**Contenido**

[Python 2](#_Toc64381329)

[Algotrimo 2](#_Toc64381330)

[Editor de código 3](#_Toc64381331)

[Resumen 1 Operadores aritméticos: 4](#_Toc64381332)

[Variables 5](#_Toc64381333)

[Operadores lógicos y de comparación 9](#_Toc64381334)

[Cmder 11](#_Toc64381335)

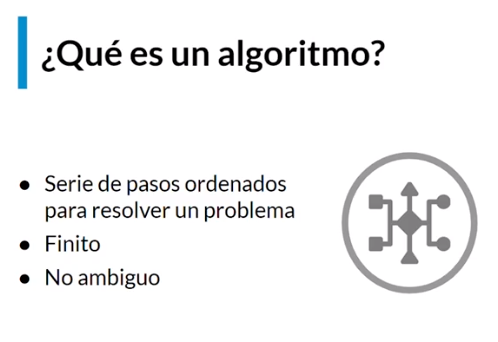
[Python Ingresar por medio de Cmder 12](#_Toc64381336)

[If\_\_name\_\_ == '\_\_main\_\_' 13](#_Toc64381337)

# Python

# Algotrimo



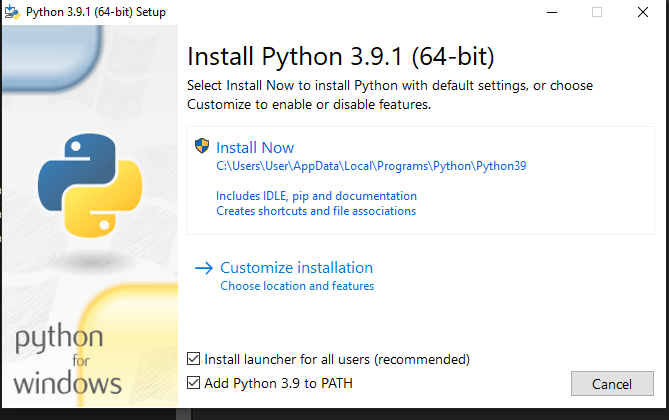


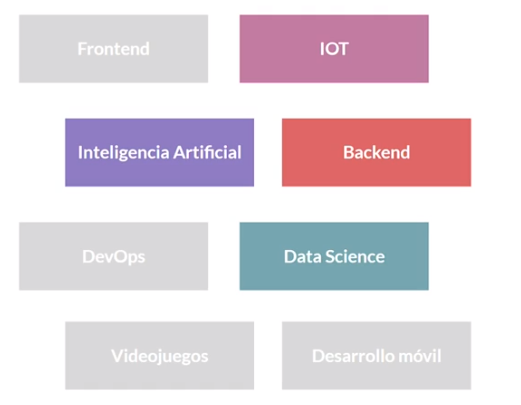
# Editor de código

[Visual Studio Code - Code Editing. Redefined](https://code.visualstudio.com/)

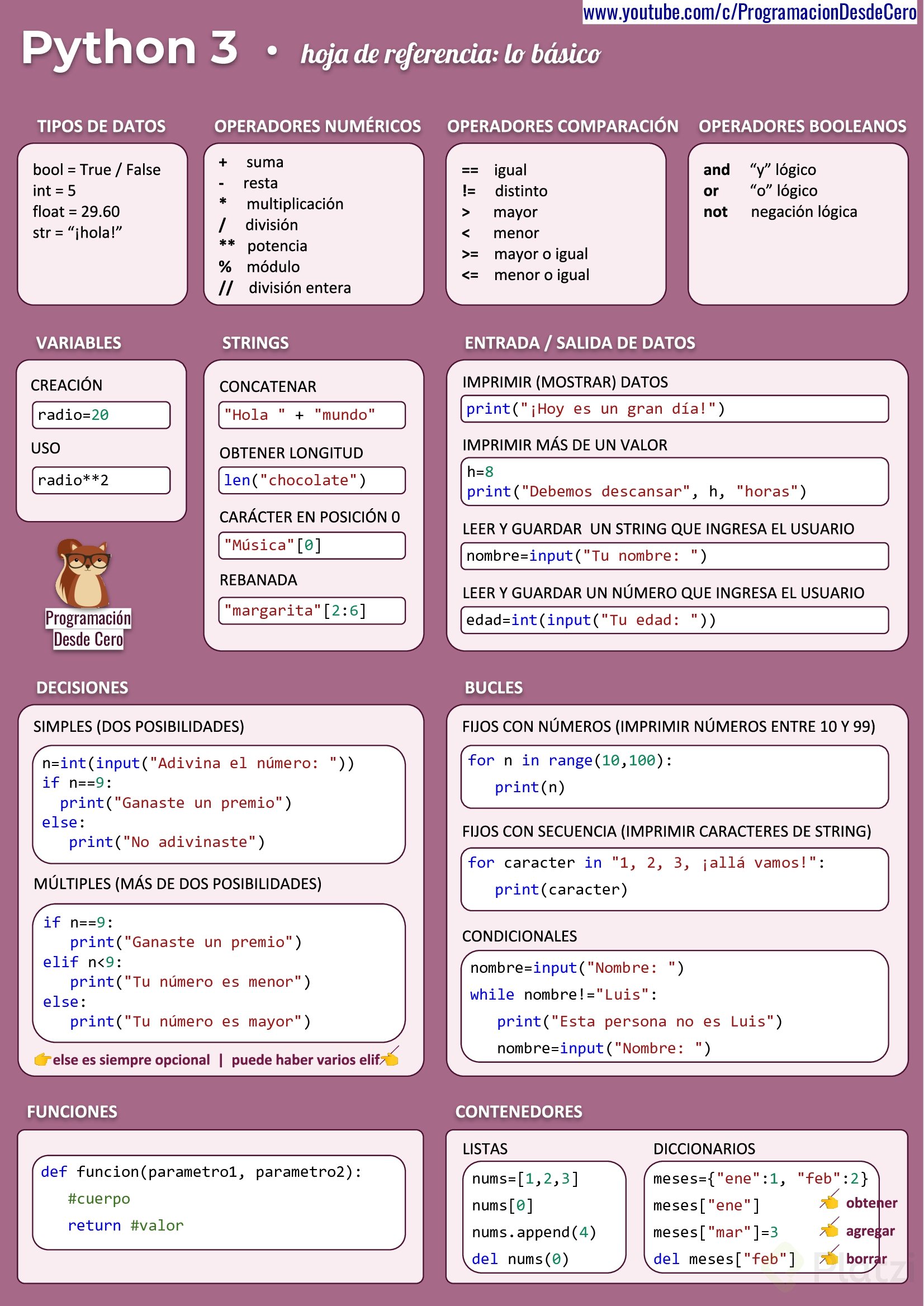
[Cmder | Console Emulator](https://cmder.net/)

