

Fundamentación

Lenguaje Python: Introducción

Sergio A. Cantillo
`sacantillo@uao.edu.co`

Facultad de Ingeniería y Ciencias Básicas
Universidad Autónoma de Occidente

Agenda

- 1 Generalidades de Python
 - ¿Qué es Python?
 - Ambientes de desarrollo
 - Google Colab

Agenda

- 1 Generalidades de Python
 - ¿Qué es Python?
 - Ambientes de desarrollo
 - Google Colab
- 2 Sintaxis
 - Variables y Operadores
 - Estructuras de Control
 - Estructuras de Datos
 - Funciones

Agenda

- 1 Generalidades de Python
 - ¿Qué es Python?
 - Ambientes de desarrollo
 - Google Colab
- 2 Sintaxis
 - Variables y Operadores
 - Estructuras de Control
 - Estructuras de Datos
 - Funciones
- 3 Instalación

Agenda

- 1 Generalidades de Python
 - ¿Qué es Python?
 - Ambientes de desarrollo
 - Google Colab

- 2 Sintaxis
 - Variables y Operadores
 - Estructuras de Control
 - Estructuras de Datos
 - Funciones

- 3 Instalación

Python

¿Qué es?

Lenguaje de programación multiplataforma, multiparadigma (POO, funcional), de tipado dinámico y estructural

Características

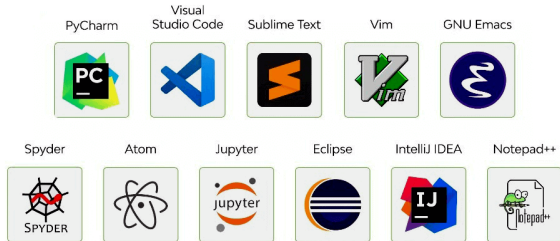
Sencillo
Libre y de fuente abierta
Lenguaje de alto nivel
Ampliable
Comunidad activa
Portable
INDENTADO



¿Qué desventajas tiene? "Lentitud"

Ecosistema fragmentado

Ambientes de desarrollo



IDE's (Integrated Development Environment)

- Aplicaciones informáticas que proporcionan servicios integrales al desarrollador o programador.
- Son editores de código fuente con herramientas de construcción automáticas y depurador.
- Incluyen: Autocompletado de código, compilador, interprete, herramientas para GUI (interfaz gráfica).

Uso de Python

Formas de Uso

Se tienen dos opciones para utilizar el lenguaje Python para el desarrollo de aplicaciones de inteligencia artificial:

- Instalacion Local.
- **Google Colaboratory** (requiere cuenta de Google y conexión a Internet)



<https://colab.research.google.com/>

Características

- Ambiente de *Jupyter Notebook* que no requiere ninguna configuración para su uso.
- Ambiente de programación interactivo **basado en la web**.
- Es de Libre Acceso (Existe una Version Gratuita)
- Esta compuesto por *celdas* (Entrada/Salida)
- El formato de archivo de un documento de Jupyter Notebook es JSON (.ipynb)
- Cada celda puede contener código, texto, ecuaciones matemáticas, graficos y texto enriquecido.

Google Colab (II)

The screenshot displays the Google Colaboratory web interface. On the left, a file explorer sidebar is open, showing a directory structure with a folder named 'sample_data'. A red box highlights this sidebar, with a red line pointing to the label 'Navegador de Archivos' below it. The main workspace contains several code cells, each starting with '[] 1'. A blue box highlights the first code cell, with a blue line pointing to the label 'Celdas de Código/texto' below it. The top of the interface shows the browser address bar with the URL 'colab.research.google.com' and the document title 'Untitled26.ipynb - Colaboratory'. The bottom of the interface shows a taskbar with various application icons and a system clock indicating 12:30 PM.

Navegador de Archivos

Celdas de Código/texto

Agenda

- 1 Generalidades de Python
 - ¿Qué es Python?
 - Ambientes de desarrollo
 - Google Colab

- 2 Sintaxis
 - Variables y Operadores
 - Estructuras de Control
 - Estructuras de Datos
 - Funciones

- 3 Instalación

Sintaxis: Elementos Básicos

General

- Python usa espacios en blanco para diferenciar bloques de código.
- La indentación (asociado a la tecla Tab) es la única forma de estructurar sentencias de control.
- No existe ningún símbolo para indicar el final de una línea de código o instrucción.
- Una buena practica es activar la numeración de líneas en el editor de código (seguimiento de errores).



