1000 PROGRAMADORES

Introducción a la **Programación en Python**

Ministerio de Educación Cultura, Ciencia y Tecnología Gobierno de Salta

Ministerio de Economía y Servicios Públicos Gobierno de Salta

Universidad Nacional de Salta

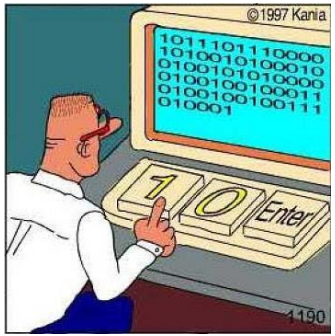
MÓDULO 2

Variables.
Números, enteros y flotantes. Cadenas de Texto.
Listas. Tuplas. Diccionarios. Conjuntos.
Índices y slicing. Métodos de colecciones.
Operadores lógicos. Operadores relacionales y de asignación.

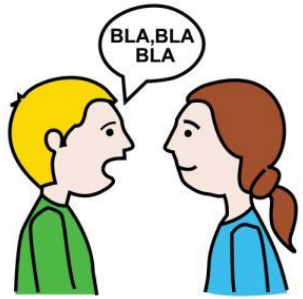
¿Hoy tendremos invitados especiales?

¿Qué es un lenguaje de programación?

Es la combinación de símbolos y reglas que permiten la elaboración de programas con los cuales la computadora puede realizar tareas o resolver problemas.



- Lenguaje de alto nivel
- Lenguaje de bajo nivel
- Lenguaje Máquina



Python

Interpretado

Multiparadigma

Multiplataforma

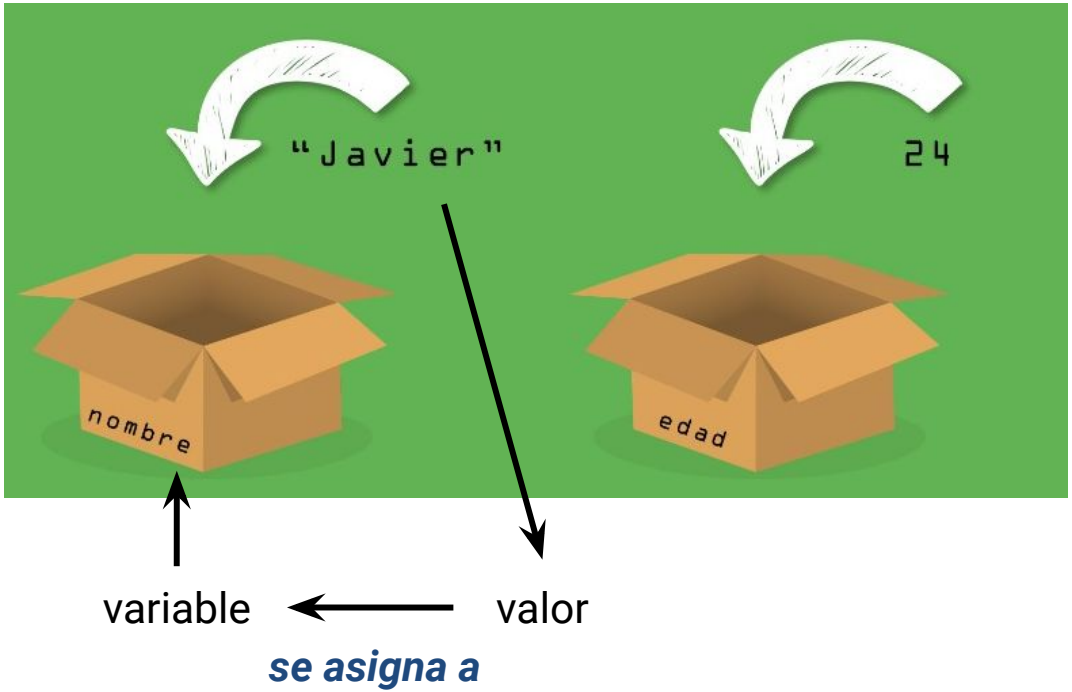
de Tipado Dinámico

Python → Google Colab

Colaboratory o Colab

*Permite escribir y ejecutar código de Python en un **navegador**, **sin configuración** requerida y ofrece facilidad para **compartir**.*

Variables



Una variable es una forma de identificar un dato que se encuentra almacenado en la memoria del ordenador

Variables

Para asignar un valor (un dato) a una variable se utiliza el operador de asignación =.

