

Autor: Harol Alvarado  
Correo: harolav3@gmail.com

## 1. Introducción al curso.

En este curso aprenderemos diferentes temas correspondientes al lenguaje de programación python desde lo más básico, para poder empezar necesitamos antes aclarar algunos términos necesarios que muchas veces confunden al estudiante, así que vamos a ello.

### 1.1. Lenguaje de Programación

#### 1.1.1. ¿Qué es un Lenguaje de Programación?

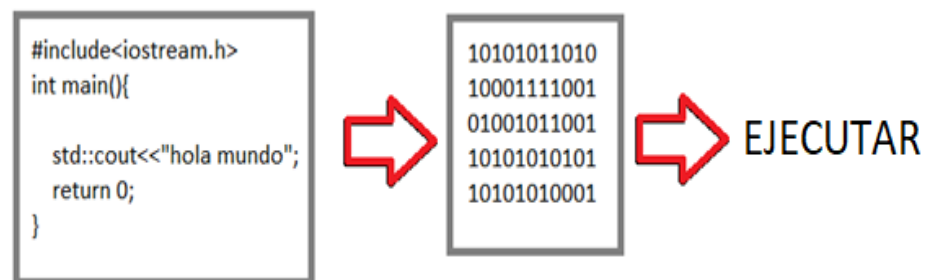
Es un lenguaje formal que nos da la capacidad de poder controlar el comportamiento de una computadora mediante una serie de instrucciones, algoritmos.

#### 1.1.2. Clasificación de un lenguaje de programación según el Nivel:

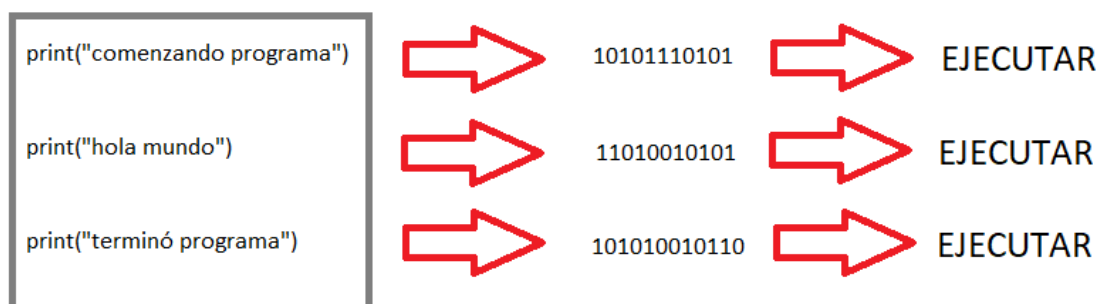
- Lenguaje de Nivel Bajo: son aquellos lenguajes que utilizan el lenguaje máquina es decir 0s y 1s.
- Lenguaje de Medio Nivel: Son aquellos lenguajes que, si bien no utilizan el lenguaje máquina, no son tan entendibles para la persona. Como por ejemplo el lenguaje Assembler
- Lenguaje de Alto Nivel: Son aquellos lenguajes que se asemejan al lenguaje humano, lo que permite un mayor entendimiento para el programador. Como por ejemplo: Python, c#, java.

#### 1.1.3. Clasificación de un lenguaje de programación según la ejecución:

- Lenguajes Compilados: uso de compilador



- Lenguajes Interpretados: uso de interprete



## 1.2. Ambientes para programar:

### 1.2.1. Editor de Código:



### 1.2.2. Entorno de Desarrollo Integrado:

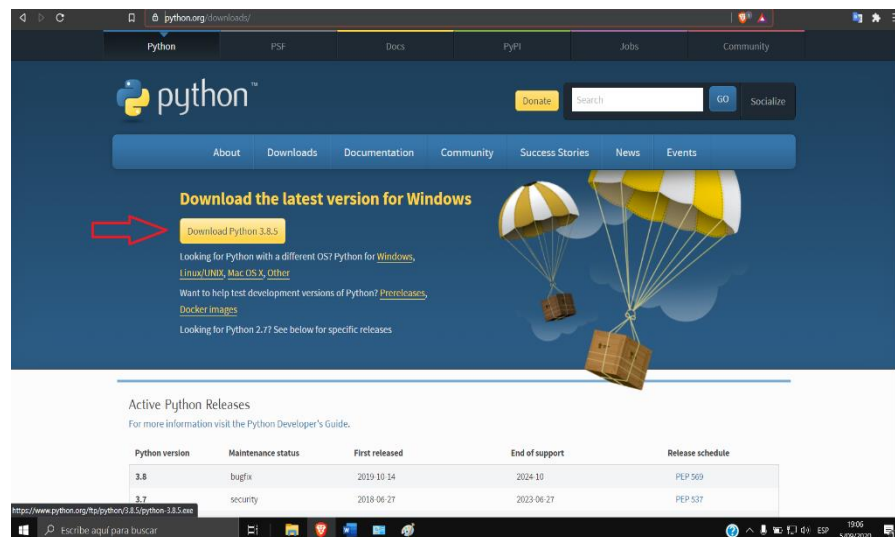


## 2. Instalación.

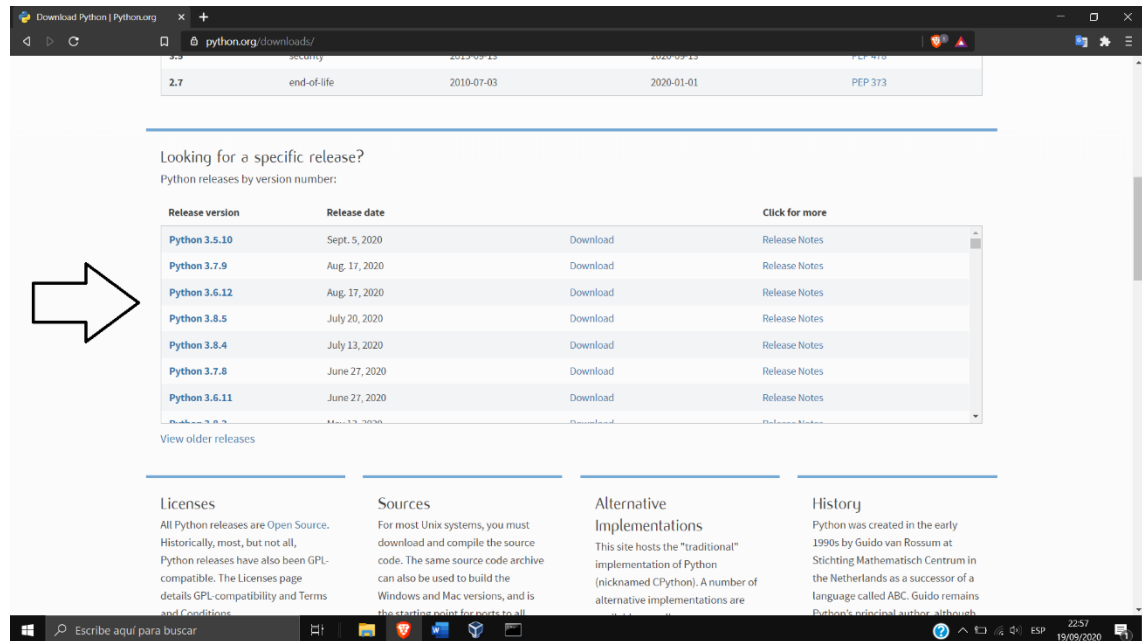
2.1. Para poder realizar la instalar Python en el sistema operativo que tengamos, nos dirigimos a la siguiente página web oficial:

