

# Curso básico de Python

Prof. Andrés Carranza



## Requisitos:

- Una PC o laptop.
- Conexión a Internet.
- No requiere conocimientos previos.
- Muchas, muchas ganas de aprender.

## Normas de clase:

- Estar puntual a la hora citada.
- Si hay alguna consulta, en cualquier momento puede presionar el botón de “levantar la mano”.
- Se pasará lista al iniciar y terminar la clase.
- Si hay algún inconveniente en estar en clase, escribirme por interno.
- Receso: 15 minutos.
- Clases son Lunes y miércoles de 7pm a 10pm.
- Nota mínima para pasar al siguiente módulo: 11

# TEMARIO:

- Introducción a Python.
- Entradas y salidas, Control de flujos y funciones.
- Programación orientada a Objetos, archivos y directorios
- Uso de objetos y módulos.
- Introducción a la programación usando Python.

# Introducción:

- ¿Qué es Python?
- Historia de Python.
- Características.
- Requisitos de Hardware.
- Instalación.
- Uso de IDE.
- PEP 8

# ¿Qué es Python?:

Python es un lenguaje de programación de alto nivel, es decir la sintaxis que se suele usar es fácil de leer para un ser humano, a comparación de otros lenguajes como java y c++, ya que la filosofía del lenguaje es proporcionar una sintaxis muy limpia y que beneficie con código legible.

```
>>> print("Hello World!")  
Hello World!  
>>>
```



## Un poco de historia y curiosidades...

Python fue creado a finales de los ochenta por Guido Van Rossum para las Matemáticas y la Informática en los países bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba.



# Un poco de historia y curiosidades...

El Zen de Python es una colección de 20 principios de software que influyen en el diseño del Lenguaje de Programación Python, de los cuales 19 fueron escritos por Tim Peters en junio de 1999:

- Bello es mejor que feo.
- Explícito es mejor que implícito.
- Simple es mejor que complejo.
- Complejo es mejor que complicado.
- Plano es mejor que anidado.



# Características:

- Lenguaje de alto nivel. Gramática sencilla y legible
- Tipado dinámico y fuerte.
- Orientado a objetos
- Código abierto (Libre y fuente abierta).
- Fácil de aprender.
- Indentado.
- Versátil.
- Incrustable.

Instalación...

# Sintaxis básica:

Las funciones se definen con def

#Ejemplo de código en Python

```
import datetime as dt
```

```
def saludo(nombre):
```

```
    """Función de bienvenida al team DSRP"""
```

```
    print( "Hola" , nombre, "!" )
```

```
    print( "Fecha de registro:" , dt.date.today() )
```

```
    saludos = ["Welcome" , "Bienvenido" , "Bem-vindo"]
```

```
    for saludos in saludos :
```

```
        print(saludo)
```

```
    saludo("Javier")
```

```
    help (saludo)
```

Comentarios de una línea con el numeral (#)

Importación de bibliotecas que necesitamos

No olvidemos los dos puntos que acompañan un: if, def, while, for, etc.