Palabras Reservadas

Definición

Palabras especiales que representan estructuras, ciclos, comparaciones, excepciones, estructuras de control, entre otras funciones para un lenguaje de programación). **No pueden ser usadas como nombres de variables.**

Usualmente se diferencian del resto de palabras (Color distinto).

| | | | | |
|--------|----------|---------|----------|--------|
| False | class | finally | is | return |
| None | continue | for | lambda | try |
| True | def | from | nonlocal | while |
| and | del | global | not | with |
| as | elif | if | or | yield |
| assert | else | import | pass | |
| break | except | in | raise | |

Y con Python 3.7:
async
await

Tipos de Datos

Cadenas (Strings)

Tipo de dato cuyo contenido admite **secuencias de caracteres alfanuméricos**.

Ejemplo

'Sophus Lie', "Hello World 2023!!"

Numéricos

Tipos de dato cuyo contenido admite distintos tipos de **valores numéricos (enteros y reales)**.

Ejemplo

92.0 (float, Real), 92 (int, Entero)

Booleanos

Tipo de dato cuyo contenido admite únicamente **información lógica (F/V)**.

Ejemplo

True, False

Asignación

Sentencia de Asignación

identificador = expresión/valor

Ejemplos

Asignación simple:

estatura = 1.70

Asignación multiple:

a, b, c = 1, 2, 3

Asignación de valor vacío:

a = None

Variables y Operadores

Todo lenguaje de programación provee:

- **Variables:** Método para almacenar y manipular datos que pueden tomar diferentes valores en diferentes momentos de la ejecución del código.
- **Operadores:** Forma de modificar/manipular las variables.

Ejemplos

```
x = 2  
five = 5  
z = 3.14159  
my_string = "Hello World!"
```


Variables y Operadores

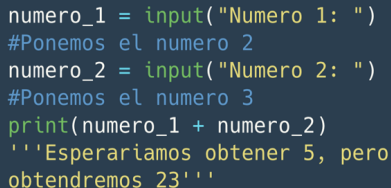
| Symbol | Operation | Example |
|--------|----------------|--------------|
| + | Addition | $2 + 2 = 4$ |
| - | Subtraction | $4 - 1 = 3$ |
| / | Division | $9/3 = 3$ |
| % | Modulo | $7 \% 2 = 1$ |
| * | Multiplication | $8 * 6 = 48$ |
| // | Floor division | $7//2 = 3$ |
| ** | Exponential | $3 ** 2 = 9$ |

Jerarquía de Operaciones

| | |
|-------------|--|
| () | Items enclosed in parentheses |
| ** | Exponentiation |
| *, /, %, // | Multiplication, division, modulo, floor division |
| +, - | Addition and subtraction |
| = | Assignment |

Entrada/Salida de datos

- Para ingresar o solicitar datos a un usuario, se utiliza la función *input*. Esta interrumpe la ejecución del código hasta que el usuario ingrese el valor esperado y pulse la tecla enter. Se recupera un dato de tipo **string**.
- Para imprimir datos (tipo string, int y float) se utiliza la función *print*.



```
numero_1 = input("Numero 1: ")
#Ponemos el numero 2
numero_2 = input("Numero 2: ")
#Ponemos el numero 3
print(numero_1 + numero_2)
'''Esperariamos obtener 5, pero
obtendremos 23'''
```

Conversión entre tipos de datos

Es posible la conversión entre tipo de datos, siempre y cuando exista **compatibilidad** en la información a convertir. Estas son:

- Cadena de caracteres a numeros enteros (**int()**) y viceversa (**str()**).
- Cadena de caracteres a numeros reales (**float()**) y viceversa (**str()**).