[Welcome to Python.org](https://python.org/)





# Resumen 1 Operadores aritméticos:



* Operadores aritméticos  
  Primero, para iniciar la consola interactiva de Python debemos escribir el comando \*\*py \*\*en Windows, pero en otros sistemas el comando es **python3**. Ahora, podemos comenzar. En la consola nos permite escribir operaciones matemáticas como 5 + 5 sin escribir nada más, pero en el editor de código debemos “imprimir” el resultado, de la siguiente manera: print(5 + 5). Con esto obtendremos el resultado. Ahora veamos como se realiza cada operación aritmética:  
  -Suma: 5 + 5  
  -Resta: 5 - 5  
  -Multiplicación: 5 \* 5  
  -División (con decimales): 5 / 5  
  -División (sin decimales): 21 // 5  
  -Resto de la división: 21 % 5  
  -Potencia: 2 \*\* 2  
  -Raíz cuadrada:

>>>math.sqrt(9)

3.0

>>>math.sqrt(11.11) }

3.3331666624997918

>>>math.sqrt(Decimal('6.25'))

2.5```

Python respeta **la** separació**n** **de** términos, por lo **que** si escribimos 5 + 5 \* 2 multiplicará primero 5 x 2. **En** el caso **de** **que** quisiéramos **que** primero sume 5 + 5 ponemos paréntesis:

(5 + 5) \* 2.

Variables  
Es un lugar en memoria (una especie de caja) en el que podemos guardar objetos (números, texto, etc). Esta variable posee un identificador o nombre con el cual podemos llamarla cuando la necesitemos.

* Asignación de variables  
  En python creamos las variables asignándoles un valor de la siguiente manera:  
  <identificador> = <valor>  
  en este caso el signo = se lee como “asignar”
* Reasignación de variables  
  Podemos en cualquier momento cambiar el valor de nuestra variable volviendo a asignar un valor al mismo identificador  
  <identificador> = <nuevo\_valor>

**Reglas en el uso de identificadores**

* No pueden empezar con un número.
* Deben estar en minúsculas
* Para separar las palabras usamos el guion bajo: \_
* Estas reglas son aplicadas al lenguaje python, en otros lenguajes pueden haber otras reglas.

**Objetos**  
Un objeto es una forma de modelar el mundo, en los lenguajes de programación se caracterizan por tener métodos y atributos. En python todo es un objeto.

Tipos de datos  
Podemos encontrar cuatros tipo de datos que vienen definidos por defecto en python, a estos tipos de datos los conocemos como primitivos.

* Integers → Números Enteros
* Floats → Números de punto flotantes (decimales)
* Strings → Cadena de caracteres (texto)
* Boolean → Boolenaos (Verdadero o Falso)

**Notas**

* Algunos operadores aritméticos pueden funcionar para operar con otros tipos de datos, por ejemplo: podemos sumar strings, lo que concatena el texto o multiplicar un entero por un strings lo que repetirá el string las veces que indique el entero

"hola" + " " + "mundo" = "hola mundo"

"hola" **\*** 4 = "holaholaholahola"

* **EJEMPLO DE CONCATENACIÓN 😢**
* >>> realidad\_1 = 'Ella'
* >>> realidad\_2 = ' NO te'
* >>> realidad\_3 = ' ama :,('
* >>> realidad = realidad\_1 + '' + realidad\_2 + realidad\_3
* >>> print (realidad)
* Ella NO te ama :,(

Super útil todo, un pequeño resumen:

* Input("") para pedirle al usuario que introduzca datos.
* int() con datos o variables dentro de parentesis para convertirlo en número entero.
* str() para convertir números tanto decimales como enteros a strings.