Documentación de la función

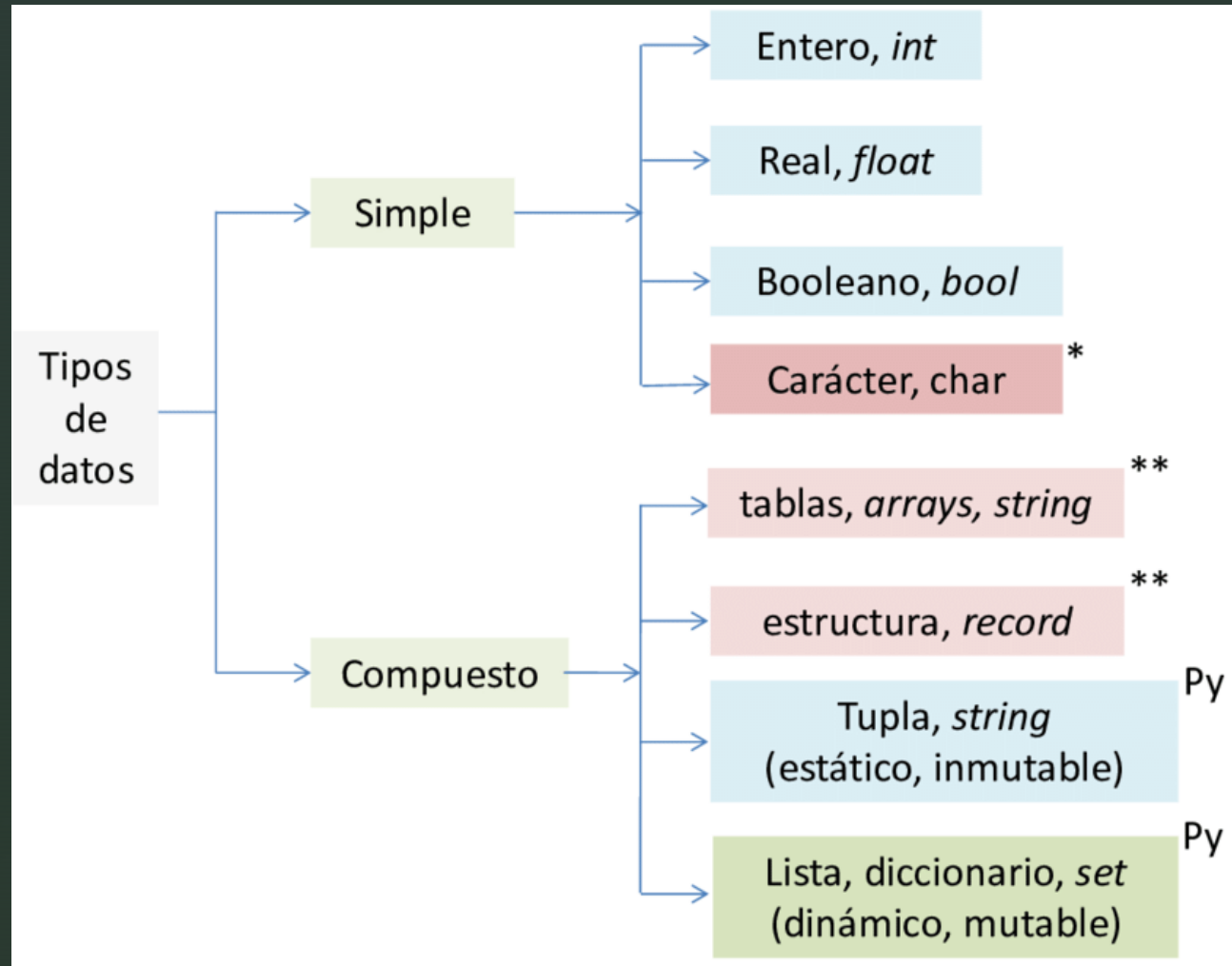
Mostramos nuestras variables en la consola

El bucle for nos indica el recorrido de una variable compuesta (lista, tupla, etc.), también podemos recorrer un rango de elementos usando range().

Llamamos a nuestra función enviando un parámetro de entrada

Pedimos información de la función

# Tipos de datos:



# Tipos de Operadores:

Operador	Descripción	Ejemplo
+	Suma	<pre>&gt;&gt;&gt; 3 + 2 5</pre>
-	Resta	<pre>&gt;&gt;&gt; 4 - 7 -3</pre>
-	Negación	<pre>&gt;&gt;&gt; -7 -7</pre>
*	Multiplicación	<pre>&gt;&gt;&gt; 2 * 6 12</pre>
**	Exponente	<pre>&gt;&gt;&gt; 2 ** 6 64</pre>
/	División	<pre>&gt;&gt;&gt; 3.5 / 2 1.75</pre>
//	División entera	<pre>&gt;&gt;&gt; 3.5 // 2 1.0</pre>
%	Módulo	<pre>&gt;&gt;&gt; 7 % 2 1</pre>

