INFORMATORIO

Hacia una mejor industria informática

INFORMATORIO

Juan Ignacio Campias

Data Engineer

campiasjuan@gmail.com.ar

https://www.linkedin.com/in/campiasjuan/

corebi

NowVertical COMPANY

INFORMATORIO

¿Qué es la programación?

- La programación es una forma de ejecutar un **algoritmo**.
- Un algoritmo es una **secuencia de pasos** que lleva a un **resultado**.
- Una **receta** es un algoritmo.
- Si se sigue el algoritmo, se llega al **resultado**.

Python, Open Source: componentes

1

Intérprete

programa intérprete, traductor a binario.

2

IDE

entorno de desarrollo, lugar donde escribiremos código.

3

Paquetes

conjuntos de funciones pre-armadas para problemas habituales.

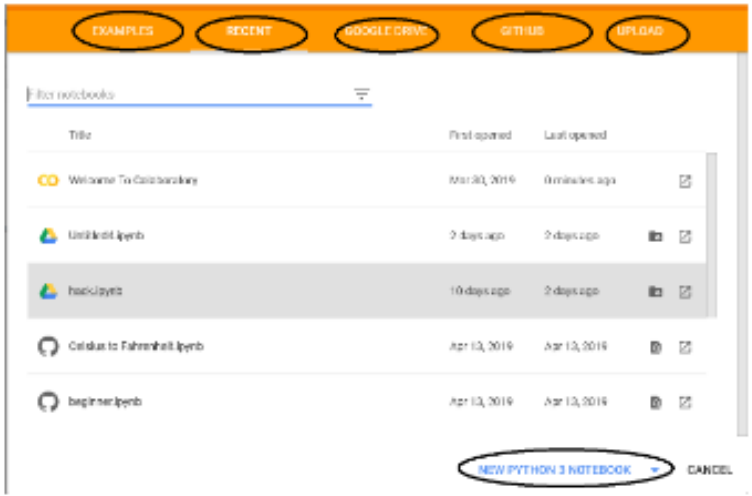
Google Colab

Permite trabajar en un entorno no local y la creación de Notebooks

Es un producto de Google Research.
Está especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación.
Jupyter es el proyecto de código abierto en el que se basa Colab.
Nos permite compartir notebooks sin la necesidad de descargar ningún software extra.
El código se ejecuta en una máquina virtual dedicada a tu cuenta y pueden eliminarse luego de cierto tiempo.

INFORMATORIO

Cómo usar Google Colab

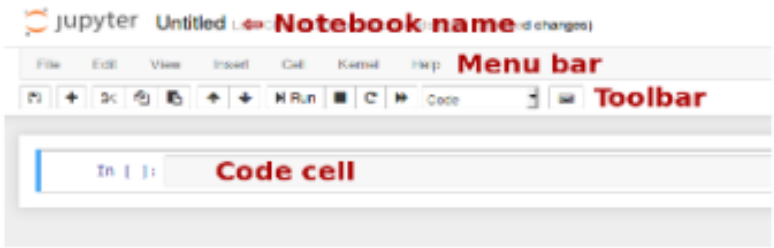


Ir al siguiente enlace:
https://colab.research.google.com

EXAMPLES: Contiene ejemplos de Jupyter notebooks con diversos ejemplos.
RECENT: Jupyter notebooks que has trabajado recientemente.
GOOGLE DRIVE: Jupyter notebooks en tu google drive.
GITHUB: Puedes añadir Jupyter notebooks desde Github pero es necesario conectar Colab con GitHub.
UPLOAD: Si deseas subir un Jupyter notebook desde tu equipo local.

Podemos encontrar 4 partes principales:

1. Nombre del notebook (termina con extensión .ipynb)
2. Barra de menú: Permite ejecutar código y opciones genéricas
3. Toolbar: Permite ejecutar celdas de código, guardar, añadir, borrar, cortar o pegarlas
4. Celdas de Código: Pueden ser Markdown (texto) o Código Python



Programa y computadora

- La computadora nació para **resolver cálculos**.
- La programación es un **complemento** para la computadora.
- Es una forma de que la computadora **entienda el funcionamiento de un algoritmo** y lo ejecute.
- La computadora entiende **ceros y unos (lenguaje binario)**, nosotros no.
- Por lo tanto, un programa **traduce un lenguaje humano a lenguaje binario**.

Programación y lenguajes

No existe **un solo lenguaje** que solucione **todos** los problemas

Cada lenguaje resuelve **un conjunto de problemas posibles:** Empresariales, Web, Ciencia, Salud, etc.

