



Introducción a Python

Conociendo Python (parte II)

***Reconocer los conceptos
fundamentales del lenguaje
Python y distinguir los tipos
de datos y sentencias para
la construcción de
programas.***

- Unidad 1:
Introducción a Python
- Unidad 2:
Sentencias condicionales e
iterativas
- Unidad 3:
Estructuras de datos y funciones



Te encuentras aquí



¿Qué aprenderás en esta sesión?

- *Reconoce el entorno de ejecución y las herramientas complementarias de Python para el desarrollo.*

¿Cuáles son las
principales áreas en las
que se utiliza Python?



/* Entorno de ejecución */

Python y sus múltiples versiones

- En este módulo trabajaremos con una versión de Python de 3.7 o superior.
- Una de las características más importantes de esta versión es que no es compatible con versiones anteriores como la 2.x.
- Además desde 2020 Python 2 dejó de tener soporte por lo que la razón de su existencia es sólo para desarrollos “legacy” que fueron diseñados en esa versión.

¿Cómo podemos manejar distintas versiones de Python?

Una alternativa es desarrollar ambientes virtuales, los cuales funcionan como un contenedor aislado donde se puede declarar una versión específica de Python y librerías específicas. Mediante éstos, nos aseguramos de que cualquier problema existente en el ambiente virtual no afecte a la máquina en su totalidad.

Para trabajar con una versión específica de Python, se debe trabajar con el administrador de versiones adecuado. Existen varios administradores de versiones de Python, como pyenv, virtualenv y Vnaconda. Para efectos prácticos, nosotros trabajaremos con Anaconda.

Anaconda

¿Qué es?

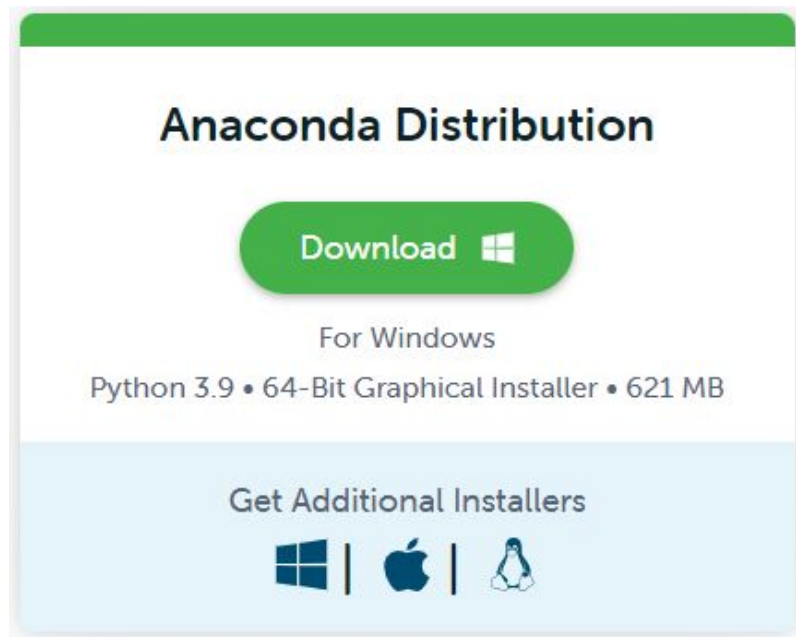
- Administrador orientado al trabajo en ciencia de datos.
- Ofrece la ventaja de que incluye muchas librerías preinstaladas lo que facilita el trabajo para alguien que se aproxima a Python por primera vez.
- Incluye de manera preinstalada algunos editores de texto como Jupyter, Spyder, VSCode y Pycharm.



ANACONDA®

Instalación

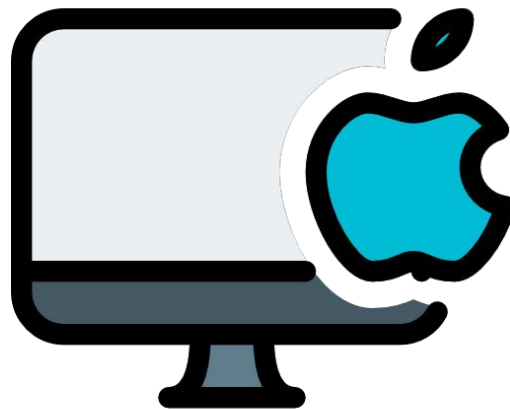
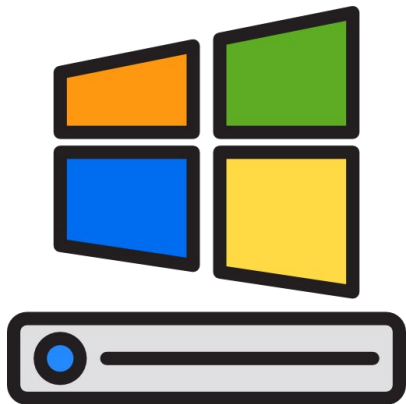
Ingresa al [siguiente enlace](#) para realizar la descarga e instalación.



Instalación

Windows -Mac

La instalación en estos sistemas será sencilla, basada en la selección de un directorio y presionar “Siguiente” hasta finalizar la instalación.



Instalación

Linux

Una vez descargado el archivo de instalación, ejecutamos los siguientes códigos en una terminal (asumimos que el archivo se encuentra en el Directorio Descargas).

```
cd ~/Descargas
```

```
bash Anaconda3-<versión del instalador>-Linux-x86_64.sh
```

¿Para qué nos sirve
Anaconda?