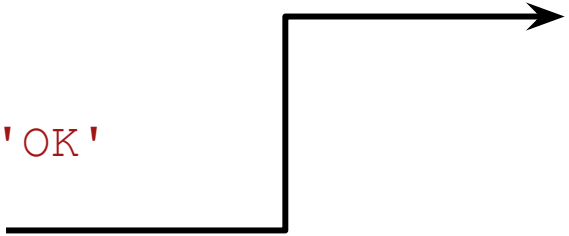
En la operación de asignación se ven involucradas tres partes:

- El operador de asignación =
- Un identificador o nombre de variable, a la izquierda del operador
- Un literal, una expresión, una llamada a una función o una combinación de todos ellos a la derecha del operador de asignación

```
suma = 2 + 2
```

Asignación de Variables

```
a = 1
b = 2
c = 2
a = (b / c) * 5
a = a + 1
a = 'Python'
a = 'Python' + 'OK'
a = A + B
```



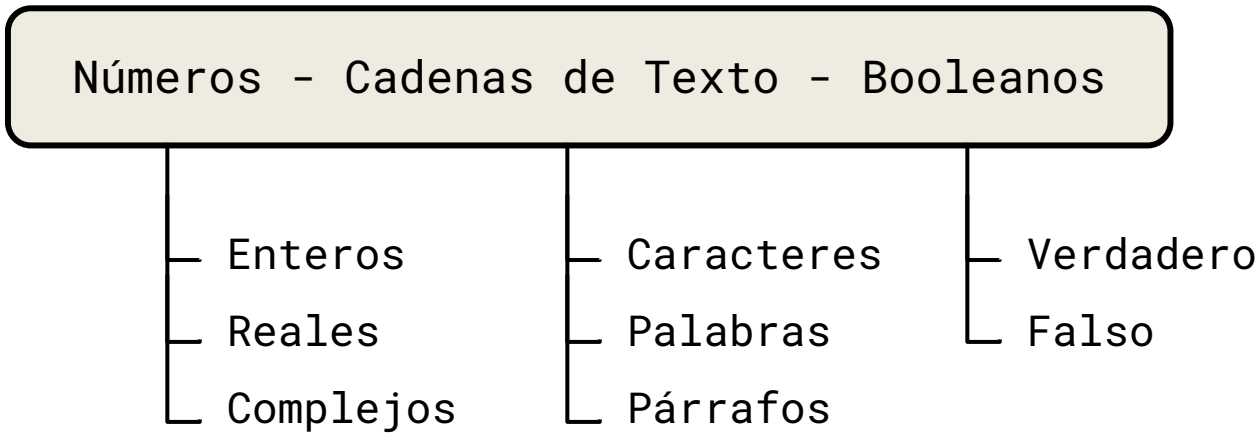
Esta asignación generará un error, porque no están definidos los valores de las variables A y B

name 'A' is not defined

Tipos de Datos en Python

Un tipo de dato es un patrón que determina el conjunto de valores que pueden tomar las variables asociadas a dicho tipo.
También define la representación interna de los datos y las operaciones definidas para ese tipo.

Tipos de Datos en Python



Números

Enteros (int)

Reales (float)

Complejos (complex)

```
entero = 2
salida  2

real2 = 3.2
salida  3.2

real3 = entero + real2
salida  5.2

complejo1 = 1 + 2j
salida  (1+2j)

complejo2 = complejo1 + real3
salida  (6.2+2j)
```

Operadores aritméticos

Operador	Ejemplo
Suma: +	1 + 2 = 3
Resta: -	5 - 3 = 2
Multiplicación: *	5 * 5 = 25
División comun (con decimales): /	10 / 4 = 2.5
División entera (sin decimales): //	10 // 4 = 2
Potencia: **	2 ** 3 = 8
Modulo de la division: %	10 % 2 = 0

Cadenas de texto (strings)

Las cadenas de texto están representadas con comillas simples, dobles, triples comillas simples o triples comillas dobles

```
'Hola soy un string'
"Hola soy un string"
'''Hola soy un string'''
"""Hola soy un string"""
```

Cadenas de texto (strings)

Esto quiere decir que el número 1 y el “1”, NO SON IGUALES, porque el 1 es un número entero, y el “1” es una cadena de texto o bien llamado string.

```
1 + “1” nos va a generar un error.
```

Cadenas de texto (strings)