# Tipos de Operadores:

Operador	Descripción	Ejemplo
==	¿son iguales a y b?	<pre>&gt;&gt;&gt; 5 == 3 False</pre>
!=	¿son distintos a y b?	<pre>&gt;&gt;&gt; 5 != 3 True</pre>
<	¿es a menor que b?	<pre>&gt;&gt;&gt; 5 &lt; 3 False</pre>
>	¿es a mayor que b?	<pre>&gt;&gt;&gt; 5 &gt; 3 True</pre>
<=	¿es a menor o igual que b?	<pre>&gt;&gt;&gt; 5 &lt;= 5 True</pre>
>=	¿es a mayor o igual que b?	<pre>&gt;&gt;&gt; 5 &gt;= 3 True</pre>



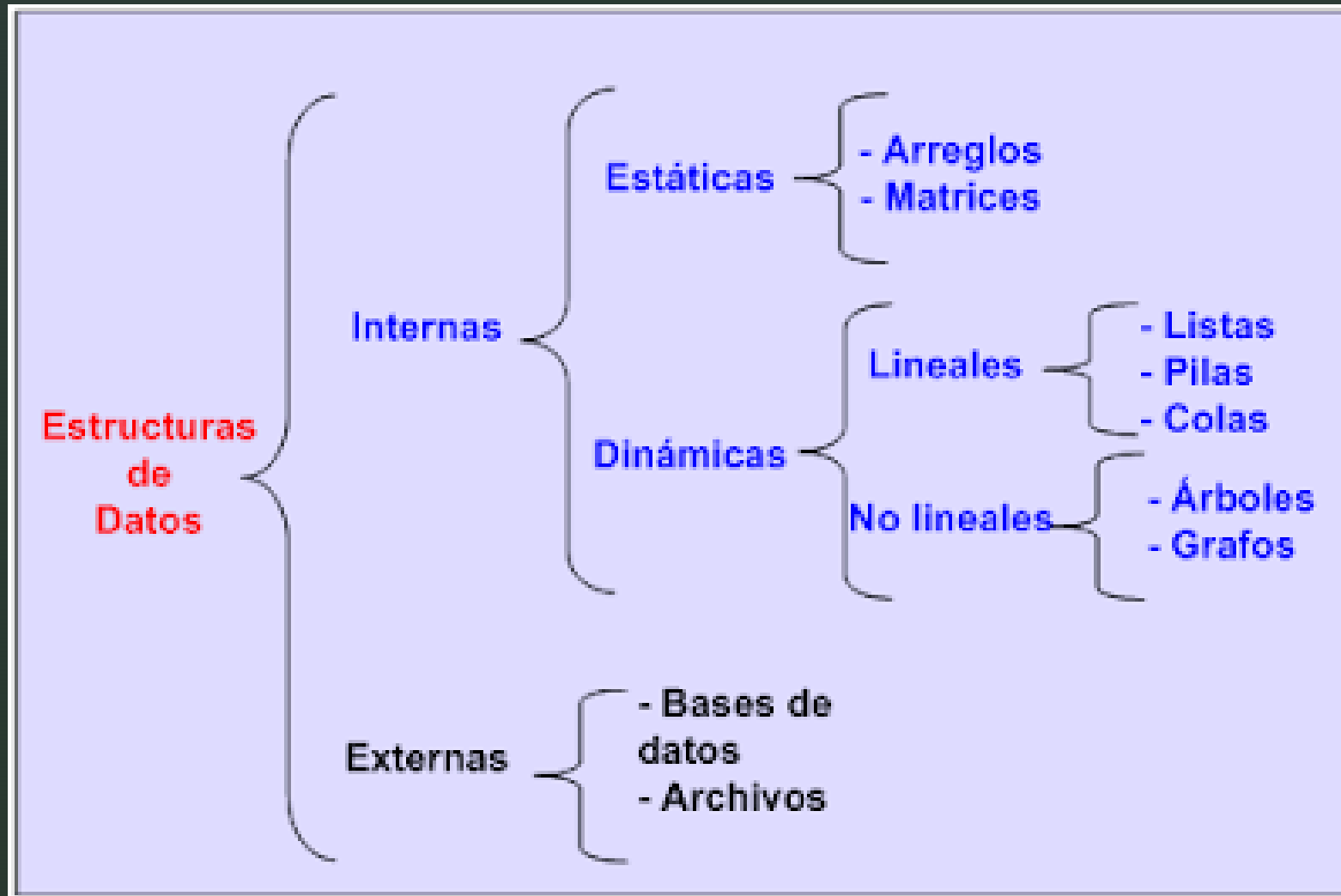
# ■ Prioridad de las operaciones en Python

