos y comas ; sino que entiende la tabulación (el espacio hacia dentro) es lo que marca el contenido del bucle *for*, por lo que completa la impresión por pantalla de los números del 0 al 4 antes imprimir *Fin*, que se encuentra sin tabular (es decir, fuera del bloque *for*). Python recomienda el uso de 4 espacios en blanco (vale con pulsar una vez la tecla Tab ↹).

Para crear variables bastará con nombrarla y asignarle el valor.

```
>>> nombre = "Juan"
>>> print("Mi amigo se llama " + nombre)
Mi amigo se llama Juan
```

En Python no es necesario declarar el tipo de dato de cada variable, ya que es un lenguaje de tipado dinámico. Esto quiere decir que el tipo de variable se asigna según el valor que contenga. Este tipo se asigna en tiempo de ejecución. Aunque no sea necesario, también se pueden utilizar anotaciones que indiquen el tipo de dato.

Por último, si deseamos escribir un comentario (imprescindible para poder explicar bloques de código cuando estemos trabajando), en Python se escriben con el carácter #

```
#Creamos una variable con el valor "Hola Mundo"

saludo = "Hola Mundo"
```