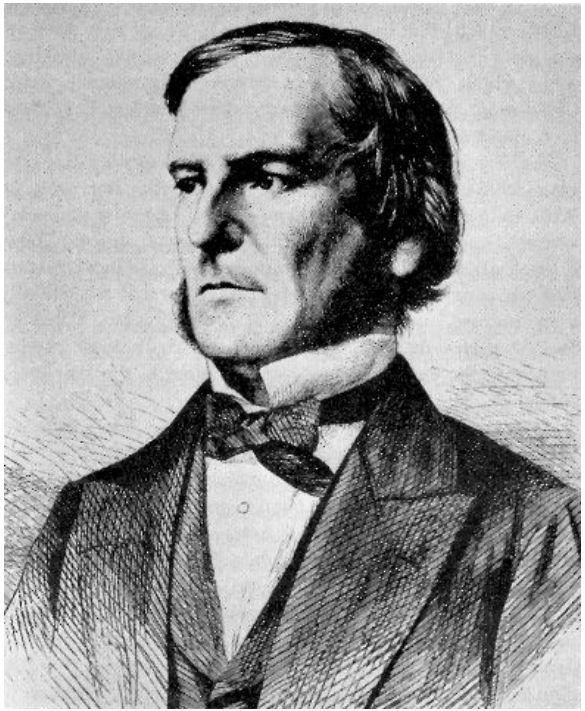
No se pueden “sumar” strings de forma literal. Esa operación se conoce como **concatenación**, lo cual significa unir dos cadenas de caracteres distintas.

```
1 + “1” nos va a generar un error

“1” + “1” nos devuelve “11”

“Hola soy” + “Pedro” nos devuelve “Hola soyPedro”

“Hola” * 3 nos devuelve “HolaHolaHola” (concatena 3 veces)
```

Booleanos

Algebra booleana

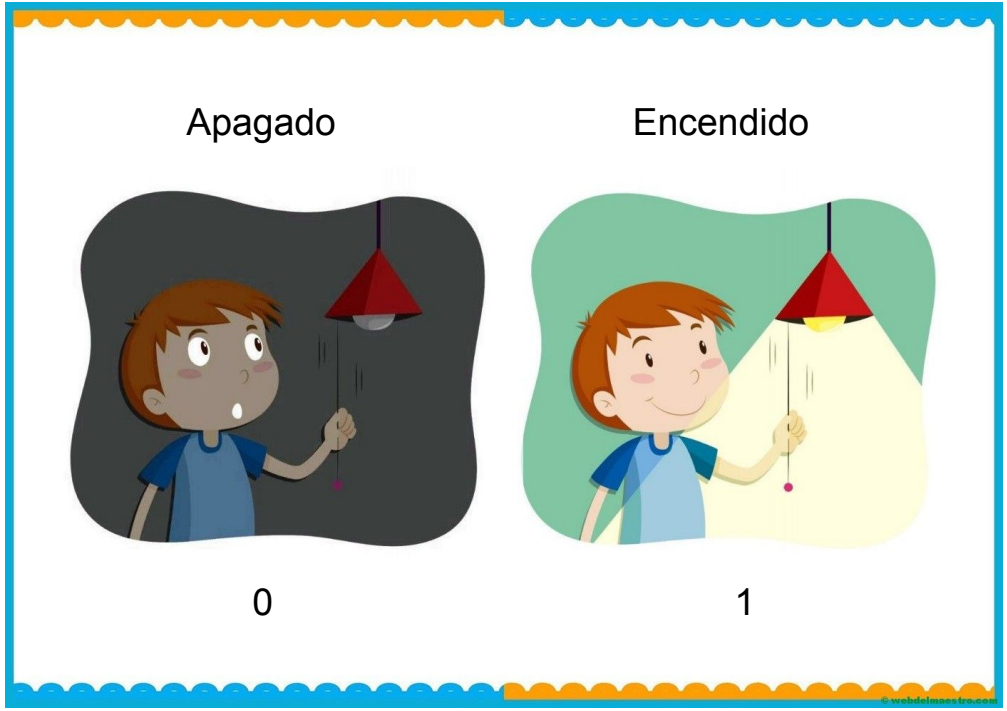
George Boole fue un matemático y lógico británico. Como inventor del álgebra de Boole, que marca los fundamentos de la aritmética computacional moderna, es considerado como uno de los fundadores del campo de las ciencias de la computación.



Booleanos

Planteo de George Boole

George planteaba que existen dos estados, que se los traducía como “verdadero” o “falso”, o mejor dicho 1 y 0 (uno y cero).