**P** - paréntesis  
**E** - exponentes  
**M** - multiplicación  
**D** - división  
**A** - adición  
**S** - sustracción



# Tipos de Operadores:

Operador	Descripción	Ejemplo
=	asigna valor a una variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r1 = r</pre>
+=	suma el valor a la variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r += 10; r 15</pre>
-=	resta el valor a la variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r -= 10; r -5</pre>
*=	multiplica el valor a la variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r *= 10; r 50</pre>
/=	divide el valor a la variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r /= 10; r 0</pre>
**=	calcula el exponente del valor de la variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r **= 10; r 9765625</pre>
//=	calcula la división entera del valor de la variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r //= 10; r 0</pre>
%=	devuelve el resto de la división del valor de la variable	<pre>&gt;&gt;&gt; r = 5 &gt;&gt;&gt; r %= 10; r 5</pre>

## ■ Estructura de datos:



# Python 2 vs Python 3:

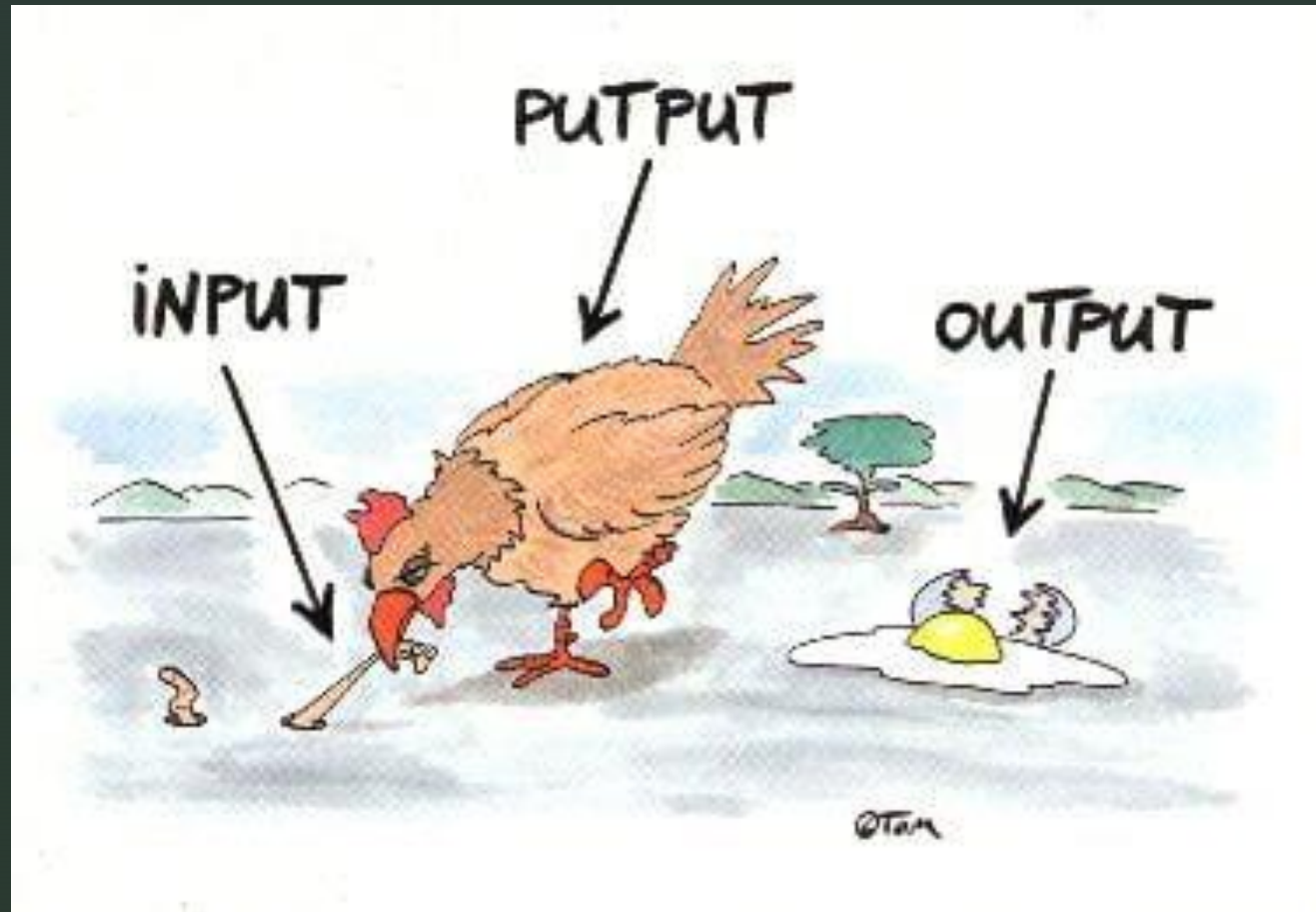
PYTHON 2.X 	PYTHON 3.X