<https://www.python.org/downloads/>

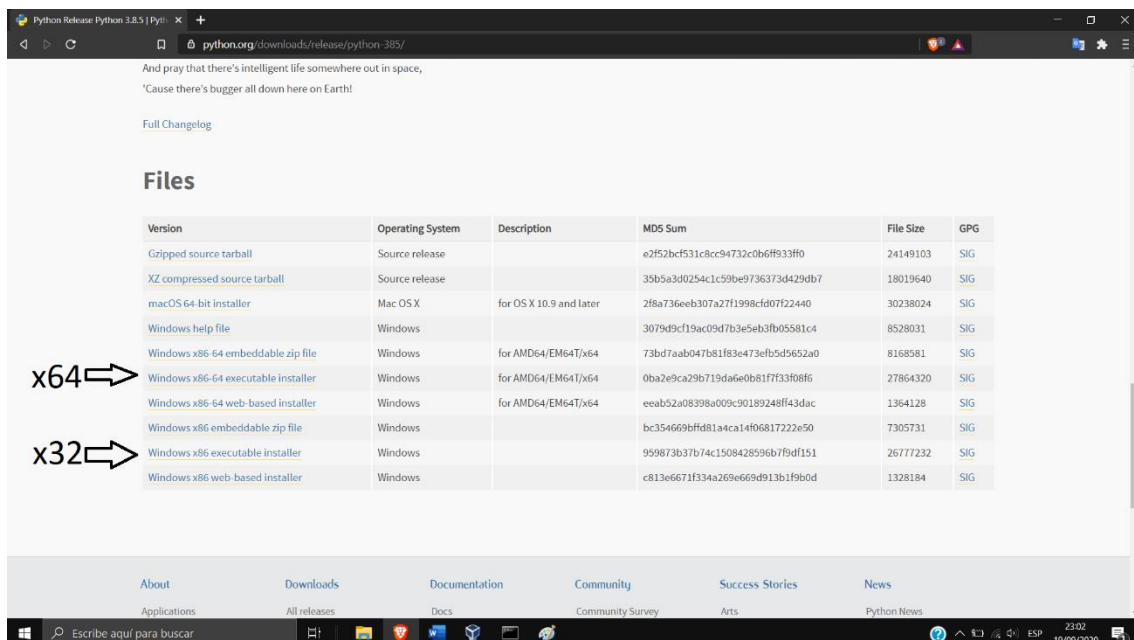
Descargaremos la versión que deseamos utilizar, en este caso utilizaremos la versión 3.8.5



2.2. Si deseamos descargar otra versión solo vamos a <https://www.python.org/downloads/> y elegimos la versión que deseamos

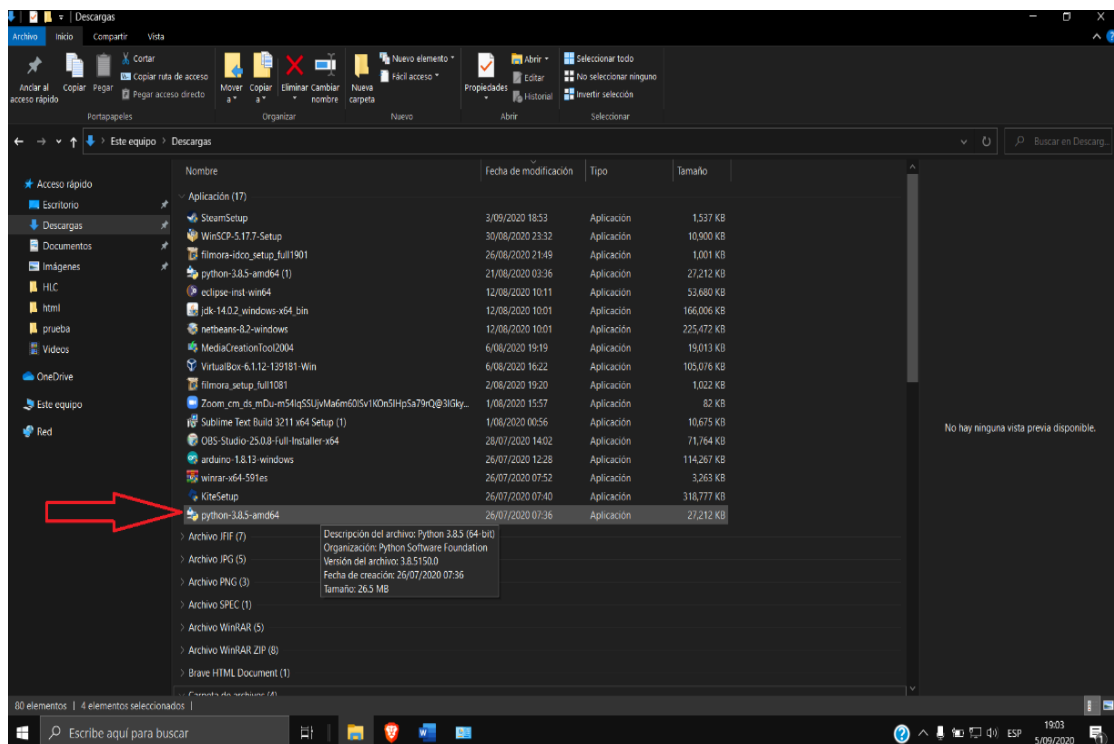


2.3. En este caso la versión 3.8.5, al hacer click aquí nos redirigirá al siguiente a la siguiente página; <https://www.python.org/downloads/release/python-385/> iremos a la sección de Files y elegiremos el instalador de acuerdo a la arquitectura que maneje nuestra computadora ya sea x64 bits o x32 bits