No siempre, es buena idea hacer esto, porque se alguien en vez de poner un número agrega una letra, tu código fallara  
Te pongo una ejemplo del fallo

Escribe un número: ho

Traceback (most recent **call** **last**):

**File** "main.py", line 1, **in** <**module**>

numero3 = int(**input**('Escribe un número: '))

ValueError: invalid literal **for** int() **with** base 10: 'ho'

La solución a eso, es un poco avanzado pero la mejor practica seria hacerlo así

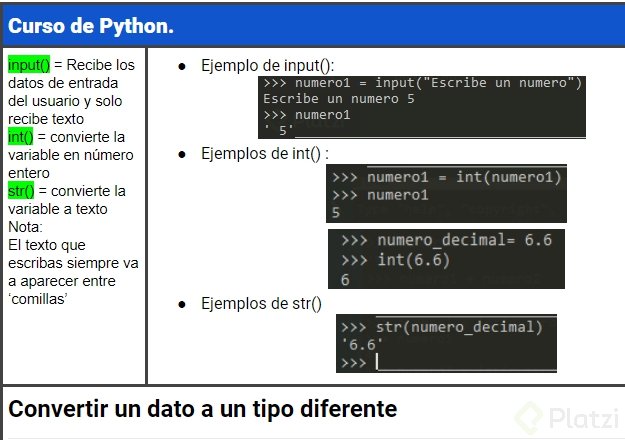
numero3 = **input**('Escribe un **n**úmero: ')

**if** numero3.isnumeric():

**print**('hola')

**else**:

**print**('hay un **error**, **no** agregaste un **n**úmero')



Un resúmen en CMDER:

numero1 = input("Digite un número: ")  
Digite un número: 20

numero2 =("Digite otro número: ")  
numero2 =input("Digite otro número: ")  
Digite otro número: 60  
**>>> El “input” sirve para indicarla una acción al usuario**

numero1 + numero2  
’2060’  
***>>> Cambiando texto a número entero------***

numero1 =int(numero1)  
numero1  
20

numero2 =int(numero2)  
numero2  
60  
**>>> El “int” se usa para convertir una variable o contenido a un número entero—**

numero1 + numero2  
80  
*>>> aqui si se suma los numeros cosa que en lo anterior no se pudo por ser texto y todo texto se concatena*  
***>>> Pasando una variable decimal a un número entero***

numero\_decimal =7.9  
int(7.9)  
7  
*>>> Aquí el “int” convierte 7.9 a su entero 7 (no lo rendodea solo saca el número entero)*  
***>>> Convirtiendo un decimal o numero entero a texto…***  
*>>> Usamos str*

numero\_decimal  
7.9

str(numero\_decimal)  
‘7.9’

numero1

nombre\_variable = input( “Ingrasa texto”)  
numero1 = input(“Escribe un numero:”)

'4’  
numero2 = input(“Escribe un numero”)

'5’  
Si sumas  
numero + numero 2 = '45’  
PARA CONVERTIRLO A NUMERO  
numero1 = int(numero1)  
numero2 = int(nimero2)  
numero1 + numero 2  
9  
CONVERTIR DE NUMERO A TEXTO  
str(numero1)  
‘4’

# Cambiar tipo de datos :

Variable = **int**(Variable) --------- cambia a tipo entero  
Variable = **str** (Variable) -------- cambia a tipo cadena de caracteres  
Variable = \*\*float \*\*(Variable)------- cambia a tipo decimal  
Variable = **bool** (25) --------------cambia a booleano True  
Variable = **bool**(0)----------------- cambia a booleano False

# Operadores lógicos y de comparación

**OPERADORES LÓGICOS**  
**and =** desde el contexto de la lógica de conjunstos and es una unión (U ) y se requiere que ambos valores sean verdaderos para que el planteamiento sea verdadero.  
\*\*or = \*\* al contrario del and or es una intersección (Ո) por lo tanto desde la lógica de conjuntos se hace verdadero cuando al menos una de las condiciones se cumple

**Operadores Lógicos**

* or - para que se pueda cumplir la condicion almenos una de las variables comparadas tiene que ser verdaderas, para que la condicion de falso todas las variables comparadas tienen que ser false
* and - para que se pueda cumplir la condicion todas las variables comparadas tienen que ser verdaderas
* not - invierte el valor de una variable, es decir si una variable es true y colocamos not la variable cambiara a false

**Operadores de comparación**

* (==) igualdad, nos valida que el contenido de las variables sean iguales
* (!=) distinto, nos valida que el contenido de las variables no sean iguales
* (>) mayor, compara si una variable es mayor que el otro
* (<) menor, compara si una variable es menor que el otro
* (<=) mayor o igual
* (≤) menor o igual

Operadores lógicos y de comparación

operador de comparación  
== Compara dos valores y te dice si son iguales o no.  
!= Compara dos valores y te dice sin son diferentes o no.  
-> Compara si es mayor que otro valor.  
-> Compara si es menor que otro valor.

= igual o mayor que el valor a comparar.  
<= igual o menor que el valor a comparar.