← <b>LEGACY</b>	<b>FUTURE</b> →
It is still entrenched in the software at certain companies	It will take over Python 2 by the end of 2019
 <b>LIBRARY</b>	<b>LIBRARY</b> 
Many older libraries built for Python 2 are not forwards compatible	Many of today's developers are creating libraries strictly for use with Python 3
 <b>ASCII</b>	<b>UNICODE</b> 
Strings are stored as ASCII by default	Text Strings are Unicode by default
 <b>7/2=3</b>	<b>7/2=3.5</b> 
It rounds your calculation down to the nearest whole number	This expression will result in the expected result
 <b>print "WELCOME TO GEEKSFORGEEKS"</b>	<b>print("WELCOME TO GEEKSFORGEEKS")</b> 
It rounds your calculation down to the nearest whole number	This expression will result in the expected result

## ■ E/S Control de flujos y funciones:

1. Entradas y salidas (E/S)
2. Operadores de asignación
3. Estructura de control de flujo
4. Manipulación de cadenas
5. Funciones e introducción a la programación funcional



## Entradas y Salidas

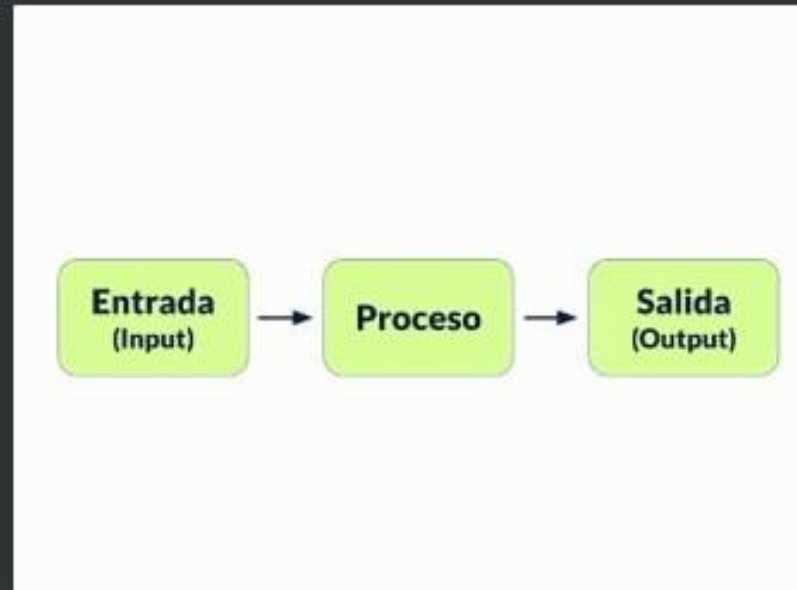




# Entradas y Salidas

Debe existir una entrada y salida de nuestra información en las computadoras.