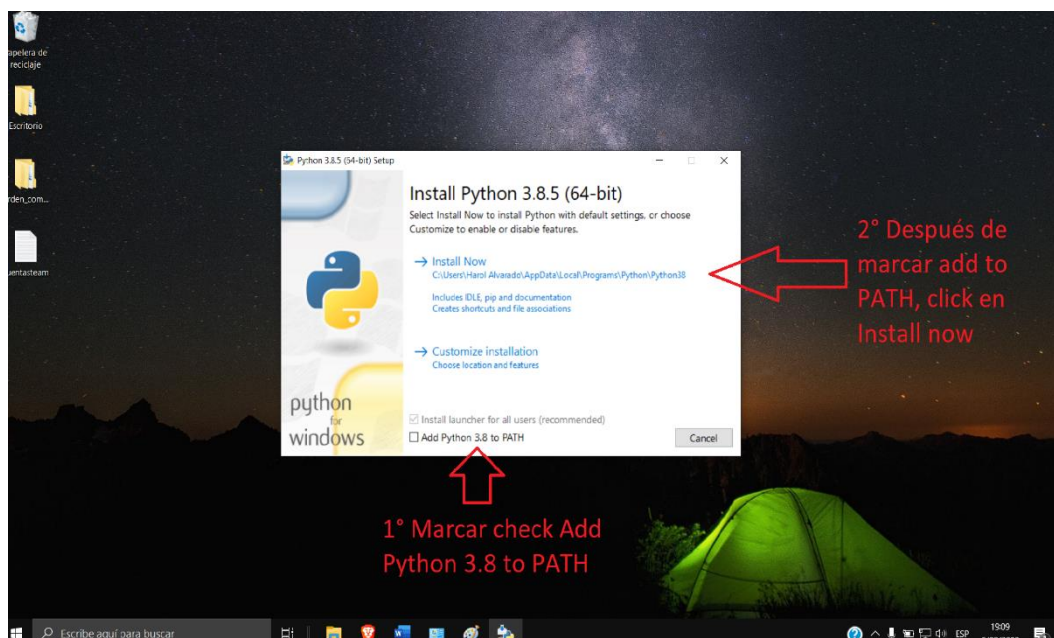


2.4. También puede obviar los pasos 2.2 y 2.3 y solo ejecutar el paso 2.1 y dirigirse directamente al paso 2.5

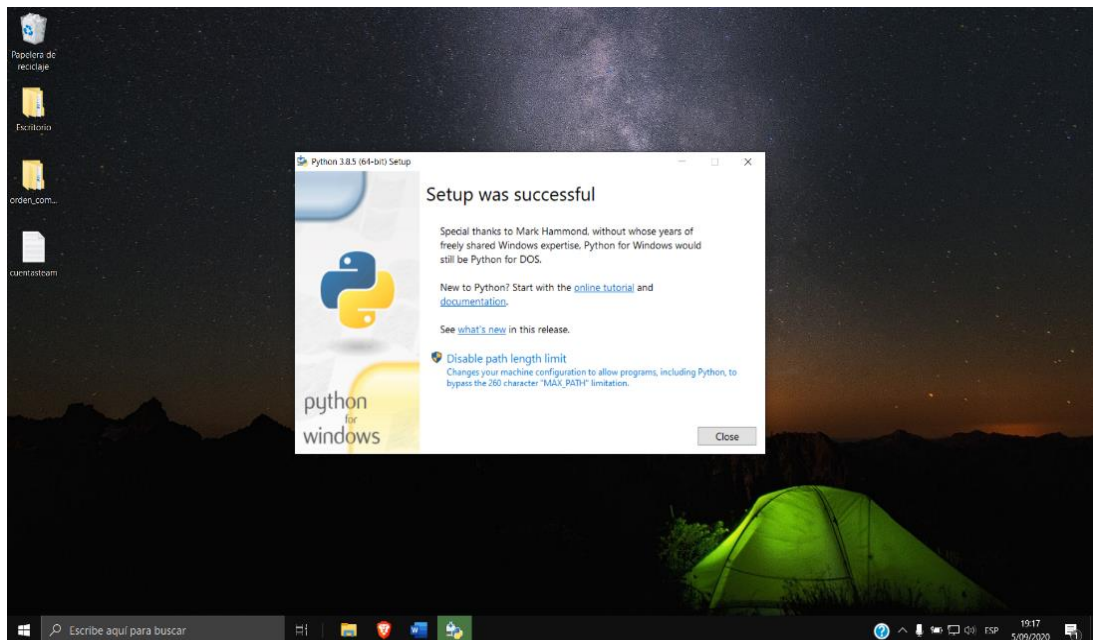
## 2.5. Ejecutamos el instalador



## 2.6. Nos aseguramos marcar el check de “Add to path” y posteriormente le damos click en “Install now”



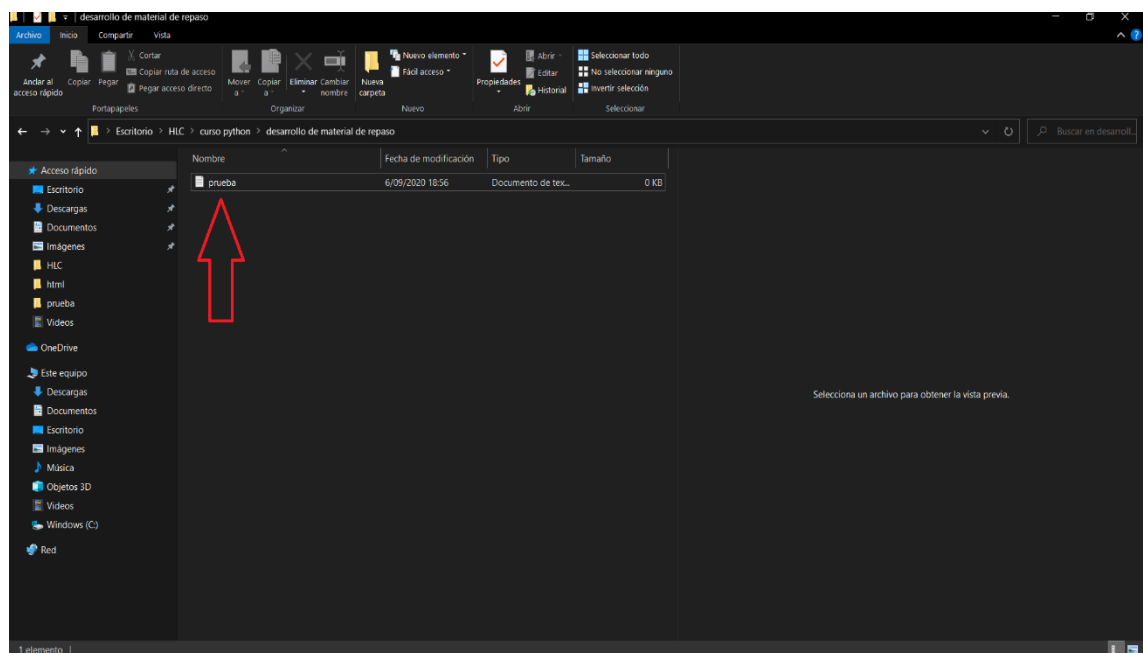
2.7. Al terminar el proceso habremos instalado satisfactoriamente Python.