Operadores lógicos  
and para comparar si dos valores son verdaderos.  
or para comparar si dos valores son falsos.  
not para invertir el valor booleano.  
Los operadores lógicos son unas operaciones que trabajan con valores booleanos.

and: “y” lógico. Este operador da como resultado True si y sólo si sus dos operandos son True:

[]>>> True and True  
True

True and False  
False

False and True  
False

False and False  
False[]

or: “o” lógico. Este operador da como resultado True si algún operando es True:  
{}>>> True or True  
True

True or False  
True

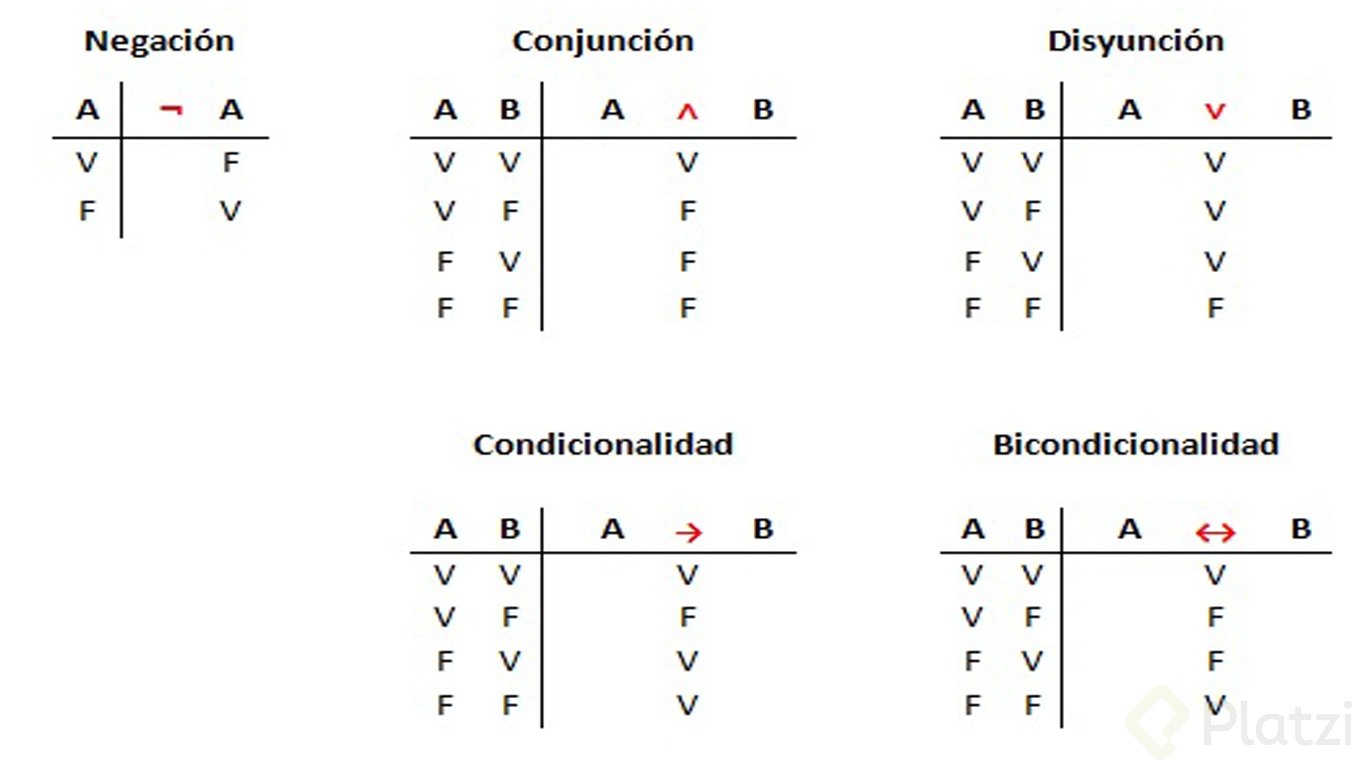
False or True  
True

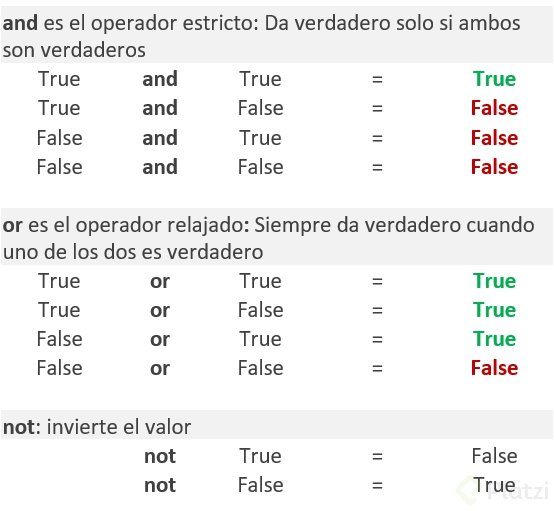
False or False  
False{}

(not: negación. Este operador da como resultado True si y sólo si su argumento es False:

not True  
False

not False  
True)