Booleanos

```
el verdadero es igual a True
el falso es igual a False
el 1 es también igual a True
el 0 es igual a False
```

```
1 <- True
```

```
0 <- False
```

Booleanos: Tablas de verdad (AND)

Tabla del AND (&&, and, ^)		
Primera entrada	**Segunda entrada**	**Resultado**
Verdadero (V)	Verdadero (V)	Verdadero (V)
Verdadero (V)	Falso (F)	Falso (F)
Falso (F)	Verdadero (V)	Falso (F)
Falso (F)	Falso (F)	Falso (F)

Booleanos: Tablas de verdad (OR)

Tabla del OR (||, or, ∨)		
Primera entrada	**Segunda entrada**	**Resultado**
Verdadero (V)	Verdadero (V)	Verdadero (V)
Verdadero (V)	Falso (F)	Verdadero (V)
Falso (F)	Verdadero (V)	Verdadero (V)
Falso (F)	Falso (F)	Falso (F)

Operadores lógicos

Operador	Ejemplo
&&, and, ^	Verdadero ^ Verdadero = Verdadero
||, or, v	Verdadero v Falso = Verdadero
NOT, not, !	not Verdadero v Falso = Falso

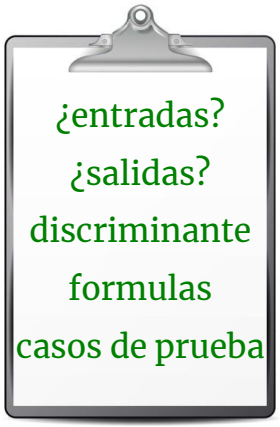
Operadores relacionales

Operador	Ejemplo
Menor que (estricto): <	1 < 2: Verdadero | 1 > 2: Falso
Menor o igual que: <=	5 <= 5: Verdadero | 10 <= 5: Falso
Mayor que (estricto): >	3 > 2: Verdadero | 1 > 10: Falso
Mayor o igual que: >=	200 >= 200: Verdadero | 10 >= 15: Falso
Distinto: !=	10 != 1: Verdadero | 5 != 5: Falso
Iguales: ==	2 == 2: Verdadero | 123 == 321: Falso

Tipos de Datos en Python

`Números - Cadenas de Texto - Booleanos`

Ejemplo 1: Diseñe un programa que calcule y muestre las raíces de una ecuación cuadrática, indicando previamente si son reales o complejas.



$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Solución al Ejemplo 1

```
import math                                                    #librería math

a=int(input("ingrese coeficiente cuadratico: "))               #ingreso de números por teclado
b=int(input("ingrese coeficiente lineal: "))
c=int(input("ingrese termino independiente: "))

mensaje1 = "raices reales"                                     #TEXTO

d=b*b-4*a*c                                                    #NÚMERO REAL
res = (d>=0)                                                   #BOOLEAN

if (res):
  print(mensaje1)
  x1=(-b+(math.sqrt(d)))/(2*a)                                 #NÚMERO REAL
  x2=(-b-math.sqrt(d))/(2*a)                                   #NÚMERO REAL
else:
  print("raices imaginarias")
  x1= complex(-b/(2*a),math.sqrt(-d)/(2*a))                    #NÚMERO COMPLEJO
  x2= complex(-b/(2*a),-math.sqrt(-d)/(2*a))                   #NÚMERO REAL

print("Los valores de las raices son r1: ",x1,"r2: ",x2)
```

Tipos de Datos en Python

Listas - Tuplas - Diccionarios - Conjuntos

```
numeros_lista = [1, 2, 3, 4, 5]
salida  [1, 2, 3, 4, 5]
```

```
numeros_conjuntos = {1, 3, 2, 9, 3, 1}
salida {1, 2, 3, 9}
```

```
numeros_tupla = 1, 2, 3, 4, 5
salida  (1, 2, 3, 4, 5)
```

```
numeros_diccionario = {1:1, 2:2, 3:3}
salida {1: 1, 2: 2, 3: 3}
```

Listas (list)

Las listas en Python son una colección o estructura de datos en la cual contiene distintos tipos de datos en su interior, ordenados o no, de forma secuencial, los cuales diferencia a través de índices o posición.

SON MUTABLES

```
["Hola", "soy", "una", "lista"]
```

Composición de una lista

Las listas en Python son dinámicas, osea que no es necesario indicarles un tamaño fijo, ni que tipos de datos puede contener dentro de ella.
Cada valor dentro de la lista se conoce como “elemento”, y cada elemento tiene una posición.