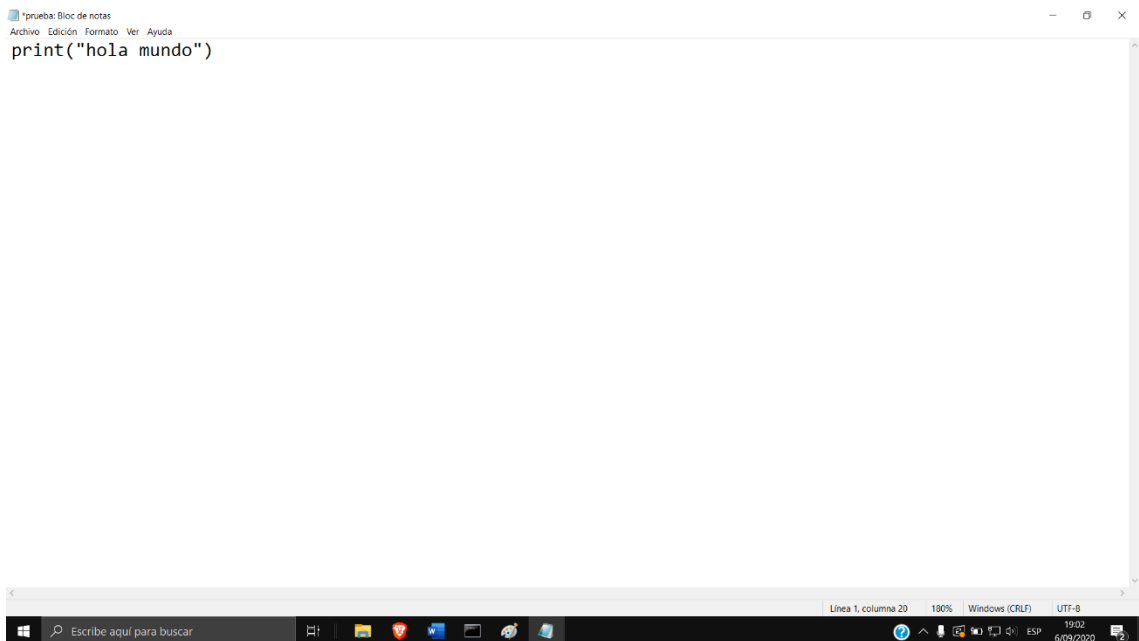
### 3. Creación de un script de Python:

Al comienzo del curso, en las primeras pruebas de código utilizaremos el bloc de notas que nos permitirá entender como crear y como ejecutar un script de Python:

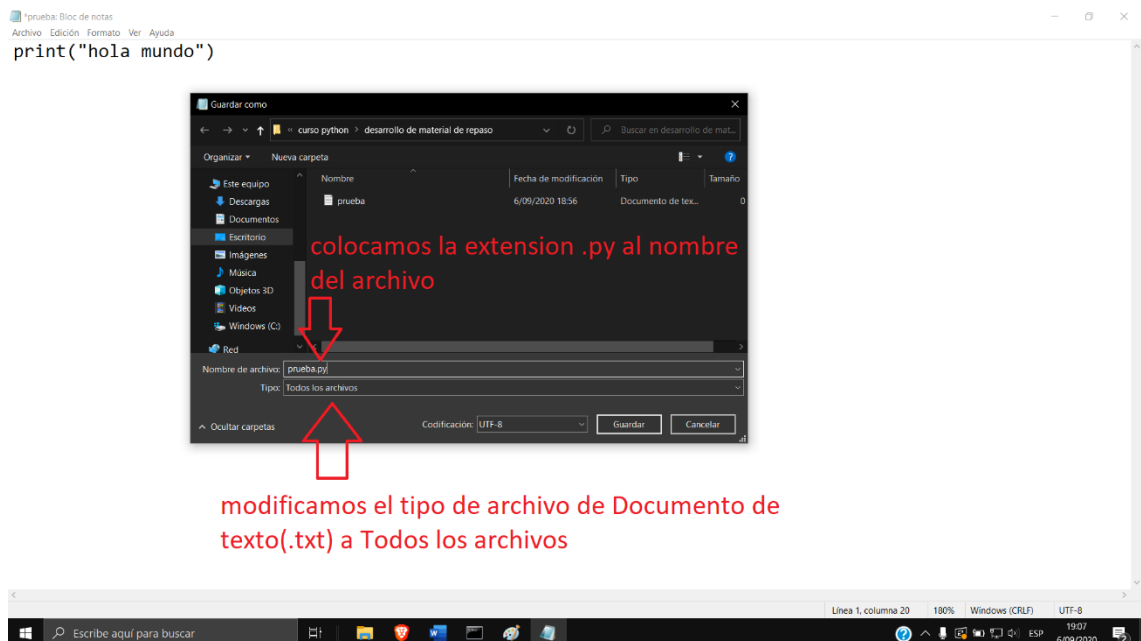
3.1. Creamos un archivo de bloc de notas y lo guardamos en la carpeta que deseamos con el nombre que elegimos, en este caso nuestro archivo de bloc de notas(.txt) se llamará prueba