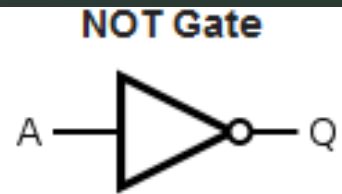
- **Input** : Es el proceso donde colocamos información de la que queremos tener resultado en nuestros cálculos.
- **Proceso**: Es la parte donde se realizan todos los cálculos con nuestra información de input.
- **Output**: Es el resultado que se nos da sobre el proceso de cálculo.



# Operadores de asignación

Operador	Ejemplo	Equivalente
=	$x=7$	$x=7$
+=	$x+=2$	$x=x+2 = 7$
-=	$x-=2$	$x=x-2 = 5$
*=	$x*=2$	$x=x*2 = 14$
/=	$x/=2$	$x=x/2 = 3.5$
%=	$x\%=2$	$x=x\%2 = 1$
//=	$x//=2$	$x=x//2 = 3$
**=	$x**=2$	$x=x**2 = 49$
&=	$x\&=2$	$x=x\&2 = 2$
=	$x =2$	$x=x 2 = 7$
=	$x^=2$	$x=x^2 = 5$
>>=	$x>>=2$	$x=x>>2 = 1$
&lt;&lt;=	$x\&lt;\&lt;=2$	$x=x\&lt;\&lt;2 = 28$

# Tabla de valores booleanos (Compuertas lógicas):



$$Q = \text{NOT}(A)$$

**Truth Table**

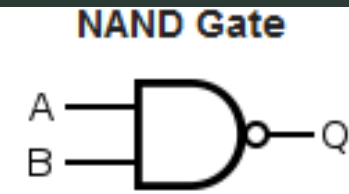
Input A	Output Q
0	1
1	0



$$Q = A \text{ AND } B$$

**Truth Table**

Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1



$$Q = A \text{ NAND } B$$

**Truth Table**

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

# Tabla de valores booleanos (Compuertas lógicas):

AND			OR		
VARIABLES		ESTADO	VARIABLES		ESTADO
SA	SB	Y	SA	SB	Y
False	False	False	False	False	False
False	True	False	False	True	True
True	False	False	True	False	True
True	True	True	True	True	True

XOR			NOT	
VARIABLE		ESTADO	VARIABLE	ESTADO
SA	SB	Y	SA	Y
False	False	False	True	False
False	True	True	False	True
True	False	True		
True	True	False		

## ■ Estructuras de control de flujo:

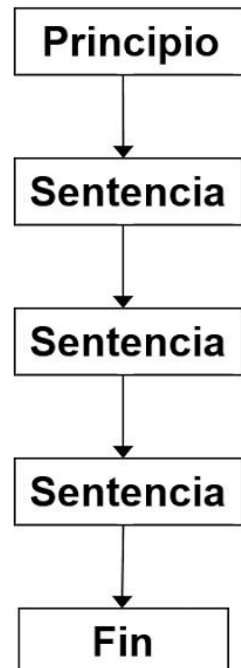
Las estructuras de control, son aquellas que permiten controlar el flujo de ejecución de un programa.