/* La consola de comandos*/

La consola de comandos

Abriremos una terminal y simplemente escribimos **python** y presionamos **enter**. Esto nos llevará a la consola de comandos de python, en la que podemos utilizar comandos en tiempo real.

```
(base) C:\Users\TuUsuario>python  
>>>
```

/* Primeras instrucciones */

print()

- Es una función para mostrar valores de variables o resultados de operaciones en la pantalla.
- Podemos entender inicialmente una función como un comando que Python ejecutará.

"**hola mundo**" es el argumento de la función print, y Python mostrará dicho valor en pantalla.

Asegúrate de utilizar print para ver en pantalla solo los resultados que te interesan.

Ejercicio guiado



Mi primer código en Python

Crearemos nuestro primer programa con el fin de entender cómo crear y ejecutar scripts en Python:

```
# Este código imprime el famoso hola mundo  
print("hola mundo")
```



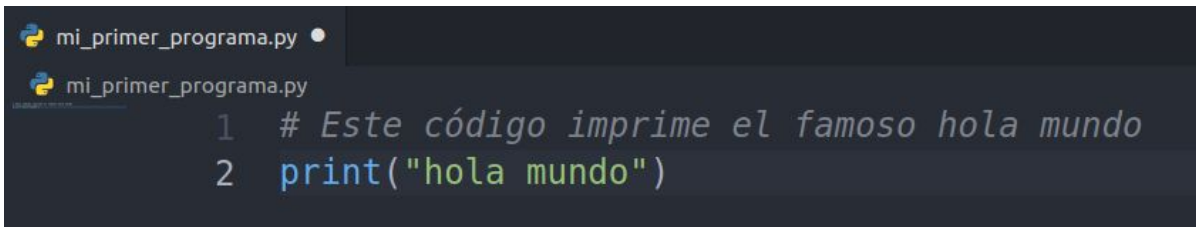
Mi primer código en Python

Solución

Paso 1: Abrir nuestro editor de texto.

Paso 2: Escribir una instrucción sencilla. Ej. `print("hola mundo")`

Paso 3: Guardar el archivo con extensión `.py`.



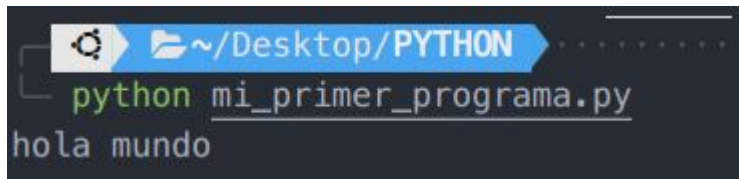
```
mi_primer_programa.py •  
mi_primer_programa.py  
1 # Este código imprime el famoso hola mundo  
2 print("hola mundo")
```



Mi primer código en Python

Solución

Paso 4: Abrir un terminal en la misma ubicación donde se encuentra el archivo y ejecutar python ***mi_primer_programa.py***

A screenshot of a terminal window with a dark background. The title bar at the top shows a gear icon, a folder icon, and the path ~/Desktop/PYTHON. The terminal content shows the command 'python mi_primer_programa.py' being executed, with the output 'hola mundo' displayed on the line below.

```
~/Desktop/PYTHON  
python mi_primer_programa.py  
hola mundo
```

*Para el caso particular de este script se creó una carpeta llamada **/python** en el Escritorio.*

.py y .pyw *¿Cuál escogemos?*

En ocasiones, podemos querer que un archivo se ejecute “directamente”, es decir, sin pasar por una línea de comandos de la terminal. En este caso, utilizaremos **.pyw**

Si queremos pasar por la línea de comandos, guardamos simplemente como **.py**

/* Operaciones aritméticas básicas */

Python como una calculadora

La funcionalidad más simple que se le puede dar a Python es la de una calculadora. Por defecto, Python **incluye operaciones básicas** como las sumas, restas, multiplicaciones, divisiones, entre otros.

Algunos ejemplos:

SUMA

```
print(2 + 2)  
2
```

RESTA

```
print(6 - 3)  
3
```

MULTIPLICACIÓN

```
print(7 * 8)  
56
```

DIVISIÓN

```
print(90 / 10)  
10
```

Limitantes

Python no permite sumar letras y números.

Cada vez que Python no sea capaz de ejecutar una instrucción que no está permitida, se mostrará algo así:

```
In [2]: "gato" + 2
-----
TypeError                                Traceback (most recent call last)
<ipython-input-2-37334b11ca62> in <module>
----> 1 "gato" + 2

TypeError: must be str, not int
In [3]:
```

El error que acabamos de ver ocurre porque no entendemos uno de los primeros aspectos a estudiar: los tipos de datos.

print() es sumamente importante para visualizar el resultado de la operación realizada; en caso de no utilizarlo los resultados no aparecerán en la pantalla.



¿Cuáles son las limitantes?



`/* Comentarios */`

Comentarios

Comentario en una sola línea

Todo texto a continuación de un signo # es un comentario.

```
# Esta línea es un comentario
# Los comentarios son ignorados
# Pueden ser una línea nueva
print(2 + 2) # 0 puede acompañar una línea de
código existente
# print(2 + 3)
# Si comentamos un código válido en Python, no
se ejecutará
```

4

{desafío}
latam_

Comentarios en múltiples líneas

Consiste en envolver todo el comentario entre tres comillas dobles al inicio y al final.

```
"""
Comentario multilínea:
Python
lo
ignoraré
"""

print("hola")
```



Próxima sesión...

- *Reconoce los operadores básicos de Python*
- *Comprende el uso de librerías.*

{desafío}
latam_

*Academia de
talentos digitales*