Para Data Science, existen algunos lenguajes que funcionan muy bien: **Python, R, Julia y Scala** son algunos de ellos.

Python

R

Julia

Scala

¿Interpretado o Compilado?

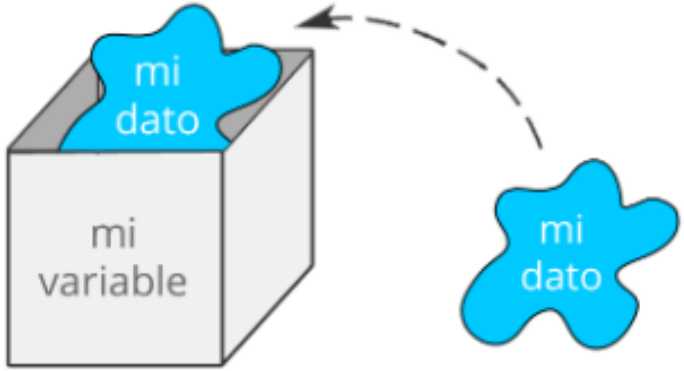
Python es un lenguaje interpretado, esto quiere decir que:

- ✔ Usa un programa intérprete que traduce en tiempo casi real nuestras órdenes a binario.
- ✔ La traducción se hace línea por línea.
- ✔ Podemos probar código “de a pedacitos”.
- ✔ El lenguaje compilado se traduce todo junto al final.

Variables

Las variables se utilizan para almacenar información para ser referenciada y manipulada en un programa de computadora. Proporcionan una forma de etiquetar los datos con un nombre descriptivo, para que los programas puedan ser entendidos con mayor claridad.

Es útil pensar en las variables como contenedores de información.
Su único propósito es etiquetar y almacenar datos en la memoria.



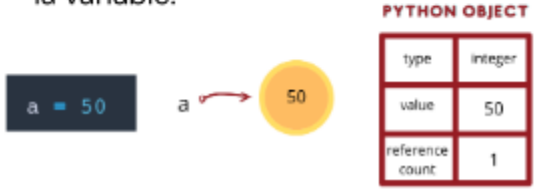
Asignación

Nombrar variables es una tarea compleja.
Cuando nombre variables, piense detenidamente en los nombres (Comprensible).
La asignación de lleva a cabo por medio del símbolo =
El nombre de la variable va a la izquierda y el valor que desea almacenar en la variable va a la derecha.

Asignación

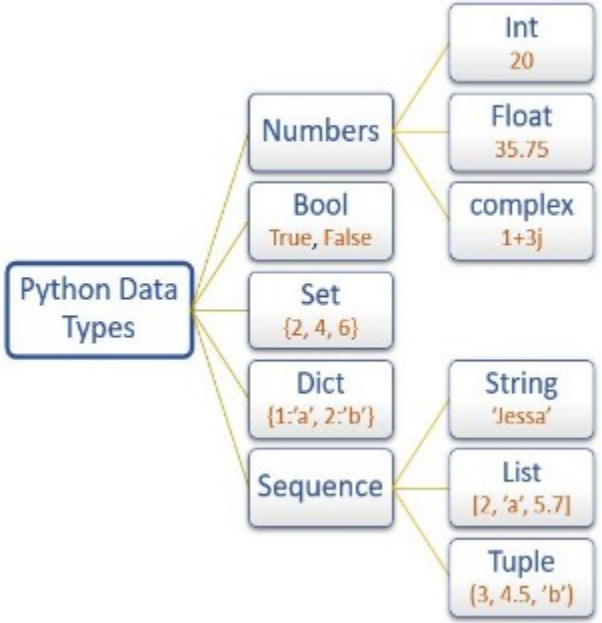
Reglas para asignación de variables en Python

- ✔ El nombre de una variable debe comenzar con una letra o el carácter de subrayado.
- ✔ Un nombre de variable no puede comenzar con un número.
- ✔ Un nombre de variable solo puede contener caracteres alfanuméricos y guiones bajos (A-z, 0-9 y _).
- ✔ Los nombres de las variables distinguen entre mayúsculas y minúsculas (nombre, Nombre y NOMBRE son tres variables diferentes).
- ✔ Las palabras reservadas (palabras clave) no se pueden usar para nombrar la variable.