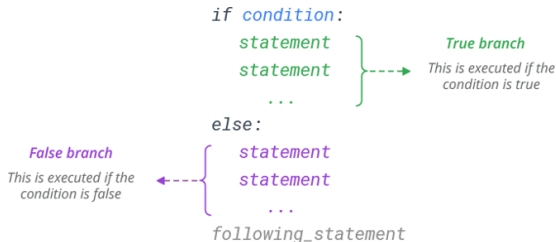
Ejemplos

```
str(19)
str(107.67)
float('3.1415')
int("107")
valor = float('3.1')
```

Estructuras de Control: Decisión

If - Else

Estructuras que permiten la ejecución de ciertas instrucciones al cumplir una condición, y otras instrucciones cuando no se cumple. En inglés *if* significa **si (condición)**, y *else* significa **sino**.



```
x = 2
if x == 2:
    print "x igual dos!"
else:
    print "x no es igual a dos."

x igual dos!
```

Ejercicio 1

Realice un programa que determine si el número entero ingresado por el usuario es par o no.

Estructuras de Control: Decisión

Ejercicio 1

Realice un programa que determine si el número entero ingresado por el usuario es par o no.

```
1 numero_1 = int(input("Digite el numero:"))
2
3 if numero_1 % 2 == 0:
4     print("el numero es par")
5 else:
6     print("el numero es impar")
```

Ejercicio 2

Escriba un programa que pida dos números enteros e indique si son iguales o cual de los dos es mayor.

Estructuras de Control: Decisión

Ejercicio 2

Escriba un programa que pida dos números enteros e indique si son iguales o cual de los dos es mayor.

```
1 numero_1 = int(input("Digite el primer numero:"))
2 numero_2 = int(input("Digite el segundo numero:"))
3
4 if numero_1 == numero_2:
5     print("los numeros son iguales")
6 elif numero_1 > numero_2:
7     print(numero_1, "es mayor que", numero_2)
8 else:
9     print(numero_2, "es mayor que", numero_1)
```


Estructuras de Control: Ciclos (I)

Ciclo For

Son estructuras de control que repiten un bloque de instrucciones un número **predeterminado y conocido** de veces.

Var

*It takes items from
iterable one by one*

Iterable

*It's a collection of objects
(like a list, tuple etc.)*

for var in iterable:

statement

statement

...

else:

statement

...

following_statement

Loop body

*It is executed once for
each item in iterable*

Else clause

*It is executed if the loop
terminates naturally*

Indentation

*Loop body must
be indented*

```
for x in range(5):  
    print (x)
```

0
1
2
3
4

Estructuras de Control: Ciclos (I)

Ejercicio

Escriba un programa que imprima los números impares del 1 al 99.

Estructuras de Control: Ciclos (I)

Ejercicio

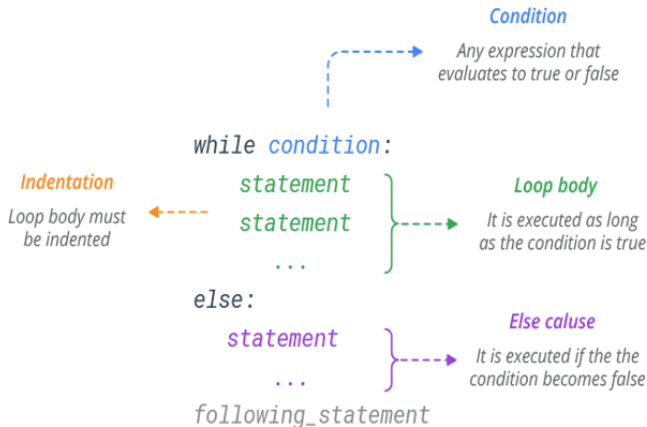
Escriba un programa que imprima los números impares del 1 al 99.

```
1 for i in range (1,100):  
2     if i%2 != 0:  
3         print(i)  
4  
5 for i in range (1,100,2):  
6     print(i)
```

Estructuras de Control: Ciclos (II)

Ciclo While

Son estructuras de control que repiten un bloque de instrucciones, siempre y cuando se **cumpla una condición**.



```
count = 0  
while (count < 5):  
    print (count)  
    count += 1
```

0
1
2
3
4

Estructuras de Control: Ciclos (II)

Ejercicio

Escriba un programa que pida al usuario un numero positivo una y otra vez hasta que el usuario lo haga correctamente

Estructuras de Control: Ciclos (II)

Ejercicio

Escriba un programa que pida al usuario un numero positivo una y otra vez hasta que el usuario lo haga correctamente

```
1 numero_1 = 0
2 while (numero_1 <= 0):
3     numero_1 = int(input("Digite un numero:"))
4     if (numero_1 <= 0):
5         print("Por favor, utilizar solo numeros positivos!")
```