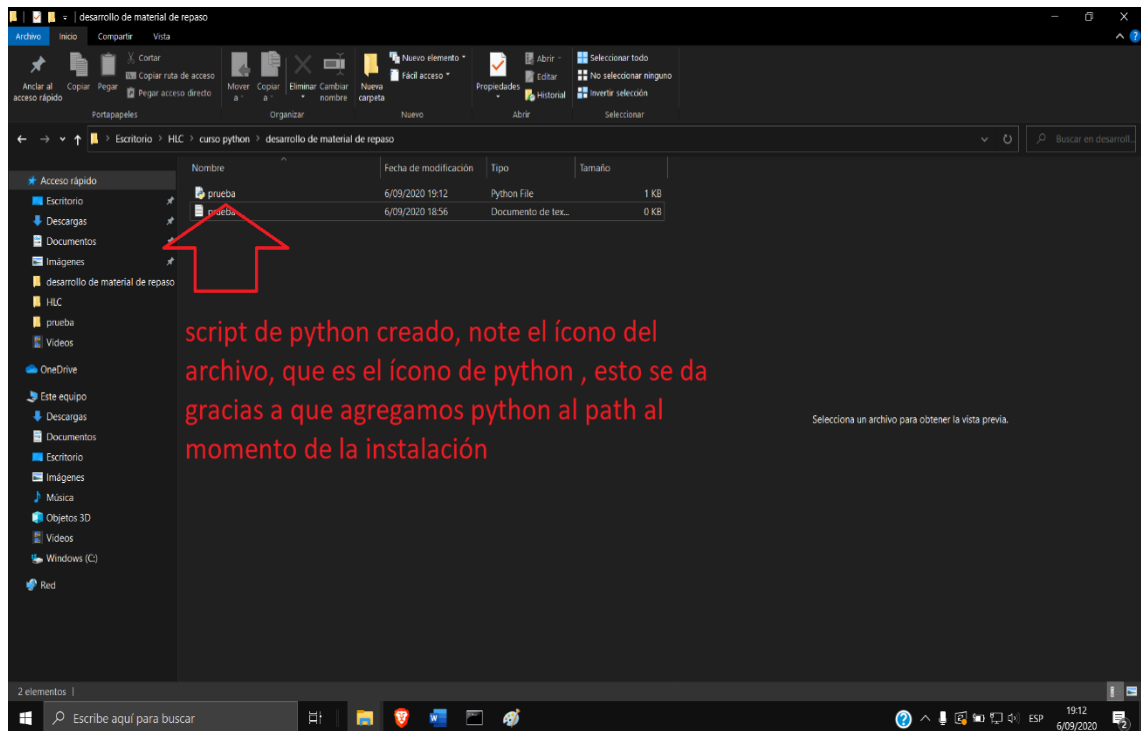
3.2. Ahora abrimos ese documento y escribimos `print("hola mundo")`, cuando ejecutemos este script gracias a este comando que estamos escribiendo podremos mostrar el mensaje hola mundo en la consola



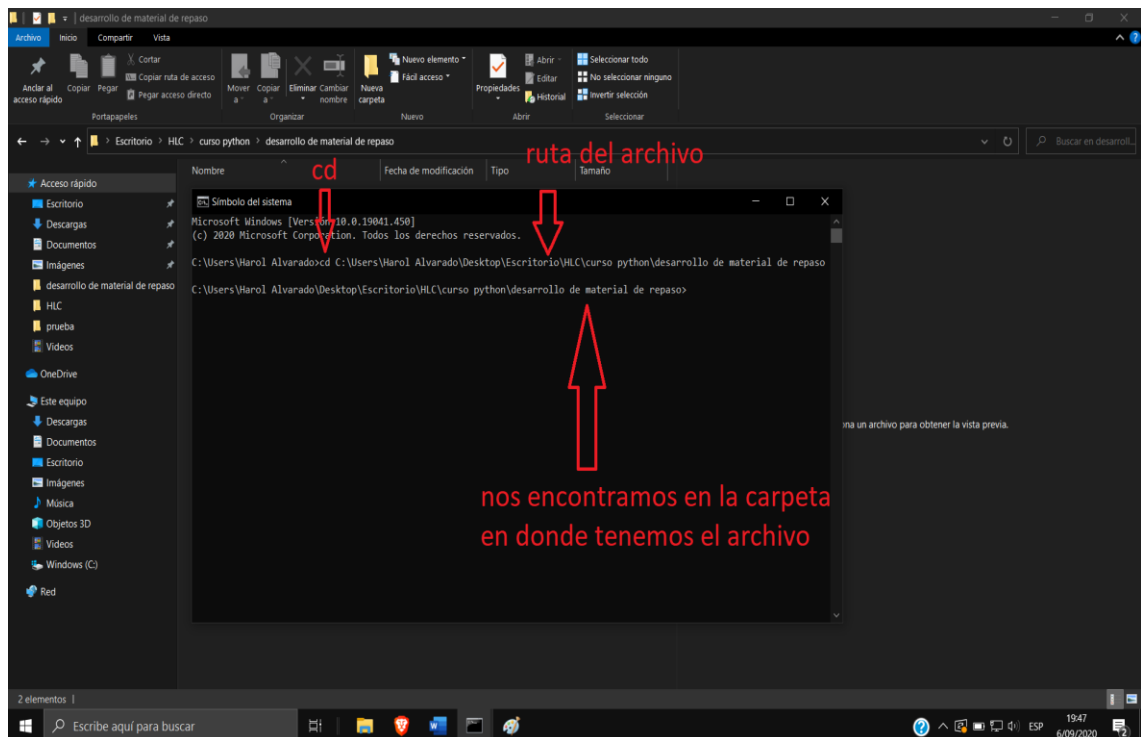
3.3. Ahora daremos en archivo->guardar como, elegiremos el nombre de nuestro archivo, el cual será prueba, pero añadiremos la extensión ".py", por lo que finalmente el nombre quedaría de la siguiente manera "prueba.py", además de eso modificaremos el tipo de archivo de documento de texto(.txt) a todos los archivos y finalmente darle click en guardar.