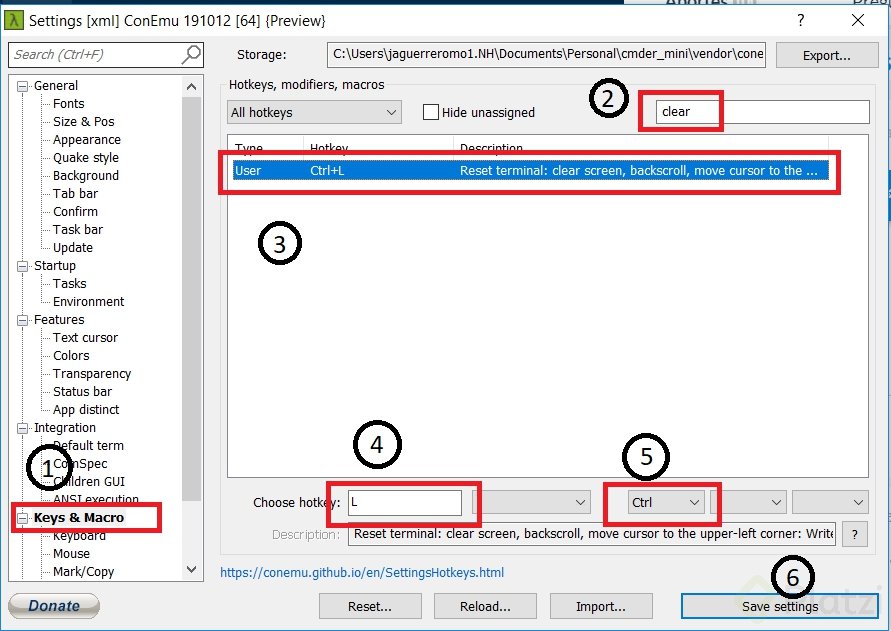


# Cmder

Para los que no puede limpiar la consola (Cmder) con **Ctrl + L**:  
Abran las configuraciones con: **Win + Alt + P**  
Luego sigan los pasos:

1. Buscar en la parte izquierda **Keys & Macro**. Seleccionarla.
2. Buscar la opción **Clear**
3. Seleccionar la opción que tenga en la Descripción **Reset terminal**
4. Seleccionar la caja de texto (que debe estar vacía), y luego oprimir la tecla **L**
5. Buscar en el desplegable la abreviatura **Ctrl**
6. Guardar cambios.

Al volver a la consola y usar **Ctrl + L** va a limpiar literalmente toda la pantalla. (Incluso la ruta si está fuera de Python), solo basta con dar un **Enter** para que vuelva al entorno que estaba usando antes de limpiar la consola.



Otros comandos utilizados **ls** -**la** Listar y ver archivos ocultos **pwd** Ver **en** **que** directorio estoy **cd** Cambiar directorio **rm** Borrar mv Mover archivo **rmdir** Borrar directorio **ls** -t Ordena los archivos por fecha **de** modificacion **ls** -x Ordena elementos por nombre y extension **ls** -X Orcena elementos por extension y nombre **ls** -S Ordena los archivos por tamaño **de** archivo **cat** Muestra el contenido completo **de** un archivo head Muestra las primeras lineas **de** un archivo tail Muestra las ultimas lineas **de** un archivo touch crea archivos

# Python Ingresar por medio de Cmder

C:\Users\User>py

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>

Salir

>>> exit()

C:\Users\User>

# If\_\_name\_\_ == '\_\_main\_\_'

Es común encontrarnos código con esta forma:

def hacer\_algo():

print("algo")

if \_\_name\_\_ == "\_\_main\_\_":

hacer\_algo()

En lugar de, por ejemplo:

def hacer\_algo():

print("algo")

hacer\_algo()

También notamos que la variable \_\_name\_\_ no es inicializada por nosotros, pero existe en el entorno.

Para ver como funciona puedes probar algún código (siguiendo tu propio ejemplo):

Creas un módulo al que llamaremos mi\_modulo.py:

print("¡Hola desde mi\_modulo.py!")

def hacer\_algo():

print("¡Soy una función y hago algo!")

if \_\_name\_\_ == "\_\_main\_\_":

print('Ejecutando como programa principal')

hacer\_algo()

print("¡Adiós desde mi\_módulo.py!") # Debajo del condicional pero fuera del mismo

En la misma carpeta creas otro módulo al que llamaremos principal.py

import mi\_modulo

print("¡Hola desde principal.py!")

mi\_modulo.hacer\_algo()

print("¡Adiós desde principal.py!")

Si ejecutas el script mi\_modulo.py directamente:

$ python mi\_modulo.py

el valor de \_\_name\_\_ será "\_\_main\_\_" y se ejecutará el bloque que hay dentro del if \_\_name\_\_ == “\_\_main\_\_”::

variable.upper() = 'todos los caracteres en MAYÚSCULAS'

variable.capitalize() = 'solo la primera en MAYÚSCULA'

variable.lower() = 'todos los caracteres en minúscula'

variable.strip() = 'eliminar espacios basura del string'

variable.replace('caractera a cambiar', 'caracter por poner') = remplazar caracter

Método: es una función que es especial para un tipo de dato en particular. Por ejemplo, si queremos que el texto ingresado se transforme en mayúsculas.

.

.

variable.strip() = El método strip eliminará todos los caracteres vacíos que pueda contener la variable

.

variable.lower() = El método lower convertirá a las letras en minúsculas.

.

variable.upper() = El método upper convertirá a las letras en mayúsculas.

.

variable.capitalize() = El método capitalize convertirá a la primer letra de la cadena de caracteres en mayúscula.

.

variable.replace (‘o’, ‘a’) = El método replace remplazará un caracterer por otro. En este caso remplazará todas las ‘o’ por el caracter ‘a’.

.

Índices: se escriben entre corchetes al lado de la variable y son apuntadores númericos a cada caracterer. Como se ve en el video, cuando utiliza la variable nombre[1] aparece la letra ‘a’ dado que dicha variable tiene almacenada en ese momento la cadena de caracteres ‘facundo’ donde la ‘a’ es el segundo caracterer.

.