Existen tres principales tipos de estructuras:

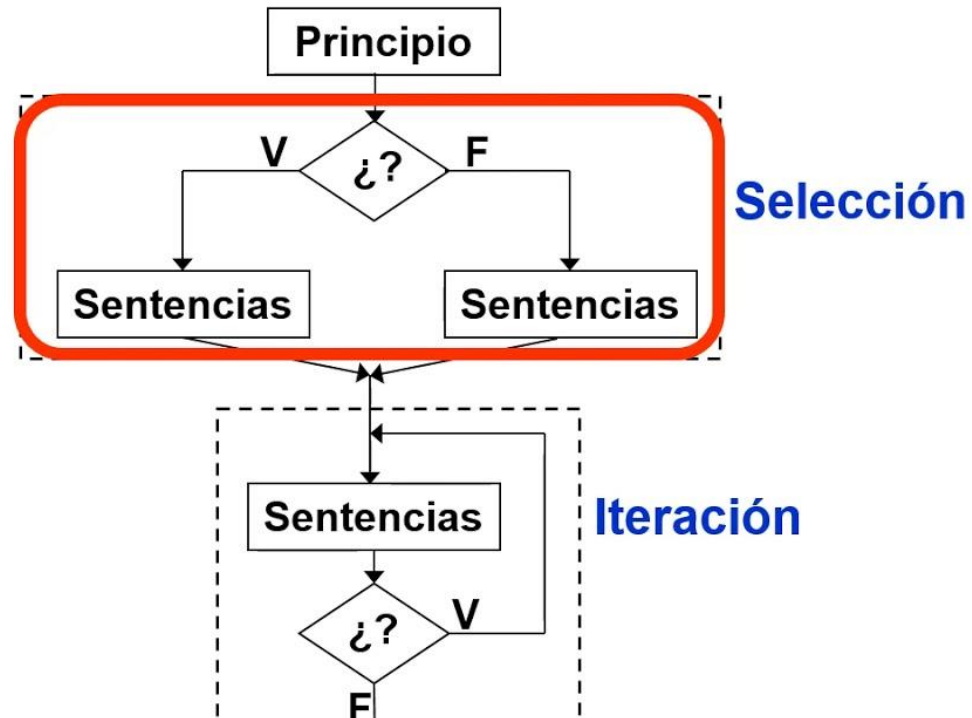
- Secuencial
- No secuenciales:
  - Instrucción condicional
  - Iteración (bucle de instrucciones)