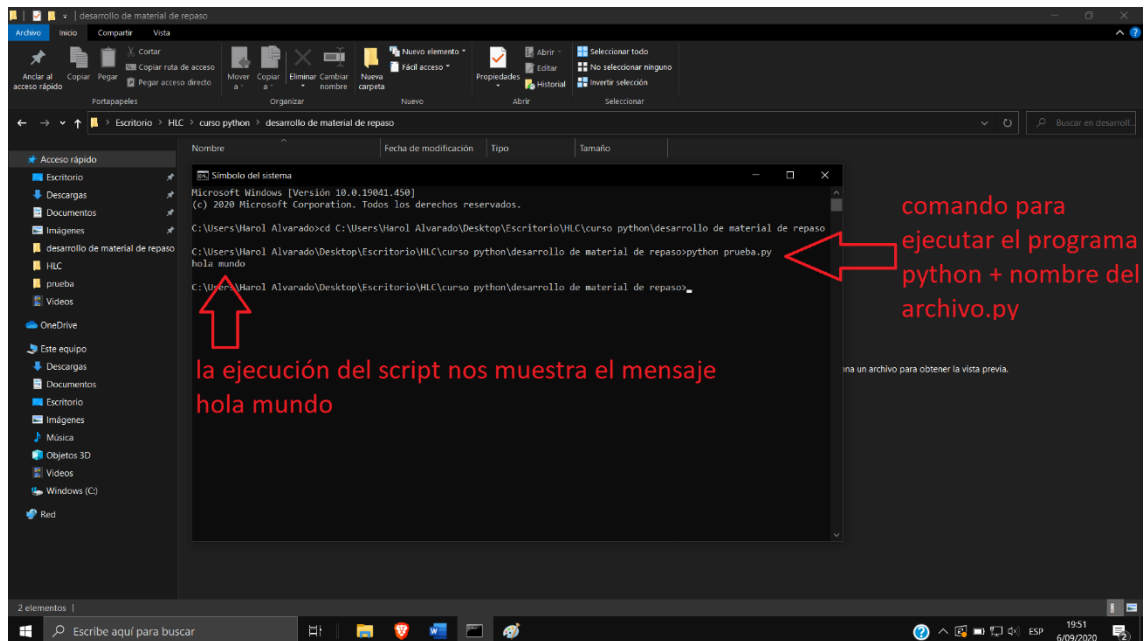
### 3.4. Ahora tenemos nuestro script de Python creado y ya que tenemos agregado Python al path entonces se mostrará el ícono de Python en el script creado



### 3.5. Copiaremos la ruta en donde se encuentra el archivo y colocaremos en la consola cd + ruta y presionamos ENTER, como se muestra en la siguiente imagen



3.6. Ya ubicados en la carpeta del archivo, escribiremos lo siguiente python (nombre del archivo con la extensión) → que en nuestro caso sería python prueba.py, esto nos permitirá ejecutar el script por lo que se mostrará en la consola: hola mundo



#### 4. Variable y Tipos de datos:

Antes de poder ver los tipos de datos que podemos manejar en Python, debemos saber de un concepto previo, la variable.

##### 4.1. ¿Qué es una variable?

Es un espacio en memoria donde podemos almacenar un valor, el valor almacenado puede ser utilizado cada vez que lo necesitemos, solo tenemos que llamarlo mediante el nombre que le asignemos a la variable.

Para poder entender mejor el concepto de la variable, utilizaremos una analogía más cercana a la realidad, imaginemos que nosotros queremos guardar un objeto, entonces buscaremos una caja donde poder guardarlo, y a esta caja le colocaremos una etiqueta con el nombre de “conector”, entonces cada vez que nosotros necesitemos usar ese objeto, tan solo tendremos que buscar el nombre de esa etiqueta y encontraremos la caja donde se guardó el objeto, y así podremos utilizar el objeto, de forma similar funciona una variable, la caja vendría a ser la memoria, el objeto vendría a ser el valor que deseamos guardar en memoria, y el nombre en la etiqueta, vendría a ser el nombre de la variable.

Listo ahora que ya sabemos que es una variable, veremos que tipos de datos podemos manejar en Python.

##### 4.2. Tipos de Datos:

###### 4.2.1. Numéricos:

Dentro de este tipo de dato tendremos:

- Los números enteros: Tipo de dato int, por ejemplo: 1, 2, 100



- Los números decimales: Tipo de dato float, por ejemplo 1.2, 5.6, 7.9
- Los números complejos: Tipo de dato complex, por ejemplo 4+5i, 3i, 2i

#### 4.2.2. Cadenas de Texto y Caracteres:

Dentro de Python el carácter se maneja como si fuera una cadena de texto, por lo tanto, en este ítem tendremos solo el tipo de dato "string".

Por ejemplo:

- "a", 'b', "cadena de texto", 'ejemplo'

Podemos ver en el ejemplo que utilizamos tanto las comillas simples como las comillas dobles, hasta el momento obtenemos la misma funcionalidad en una cadena de texto, más adelante veremos ejemplos en donde habrá una pequeña diferencia.

#### 4.2.3. Booleanos.

Este tipo de dato solo puede tomar dos valores: True o False.


#### 4.2.4. Listas, Tuplas y Diccionarios

Estos son estructuras de datos que me permiten almacenar diferentes tipos de datos y hasta estructuras de datos, veremos con mayor detalle estas estructuras más adelante.

### 4.3. Creación de una variable:

Una de las características de Python es el tipado dinámico, esto nos permite a la hora de crear una variable no definir el tipo de variable a almacenar, es decir, tan solo necesitamos el nombre de la variable y el valor que deseamos almacenar en ella. Veamos el siguiente ejemplo:

variable = 5



nombre de la variable    valor de la variable

Como podemos observar en la figura, tan solo necesitamos el nombre de la variable, y el valor que queremos asignar a la variable.

Al proceso de asignar un valor a una variable por primera vez se le conoce como inicializar una variable. En este caso nosotros le asignamos un valor, pero en el transcurso del curso, notarán que asignamos de una manera más dinámica un valor a una variable, ya sea mediante la función input (que veremos más adelante) o de alguna otra forma.