Aclaración: se comienza a contar caracteres con el 0, siguiendo el ejemplo, la letra ‘f’ de ‘facundo’ es el caracter número 0.

.

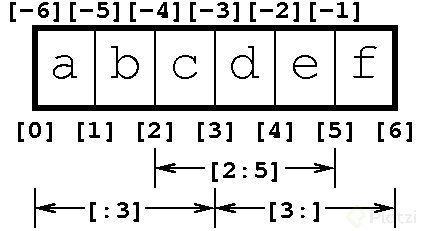
Finalmente, se menciona a la funcion len() la cual muestra como resultado la cantidad de caracteres que contiene un string, tanto si este está almacenado en una variable como si ingresamos directamente un string para que cuente los caracteres.

Metodo va después de un punto y es una funciona asociada

## 30 métodos de las cadenas

<https://recursospython.com/guias-y-manuales/30-metodos-de-las-cadenas/>

## Trabajando con texto: slices



var\_txt[a:] → String desde el 'a'

var\_txt[:b] → String hasta el 'b'

var\_txt[a:b] → String desde el 'a' hasta el 'b'

var\_txt[::x] → String en saltos de x

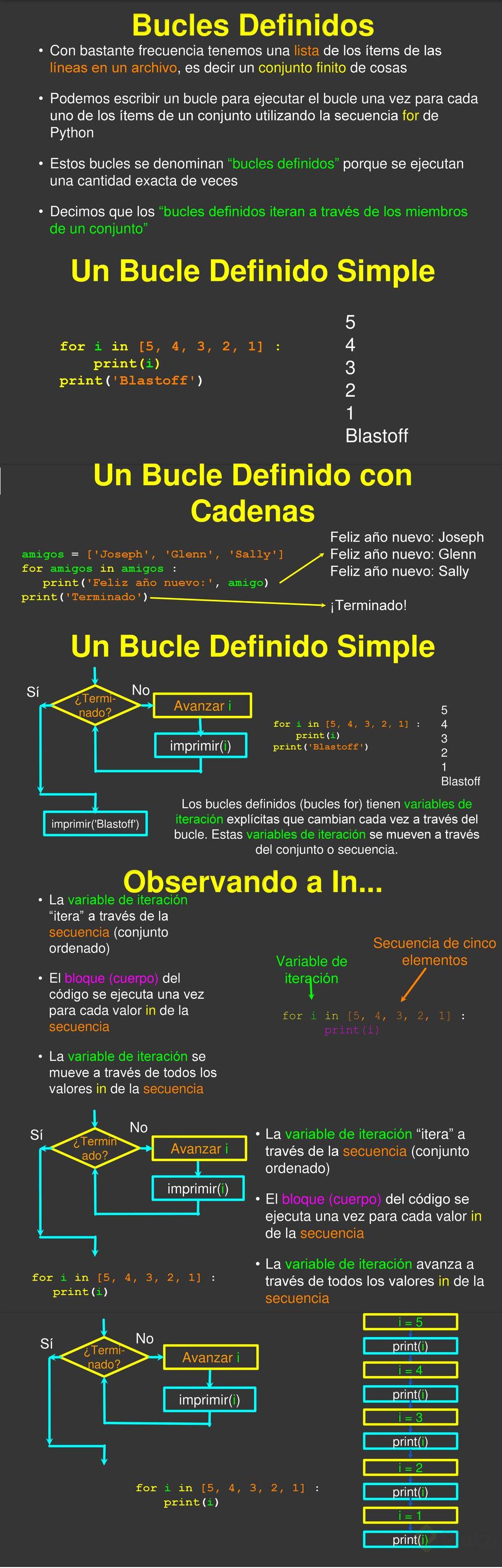
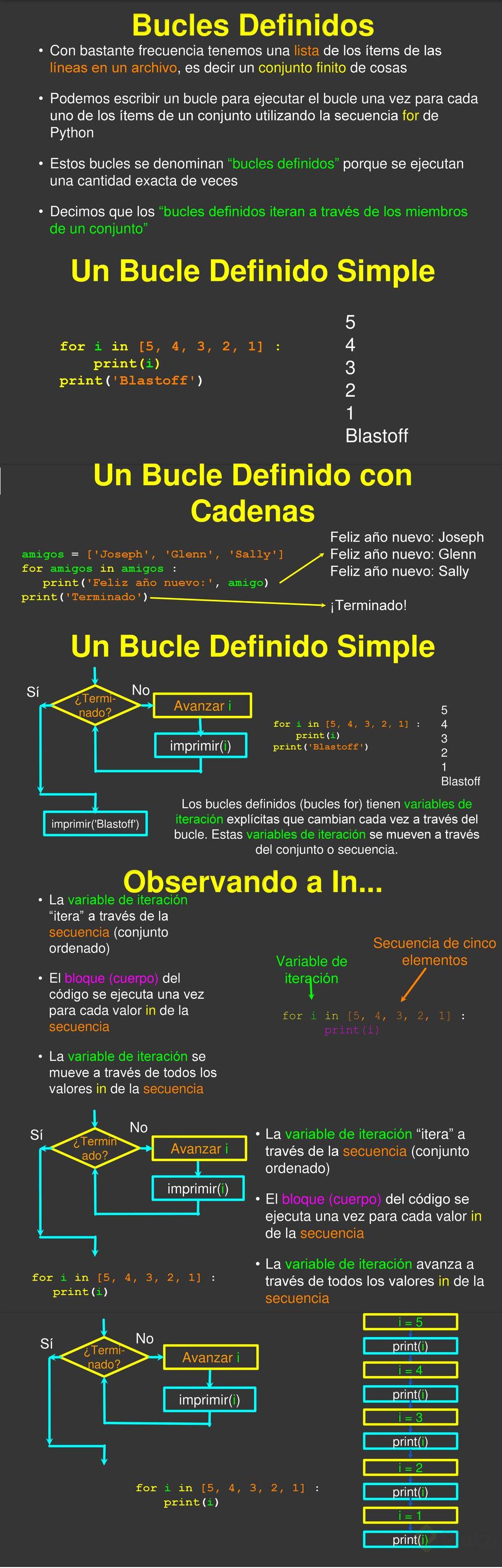
var\_txt[::-x] → String invertido en saltos de x```