Tipo de dato

- ✓ Define qué tipos de operaciones se puede hacer con él. Por ejemplo, un número se puede sumar, pero un texto no.
- ✓ Python define dos grandes grupos de tipos de datos: simples y estructurados.
- ✓ Podemos saber el tipo de un dato x con la función type(x)



Operadores aritméticos

Los **operadores aritméticos** son directamente operaciones matemáticas estándar.

Aritméticos	
a + b	Suma
a - b	Resta
a * b	Multiplicación
a / b	División
a // b	División entera (resultado sin decimal)
a % b	Módulo (resto de la división entera)
a ** b	Exponenciación
-a	Negativo

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

INFORMATORIO

Operadores de asignaciones

Los asignadores simplifican operadores aritméticos comunes.

Asignaciones	
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a //= b	a = a // b
a %= b	a = a % b
a **= b	a = a ** b

Operadores de comparación

Los comparadores dan resultados lógicos **(si/no, true/false)**

Comparadores	
a == b	a igual a b
a != b	a distinto de b
a < b	a menor a b
a > b	a mayor a b
a <= b	a menor o igual que b
a >= b	a mayor o igual que b

INFORMATORIO

20 minutos a las salas:
Resolución de ejercicios

¿Quién soy?
¿A qué me dedico?
¿Qué puedo aportar al grupo?

INFORMATORIO

Estructuras de control: FOR, WHILE, IF

INFORMATORIO

¿Qué son y para qué sirven?

Las estructuras de control sirven para **dar claridad y orden al código.**

Si hay que hacer operaciones repetitivas, estas estructuras nos ayudan a organizarlas.

Las estructuras de control más comunes son:

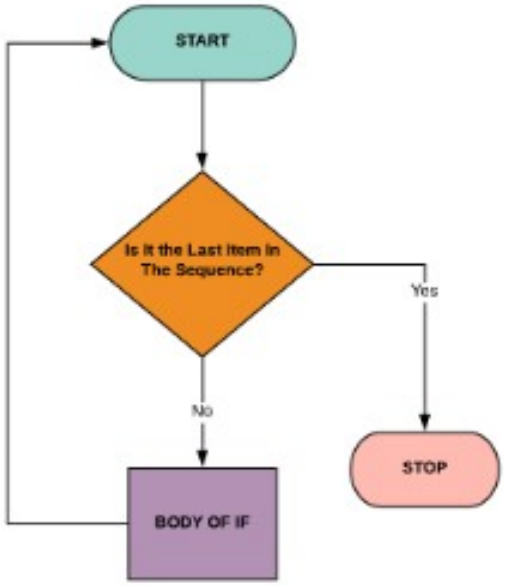
- For
- While
- If

Estructura FOR

- ✓ Repite un comando **una cantidad fija** de veces

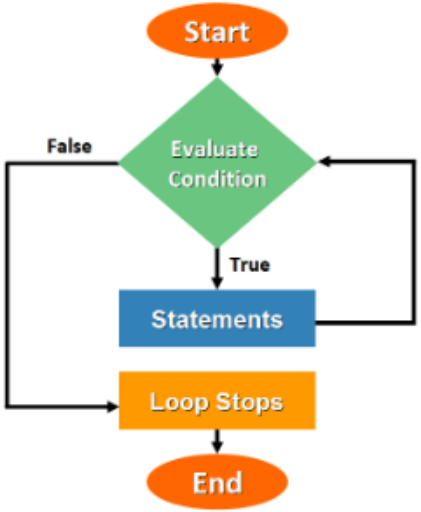
```
for i in range(1,10):
    print(i)              # muestra los números del 1 al 9

for i in [1,4,6,2]:
    print(i)              # muestra los números de la lista
```



Estructura WHILE

Repite una secuencia de comandos **"mientras" una condición se cumpla**. Cuando la condición no se cumple más, termina la repetición.




```
i = 1
while i < 10:  # el código luego de los dos puntos se ejecuta
    print(i)   # mientras i es menor a 10.
    i += 1         # cuando i llega a 10 termina la ejecución
```

Estructura condicional (IF)

- ✓ **Si se cumple** una condición, se ejecuta una secuencia de comandos. **En otro caso**, se ejecuta otra.
- ✓ Pueden manejarse **más de dos opciones**.

```
x = 1
if x < 10:                                  # Pregunto si x es menor a 10
    print(x, "es menor a 10") # Si es así muestro mensaje
elif x > 10:                              # Si no es así, pregunto si x es
mayor
    print(x, "es mayor a 10") # a 10 y si es así muestro mensaje
else:                                       # Si nada de lo anterior se
    print(x, "es 10")                 # cumple, ejecuto esto
```