Cantidad de elementos: 5

```
["Hola", 12, 5.0, True, False]
```

{ 0	1	2	3	4 }
-1	-2	-3	-4	-5

Posiciones

Slicing (“rebanar”)

El slicing es una característica de las estructuras de datos que nos permite acceder a una “porción” definida de ellas.

```
lista = ["Hola", True, False, 12]

lista[inicio:limite]
```

```
lista = ["Hola", True, False, 12]

lista[:]  # ["Hola", True, False, 12]
lista[0]  # "Hola"
lista[1]  # True
lista[2]  # False
lista[3]  # 12
lista[0:] # ["Hola", True, False, 12]
lista[1:3]  # [True, False]
lista[-1]  # 12
lista[-1:-3] # [12, False]
```

Métodos de Listas

```python
lista = ["Hola", True, False, 12]

# Agregar
lista.append("Python")  # ["Hola", True, False, 12, "Python"]

# Quitar
lista.remove("Hola")  # [True, False, 12, "Python"]

# Obtener el indice de un elemento
lista.index(12)  # Devuelve 2, porque esta en la posicion 2 de la lista

# Insertar segun una posicion
lista.insert(0, "Hola")  # Agregamos el elemento "Hola" en la posicion cero, como
estaba antes, por lo tanto nos queda ["Hola", True, False, 12, "Python"]
```

Operaciones con las listas

```
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]
lista_concatenada = lista1 + lista2  # Resultado: [1, 2, 3, 4, 5, 6]
```

Tuplas (tuple)

Las tuplas en Python son una colección de datos al igual que una lista, pero con la diferencia que son inmutables, esto quiere decir que no podemos modificar la tupla, ni quitar ni agregar nuevos datos, es una “estructura fija”, una vez creada, no se puede modificar.

```
(“Hola”, “soy”, “una”, “tupla”)
```

Diferencia entre la tupla y la lista

[ ,  , ]

( ,  , )

¿Chicos, puedo entrar?

No hay drama

Dele con fe

[ ,  , ]

:)

[ ,  ,  , ]

Dale, nos vemos

Chicos me tengo que ir

[ ,  ,  , ]

Avise cuando llegue

Vaya por la sombra

La posta que si

[ ,  , ]

Un tipazo

¿Chicos, puedo entrar?

( ,  , )

:(

Conjuntos (set)

Los conjuntos en Python son una estructura de datos NO determinista, esto quiere decir que automáticamente todos sus elementos van a ser ordenados, sin importar en el orden en que se crean, siempre se van a terminar ordenando de menor a mayor. Tambien son mutables.

```
{'soy', 'Hola', 'conjunto', 'un'}
```

Conjuntos (set)

{  ,  ,  }

{  ,  ,  }

Diccionarios (dict)

Los diccionarios en Python son una estructura de datos en los cuales están representados por “clave” y “valor”, la misma lógica que la de un diccionario, además es NO determinista.

```
k     v
↓     ↓
{clave: valor}
```

```
{“Hola”: “soy”, “un”: “diccionario”}
```

Ejemplo 2: Diseñe un diccionario que describa cuáles días de la semana son laborables y cuáles no. Muestre los días laborables. Luego cree y muestre una lista con los días antes mencionado

definición del diccionario

```
dias_semana = dict()

dias_semana = {
    "Lun" : "Trabajo",
    "Mar" : "Trabajo",
    "Mie" : "Trabajo",
    "Jue" : "Trabajo",
    "Vie" : "Trabajo",
    "Sab" : "Fiesta",
    "Dom" : "Fiesta"
}
```

Ejemplo 2: Diseñe un diccionario que describa cuáles días de la semana son laborables y cuáles no. Muestre los días laborables. Luego cree y muestre una lista con los días antes mencionado

```
for k,v in dias_semana.items():
  if (v=="Trabajo"):
    print(k)
```

salida

```
Lun
Mar
Mie
Jue
Vie
```

dias_semana.items()
representa la lista de todas tuplas formadas por los pares clave:valor

v=="Trabajo"
separa el valor deseado

print(k)
imprime la clave de las tuplas que cumplen la condición

Ejemplo 2: Diseñe un diccionario que describa cuáles días de la semana son laborables y cuáles no. Muestre los días laborables. Luego cree y muestre una lista con los días antes mencionado

```
lista=list()

for k,v in dias_semana.items():
 if (v=="Trabajo"):
   lista.append(k)
print(lista)
```

salida

```
['Lun', 'Mar', 'Mie', 'Jue', 'Vie']
```

`lista= list()`
inicializa una lista vacía

`lista.append(k)`
agrega elementos a la lista, en este caso, los valores de clave del diccionario que cumplen la condición

`print(lista)`
imprime la lista de claves creada

Acceder a un valor del diccionario

```
valores = dias_semana.values()
```

salida

```
dict_values(['Trabajo', 'Trabajo', 'Trabajo', 'Trabajo',
'Trabajo', 'Fiesta', 'Fiesta'])
```

```
elemento = dias_semana.get("Vie")
```

salida

```
Trabajo
```