## Proyecto: palíndromo

Una palabra que se le al derecho o aleves igual

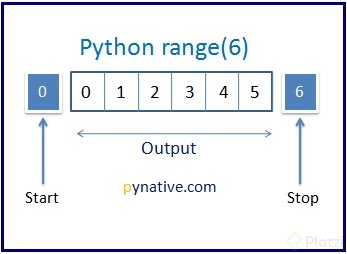
Luz Azul

## Aprendiendo bucles

## El ciclo while

## Explorando un bucle diferente: el ciclo for



## Recorriendo un string con for

## Interrumpiendo ciclos con break y continue

## Proyecto: prueba de primalidad

## Proyecto: videojuego

## Almacenar varios valores en una variable: listas

Listas son un conjunto de elementos asociados a una variable!!

Algunos de los métodos del tipo lista

lista.append(“Elemento”) Este método permite agregar al final de la lista un nuevo elemento

lista.pop(1) Este método elimina un elemento de la lista por medio de su índice

Recuerda que puedes recorrer una lista por medio de un for, también por usar slice(rebanadas) en las listas

lista[::-1] Retorna la lista en reversa

lista[1:3] Nos devuelve los elementos de la lista desde el índice 1 al índice 2 ya que el final es excluyente

lista.extend(iterable) #extiende la lista con valores dentro de un iterable como un range()

lista.insert(i, ‘valor’) #Agrega un valor en la posición i y recorre todos los demás. No borra nada.

lista.pop(i) #Elimina valor en la posición i de la lista.

lista.remove(‘valor’) #Elimina el primer elemento con ese valor.

lista.clear() #Borra elementos en la lista.

lista.index(‘valor’) #Retorna posición del primer elemento con el valor.

lista.index(‘valor’, start, end) #Retorna posición del elemento con el valor dentro de los elementos desde posición start hasta posición end)

lista.count(‘valor’) #Cuenta cuántas veces esta ese valor en la lista.

lista.sort() #Ordena los elementos de mayor a menor.

lista.sort(reverse = True) #Ordena los elementos de menor a mayor.

lista.reverse() #Invierte los elementos

lista.copy() #Genera una copia de la lista. También útil para clonar listas.

## Entendiendo cómo funcionan las tuplas

#declarar tupla

mi\_tupla = ()

mi\_tupla = (1,2,3)

#generar una tupla de 1 solo valor (Obligatorio la ,)

mi\_tupla = (1,)

#acceder a un indice de la tupla

mi\_tupla = (1,2,3)

mi\_tupla[0] #1

mi\_tupla[1] #2

mi\_tupla[2] #3

#reasignar una tupla

mi\_tupla = (1,2,3)

mi\_otra\_tupla = (4,5,6)

mi\_tupla =+ mi\_otra\_tupla

#metodos de las tuplas

mi\_tupla = (1,1,1,2,2,3)

mi\_tupla.count(1) #3 el numero 1 aparece 3 veces en la tupla

mi\_tupla.index(3) #5 indice de la primera instancia donde se encuentra un elemento

mi\_tupla.index(1) #0

mi\_tupla.index(2) #3

## ¿Qué son los diccionarios?

Diccionarios: Son una estructura de datos mutable las cuales almacenan diferentes tipos de valores sin darle importancia a su orden. Identifican a cada elemento por una clave (Key). Se escriben entre {}.

Operaciones:

.keys() —> Retorna la clave de nuestro elemento

.values()—> Retorna una lista de elementos (valores del diccionario)

.items() —> Devuelve lista de tuplas (primero la clave y luego el valor)

.clear() —> Elimina todos los items del diccionario

.pop(“n”) —> Elimina el elemento ingresado

Me encontré esto

Tipos de datos en Python

.

Inmutables:

Strings (Texto)

Int (Número entero)

Float (Número de coma flotante)

Decimal (Número decimal)

Complex (Complejo)

Bool (Booleano)

Tuple (Tupla)

Frozenset (Conjunto congelado)

Bytes (8 bits)

Range (Rango)

None (Nulo)

.

Mutables:

List (Listado)

Dict (Diccionario)

Set (Conjunto)

Bytearray (array de bits)

MemoryView (Vista de la memoria)

## Proyecto: generador de contraseñas

## Sigue aprendiendo

Lecturas recomendadas

Inteligencia Artificial y Machine Learning

https://platzi.com/ai/

Escuela de Data Science

https://platzi.com/datos/

Facundo Nicolas

https://facundonicolas.com/

¿Qué palabra clave se utiliza para crear una función?

def

2.