## 5. Función print

Cada vez que nosotros necesitemos imprimir ya sea un mensaje o un valor en la consola utilizaremos la función print():

Ejemplo:

- print("hola mundo")
- print("un valor entero es:",5)

La función print nos ofrece diferentes parámetros que podemos modificar, como la separación entre los mensajes, el salto de línea, o el almacenamiento en el buffer

⇒ print(objeto\* , sep = " ", end = "\n", file=sys.stdout, flush = False)

Explicemos un poco más cada parámetro

- objeto\*: son los objetos que se va a imprimir o mostrar en pantalla, el asterisco indica que se pueden pasar varios objetos.
- sep: parámetro que recibe un carácter separador de los objetos, por defecto el separador es " " (espacio).
- end: parámetro que recibe un carácter que se agrega al final de la impresión, por defecto el carácter final es "\n" (salto de línea)
- file: parámetro que recibe un objeto de escritura, por defecto es sys.stdout
- flush: parámetro que recibe un valor booleano, por defecto es False

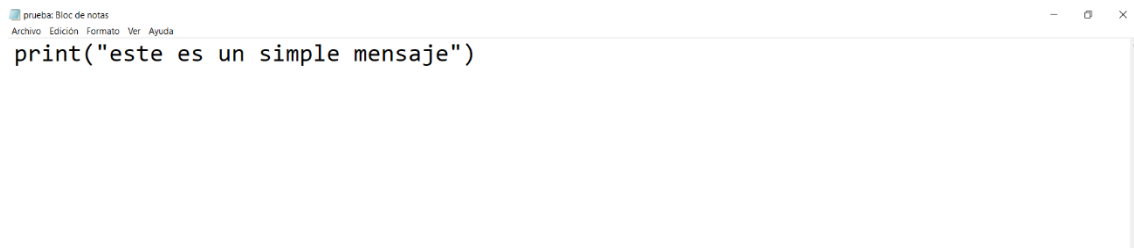
Si en el parámetro flush se recibe un valor True entonces la salida se almacena en el buffer, si recibe el valor False entonces la salida no se almacena en el buffer.

Los parámetros sep, end, file, flush, son parámetros opcionales, esto quiere decir que no es necesario asignarles un valor cada que se necesite utilizar la función print(), ya que tienen valores asignados por defecto.

Ahora veremos ejemplos de dos parámetros, posteriormente trabajaremos los demás:

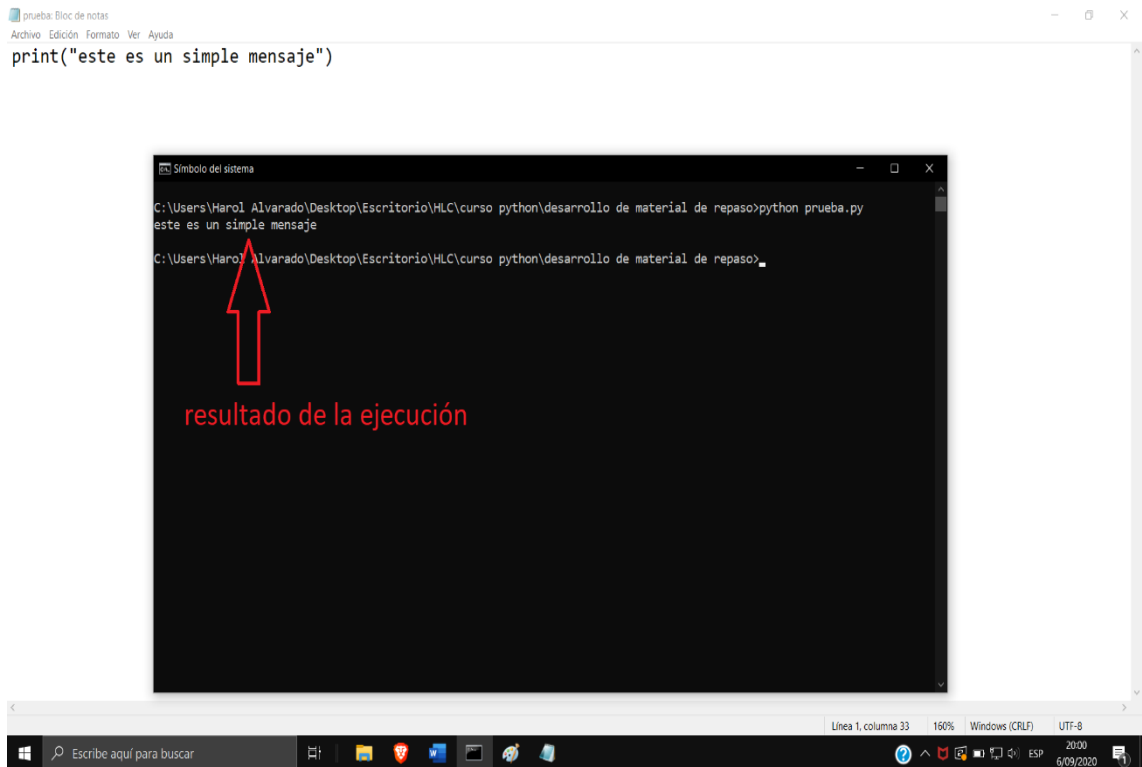
Para estos ejemplos crearemos un script de python llamado prueba.py (ver creación de un script python inciso 3).

Si tan solo necesitamos imprimir un solo mensaje en la consola pues escribiríamos lo siguiente y guardamos cambios (antes de ejecutar un script de python siempre debe verificar que guardó los cambios en el archivo correctamente, si lo está creando por primera vez el archivo ver inciso 3, de lo contrario solo haga click en archivo → guardar (recuerde que por el momento estamos utilizando el bloc de notas):



```
print("este es un simple mensaje")
```

Como resultado en la consola tendríamos lo siguiente:



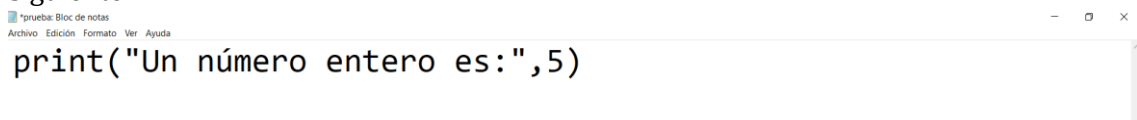
The image shows a Notepad window titled "prueba: Bloc de notas" with the code `print("este es un simple mensaje")`. Below it is a Windows Command Prompt window titled "Símbolo del sistema" showing the command `C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>python prueba.py` and its output `este es un simple mensaje`. A red arrow points from the text "resultado de la ejecución" to the output line in the command prompt.

```
prueba: Bloc de notas
Archivo Edición Formato Ver Ayuda
print("este es un simple mensaje")
```

```
Símbolo del sistema
C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>python prueba.py
este es un simple mensaje
C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>
```

resultado de la ejecución

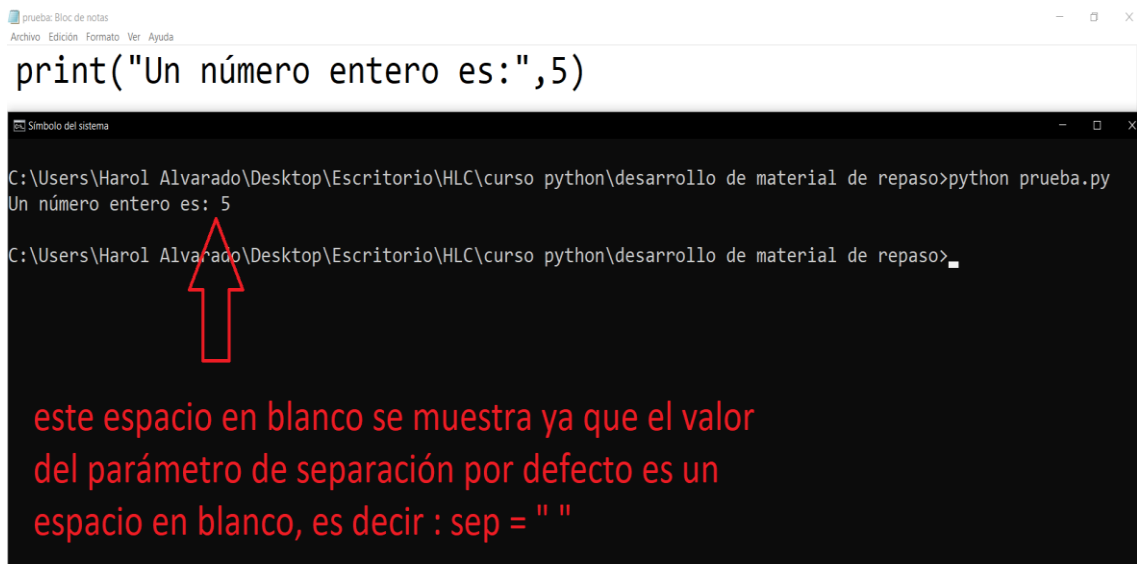
A continuación, imprimiremos varios mensajes en la consola para esto hacemos lo siguiente:



The image shows a Notepad window titled "prueba: Bloc de notas" with the code `print("Un número entero es:",5)`.

```
prueba: Bloc de notas
Archivo Edición Formato Ver Ayuda
print("Un número entero es:",5)
```

Como podemos observar si necesitamos mostrar por pantalla varios objetos solo necesitamos separarlos con una coma, en la ejecución del script notaremos que los objetos se mostrarán separados por defecto por un espacio



The image shows a Notepad window titled "prueba: Bloc de notas" with the code `print("Un número entero es:",5)`. Below it is a Windows Command Prompt window titled "Símbolo del sistema" showing the command `C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>python prueba.py` and its output `Un número entero es: 5`. A red arrow points from the text "este espacio en blanco se muestra ya que el valor del parámetro de separación por defecto es un espacio en blanco, es decir : sep = ' '" to the space between "es:" and "5" in the command prompt output.

```
prueba: Bloc de notas
Archivo Edición Formato Ver Ayuda
print("Un número entero es:",5)
```

```
Símbolo del sistema
C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>python prueba.py
Un número entero es: 5
C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>
```

este espacio en blanco se muestra ya que el valor del parámetro de separación por defecto es un espacio en blanco, es decir : sep = " "

Ahora cambiaremos el parámetro de separación y le asignaremos un guion, es decir:

```
sep = "-"
```

prueba: Bloc de notas  
Archivo Edición Formato Ver Ayuda

```
print("Un número entero es:",5,sep="-")
```

cambiamos el valor  
del parámetro sep

Símbolo del sistema

```
C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>python prueba.py  
Un número entero es:-5
```

ya que cambiamos el valor de sep = "-", ya no nos muestra un espacio en blanco, ahora nos muestra un guión como caracter separador

Ahora veamos el parámetro end, para esto utilizaremos dos veces la función print() y analicemos la salida de la ejecución.

prueba: Bloc de notas  
Archivo Edición Formato Ver Ayuda

```
print("hola como")  
print("estas")
```

utilización de dos print()

Símbolo del sistema

```
C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>python prueba.py  
hola como  
estas
```

salida

Notamos que en la salida los dos mensaje salen separados por un salto de línea, esto se debe al valor por defecto que tiene asignado el parámetro end, es decir, end="\n", ahora pasaremos a modificar este parámetro y observemos el comportamiento

prueba: Bloc de notas  
Archivo Edición Formato Ver Ayuda

```
print("hola como",end="-")  
print("estas")
```

modificación del parámetro end

Símbolo del sistema

```
C:\Users\Harol Alvarado\Desktop\Escritorio\HLC\curso python\desarrollo de material de repaso>python prueba.py  
hola como-estas
```

debido a que ahora hemos modificado el valor de end="-", los mensajes ya no se separan por un salto de línea, ahora se separan con un guión

6. Función input
7. Métodos de strings
8. Operadores Aritméticos
9. Operadores Lógicos.
10. Operadores a nivel de bit.
11. Estructuras condicionales.
12. Estructuras repetitivas.
13. Continue, pass.
14. Listas, tuplas, diccionarios.
15. Funciones.
16. Módulos.
17. Paquetes.
18. Excepciones.
19. Programación Orientada a Objetos
  - 19.1. Clases.
  - 19.2. Objetos.
  - 19.3. Métodos.
  - 19.4. Atributos.
  - 19.5. Constructores.
  - 19.6. Herencia.
  - 19.7. Polimorfismo.
20. Serialización.
21. Manejo de Archivos.
22. Módulo Thread.
23. Interfaces gráficas con Tkinter.
  - 23.1. Widgets.
  - 23.2. Eventos.
  - 23.3. Layouts.
24. Módulo Sqlite3.
25. Módulo Mysql-connector.
26. Módulo Sockets.
27. Expresiones Regulares.