Estructura condicional (IF)

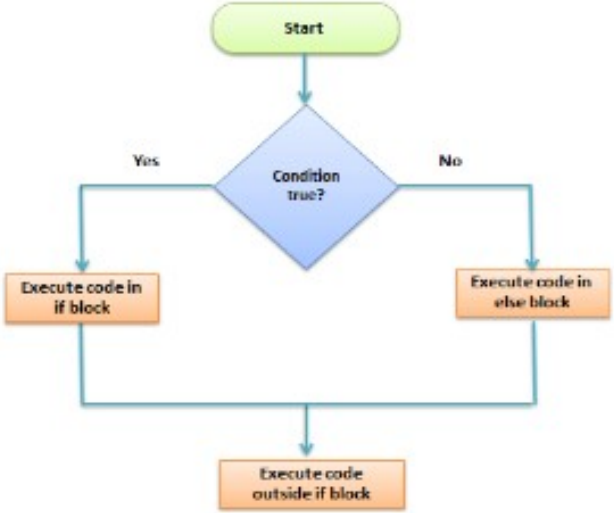
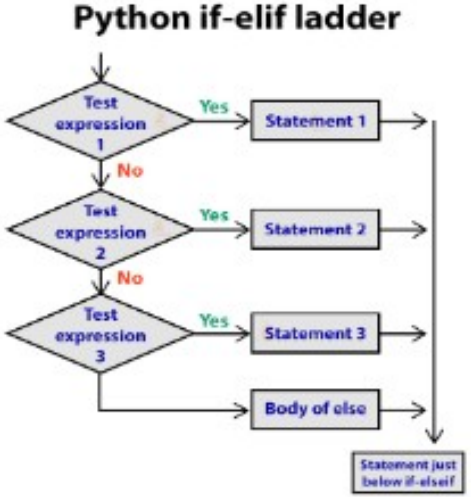
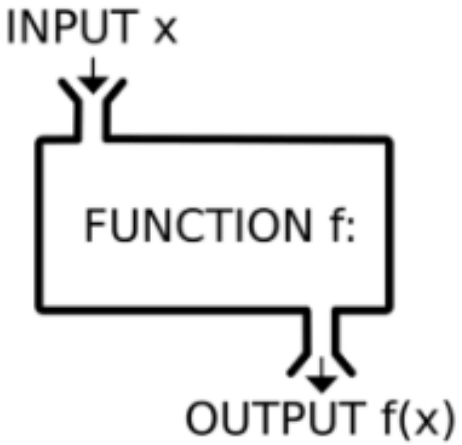


Diagrama de flujo condicionales



Estructura if-elif-else Python



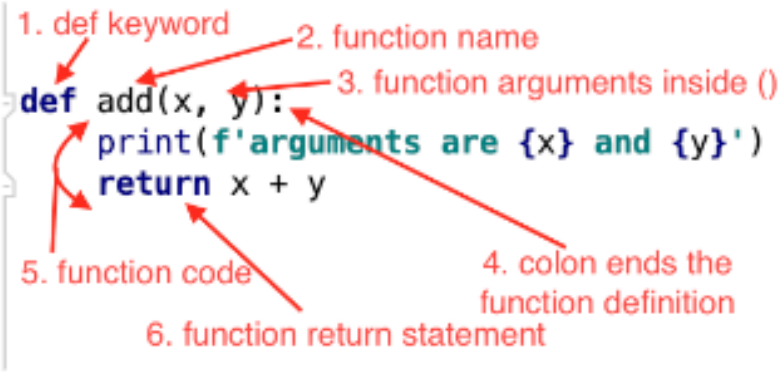
Funciones

- ✓ Para trabajar profesionalmente en programación, el código que se usa en forma repetitiva se organiza en funciones.
- ✓ Puede hacerse una analogía con una función matemática y = f(x): la función f recibe un argumento x, ejecuta una serie de comandos y devuelve un valor y.

Argumentos y retorno

Las funciones tienen al menos 3 elementos:

El nombre de la función
Cero o más argumentos (variables de entrada)
Un valor de retorno (salida de la función)



INFORMATORIO

Ejemplo

```
def suma(x,y):    # Aquí definimos una función “suma”.
    z = x + y     # Esto es lo que pide el ejercicio
    return z

res = suma(2,3)   # Aquí probamos la función suma con dos números concretos
                  # Esta es la prueba para verificar que el código funciona
print (res)
```