¿Cuál es el orden correcto de elementos a indicar cuando se trabaja con slices?

[pasos : índice final : índice inicial]

3.

¿Cuál es la ventaja de definir strings con triple comilla doble (""") ?

Los saltos de línea visibles en la cadena de caracteres son respetados

4.

¿Por qué debería usar un editor de código para programar?

Todas las opciones

5.

¿Qué método sirve para agregar un elemento a una lista?

append

6.

Los pasos de un algoritmo son finitos

Verdadero

7.

¿Cuál es la función que sirve para pedirle datos al usuario por consola?

input

8.

¿Cuál de los siguientes identificadores es válido y correcto según los estilos de Python?

mi\_variable

9.

¿Cuál de las siguientes especialidades de la tecnología es más afín a Python?

Frontend

10.

¿Qué comando de la consola sirve para ver los archivos en una carpeta?

ls

11.

¿Cuál es el resultado de la siguiente operación en Python? 10 / 2 + 5 \* 7

40

12.

Existe una variable es\_estudiante con el valor True, y otra variable es\_profesor con el valor False. ¿Cuál es el resultado de es\_estudiante and es\_profesor?

False

13.

¿Qué es una variable?

Un sector especial, cuyo nombre es un identificador, en donde puedo guardar objetos

14.

¿Qué son los parámetros de una función?

Variables que se pasan al momento de la invocación, y que estarán disponibles para ser usadas dentro de la función

15.

¿Qué comando de la consola sirve para crear un archivo?

mkdir

16.

¿Cuál es el resultado de hacer round(10.3456, 2)?

10.35

17.

Puedo dividir un string por un número entero

Falso

18.

¿Cuál es la diferencia entre una tupla y una lista?

Todas las opciones

19.

¿Cuál de los siguientes ejemplos de una sentencia if es el correcto?

if numero > 5:

20.

Existe una variable de nombre mi\_numero que contiene el número 1 como string. ¿Cómo transformo el valor a entero y lo guardo nuevamente en mi\_numero?

mi\_numero = int(mi\_numero)

21.

Puedo multiplicar un string por un número entero

Falso

22.

Existe una variable es\_estudiante con el valor True, y otra variable es\_profesor con el valor False. ¿Cuál es el resultado de es\_estudiante or es\_profesor?

True

23.

¿Cuál es la función de la palabra clave break?

Cortar la ejecución de un ciclo

24.

¿Qué comando de la consola sirve para moverse entre carpetas?

cd

25.

¿Qué método de las strings coloca la primera letra en mayúsculas?

capitalize

¿Cuál es el orden correcto de elementos a indicar cuando se trabaja con slices?

(inical , final , pasos)

¿Cuál es el orden correcto de elementos a indicar cuando se trabaja con slices?

[índice inicial : índice final : pasos]

2.

¿Cuál es la ventaja de definir strings con triple comilla doble (""") ?

Los saltos de línea visibles en la cadena de caracteres son respetados

3.

¿Por qué debería usar un editor de código para programar?

Todas las opciones

4.

¿Qué método sirve para agregar un elemento a una lista?

append

5.

Los pasos de un algoritmo son finitos

Verdadero

6.

¿Cuál de los siguientes ejemplos de una sentencia while es el correcto?

while numero == 3:

7.

¿Cuál de las siguientes especialidades de la tecnología es más afín a Python?

Inteligencia Artificial

8.

¿Qué comando de la consola sirve para ver los archivos en una carpeta?

ls

9.

¿Cuál es el resultado de la siguiente operación en Python? 10 / 2 + 5 \* 7

40

10.

Existe una variable es\_estudiante con el valor True, y otra variable es\_profesor con el valor False. ¿Cuál es el resultado de es\_estudiante and es\_profesor?

False

11.

¿Qué es una variable?

Un sector especial, cuyo nombre es un identificador, en donde puedo guardar objetos

12.

¿Qué son los parámetros de una función?

Variables que se pasan al momento de la invocación, y que estarán disponibles para ser usadas dentro de la función

13.

¿Qué comando de la consola sirve para crear un archivo?

touch

14.

¿Cuál es el resultado de hacer round(10.3456, 2)?

10.35

15.

Puedo dividir un string por un número entero

Verdadero

16.

¿Cuál es la diferencia entre una tupla y una lista?

Todas las opciones

17.

¿Cuál de los siguientes ejemplos de una sentencia if es el correcto?

if numero > 5:

18.

Existe una variable de nombre mi\_numero que contiene el número 1 como string. ¿Cómo transformo el valor a entero y lo guardo nuevamente en mi\_numero?

mi\_numero = int(mi\_numero)

19.

Puedo multiplicar un string por un número entero

Falso

20.

¿Qué son las funciones?

Un sector especial, cuyo nombre es un identificador, en donde puedo guardar objetos

21.

Existe una variable es\_estudiante con el valor True, y otra variable es\_profesor con el valor False. ¿Cuál es el resultado de es\_estudiante or es\_profesor?

True

22.

¿Cuál es la función de la palabra clave break?

Cortar la ejecución de un ciclo

23.

¿Qué es programar?

Darle instrucciones a la computadora para resolver un problema

24.

¿Cuántos espacios debo dejar de sangría para escribir un bloque de código (debajo de una sentencia if, una función o un ciclo)?

4

25.

¿Qué método de las strings coloca la primera letra en mayúsculas?

capitalize