## ■ Estructuras de control de flujo:

### Estructura secuencial

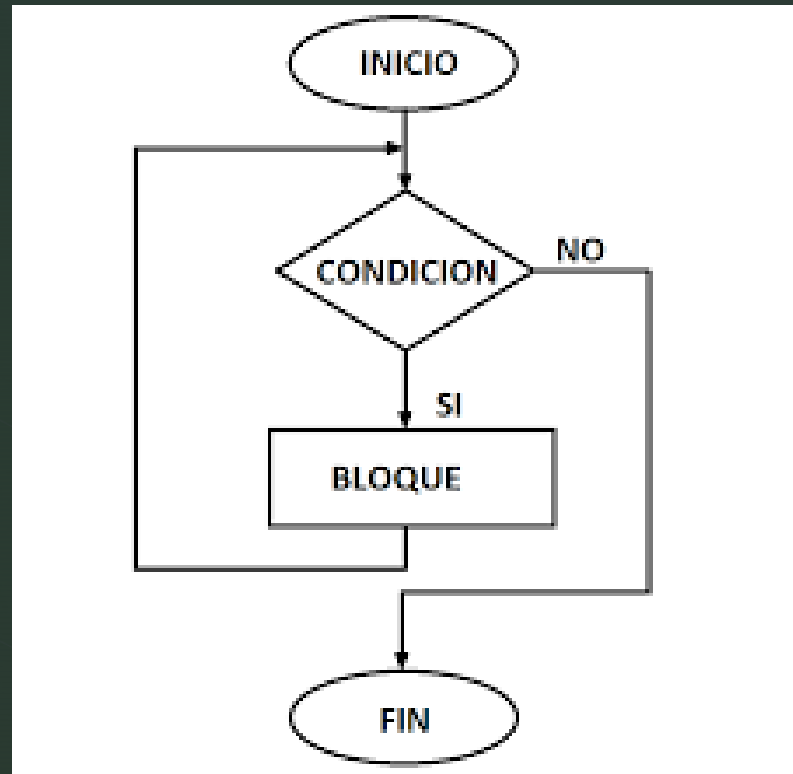
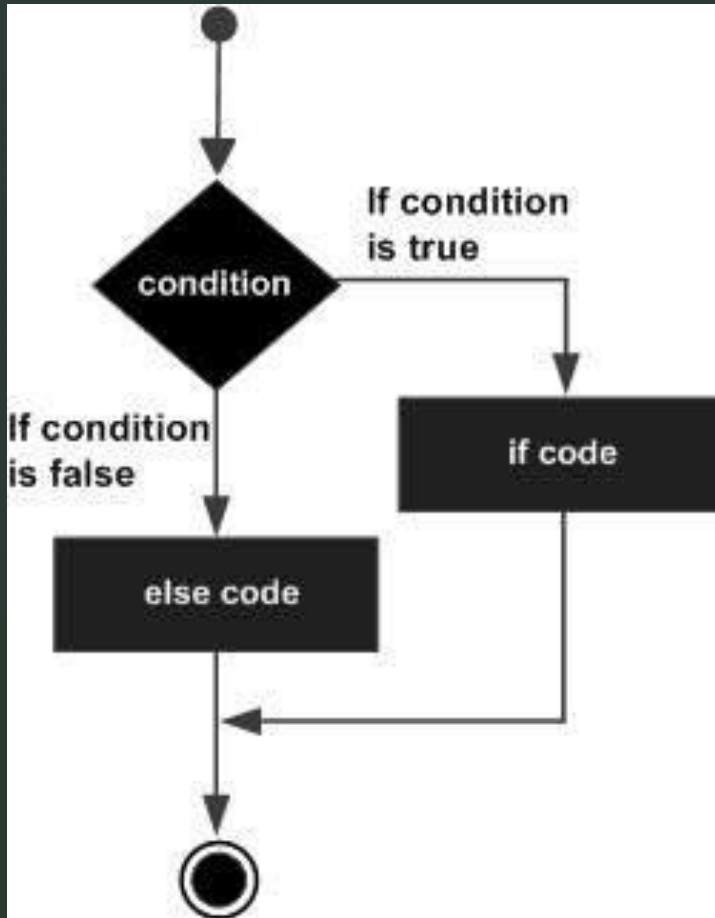


### Estructuras no secuenciales





## ■ Estructuras de control de flujo:



## Manipulación de cadenas:

```
>>> #CONCATENAR
>>> mensaje1 = 'Hola' + ' ' + 'Mundo'
... print(mensaje1)
... -> Hola Mundo
```

```
>>> #EXTENSIÓN
>>> mensaje4 = 'hola' + ' ' + 'mundo'
... print(len(mensaje4))
... -> 10
```

```
>>> #MULTIPLICAR
>>> mensaje2a = 'Hola ' * 3
... mensaje2b = 'Mundo'
... print(mensaje2a + mensaje2b)
... -> Hola Hola Hola Mundo
```

```
>>> #ENCONTRAR
>>> mensaje5 = "Hola Mundo"
... mensaje5a = mensaje5.find("Mundo")
... print(mensaje5a)
... -> 5
```

```
>>> #AÑADIR
>>> mensaje3 = 'Hola'
... mensaje3 += ' '
... mensaje3 += 'Mundo'
... print(mensaje3)
... -> Hola Mundo
```

```
>>> #MINÚSCULAS
>>> mensaje7 = "HOLA MUNDO"
... mensaje7a = mensaje7.lower()
... print(mensaje7a)
... -> hola mundo
```

## Manipulación de cadenas:

```
>>> #REEMPLAZAR
>>> mensaje8 = "HOLA MUNDO"
... mensaje8a = mensaje7.replace("L", "pizza")
... print(mensaje8a)
... -> HOpizzaA MUNDO
```

```
>>> #CORTAR
>>> mensaje9 = "Hola Mundo"
... mensaje9a = mensaje9[1:8]
... print(mensaje9a)
... -> ola Mun
```