Estructuras de Control: Ciclos (II)

Ejercicio

Escriba un programa que pida al usuario un numero positivo una y otra vez hasta que el usuario lo haga correctamente

```
1 numero_1 = 0
2 while (numero_1 <= 0):
3     numero_1 = int(input("Digite un numero:"))
4     if (numero_1 <= 0):
5         print("Por favor, utilizar solo numeros positivos!")
```

https://www2.eii.uva.es/fund_inf/python/notebooks/00_Introduccion/Introduccion.html

Estructuras de Datos

En Python (base), existen 4 tipos de estructuras de datos para el almacenamiento *masivo* de información:

- **Tuplas:** Colecciones de elementos de solo lectura ("read-only"), es decir *inmutables*.
- **Listas:** Colecciones de elementos *ordenados* que permiten acceso mediante *índices* (Arreglos).
- **Diccionarios:** Colecciones de elementos iterables, sin orden, que permiten acceso mediante un identificador (Llave)
- **Conjuntos:** Al igual que los diccionarios, son colecciones desordenadas de elementos únicos. No poseen llave.

Ejemplos

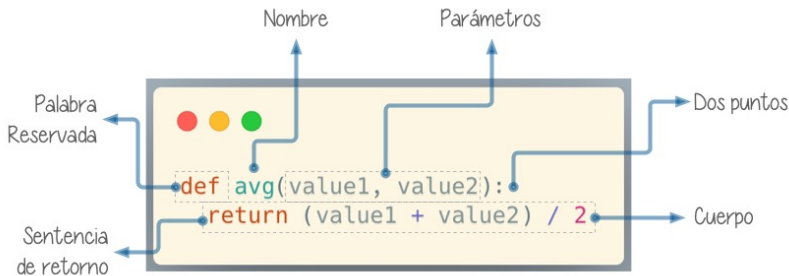
```
a = (1, 2, 3) - Tupla
fruits = [ 'manzana', 'banano', 'cereza' ] - Lista
car_dict = { 'brand': ['Ford', 'Ferrari'], 'model': ['Mustang', 'LaFerrari'], 'año': [1964, 2013] }
numb = {1,2,3,4}
```


Estructuras de Datos

| Aspecto | Listas | Tuplas | Diccionarios | Conjuntos |
|--------------------|---|-----------------------------------|--|---|
| Tipo de Acceso | Por índice | Por índice | Por clave | No se accede por índice |
| Declaración | <code>mi_lista = [1, 2, 3]</code> | <code>mi_tupla = (1, 2, 3)</code> | <code>mi_dict = {'clave': 'valor'}</code> | <code>mi_set = {1, 2, 3}</code> |
| Símbolo | [] | () | { : } | { } |
| Mutabilidad | Mutable | Inmutable | Mutable | Mutable |
| Métodos Comunes | <code>append()</code> , <code>extend()</code> , <code>remove()</code> | Ninguno | <code>keys()</code> , <code>values()</code> , <code>items()</code> | <code>add()</code> , <code>remove()</code> , <code>union()</code> |
| Orden de Elementos | Ordenadas | Ordenadas | No hay orden específico | No hay orden específico |

Funciones

- Bloques de código con instrucciones que se ejecutan **únicamente cuando son llamados**. Se asocian a *labores repetitivas y de responsabilidad única*.
- Los datos de entrada reciben el nombre de *parámetros* o *argumentos*.
- Estas funciones pueden (*return*) o no (*print*) retornar información.



Se puede llamar en cualquier parte del código así:
`promedio = avg(10,20)`

Ejemplo

Escriba una función que determine si un año determinado es bisiesto.

Pista: Un año bisiesto **principalmente** es divisible por 4, no es divisible por 100, y es divisible por 400.

Agenda

- 1 Generalidades de Python
 - ¿Qué es Python?
 - Ambientes de desarrollo
 - Google Colab

- 2 Sintaxis
 - Variables y Operadores
 - Estructuras de Control
 - Estructuras de Datos
 - Funciones

- 3 Instalación

Instalación de Python

Existen dos formas de instalar Python de forma local en un computador, independientemente del sistema operativo:

- Instalación Nativa (Convencional).
- Entornos (ambientes) Virtuales, estos son Usualmente asociado a desarrollos preliminares en IA.

