

# Introducción a la programación con Python

Alexis Rodríguez

Marcel Morán C

# Instructores

Alexis Rodríguez



Marcel Morán C.



# Esquema

- Información del curso
- ¿Qué es programar?
- ¿Qué es Python?
- Sintaxis y semántica
- Tipos de datos
- Variables
- Operadores
- Control de flujo

# Información del curso

## Horarios

- Lunes a Viernes: 2 pm - 4 pm
- ~1h lectura y ~1h taller

## Cronograma

- Semana 1. Python y conceptos fundamentales
- Semana 2. Programación orientada a objetos con Python
- Semana 3. Test y Desarrollo de juego I
- Semana 4. Desarrollo de juego II y evaluación

## Evaluación

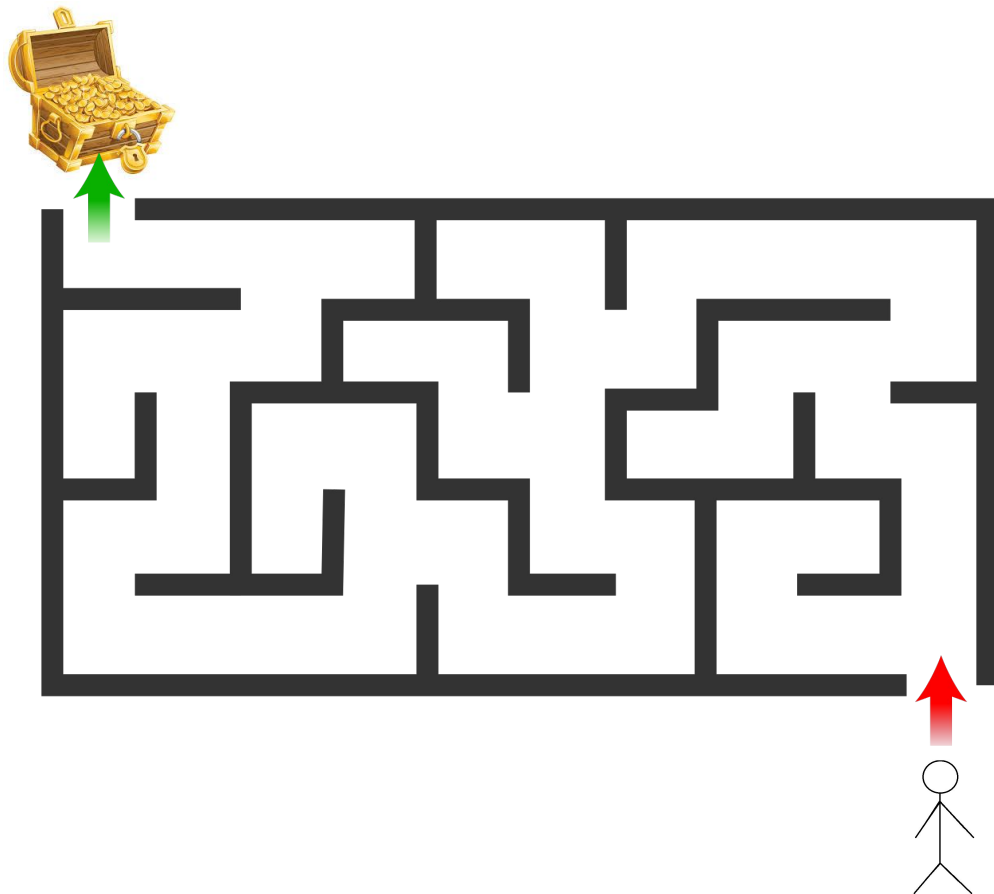
- Examen **12/04/2022**
- Proyecto en las dos últimas semanas
- Grupos de máximo dos personas
- Creación de un videojuego

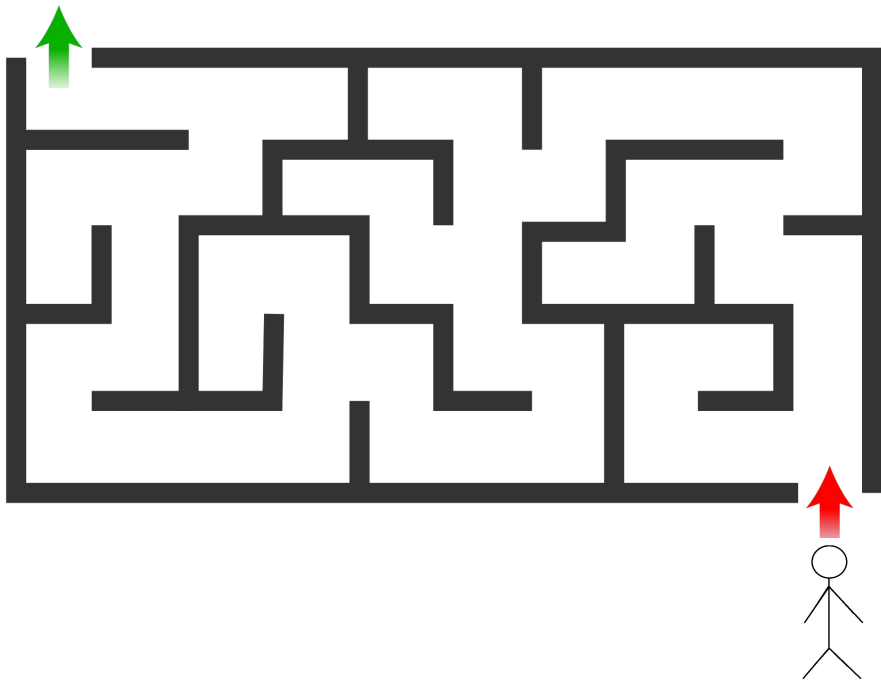
Servidor de discord: <https://discord.gg/zsqldW4TBc>



Github del curso: [shorturl.at/yGNS4](https://shorturl.at/yGNS4)

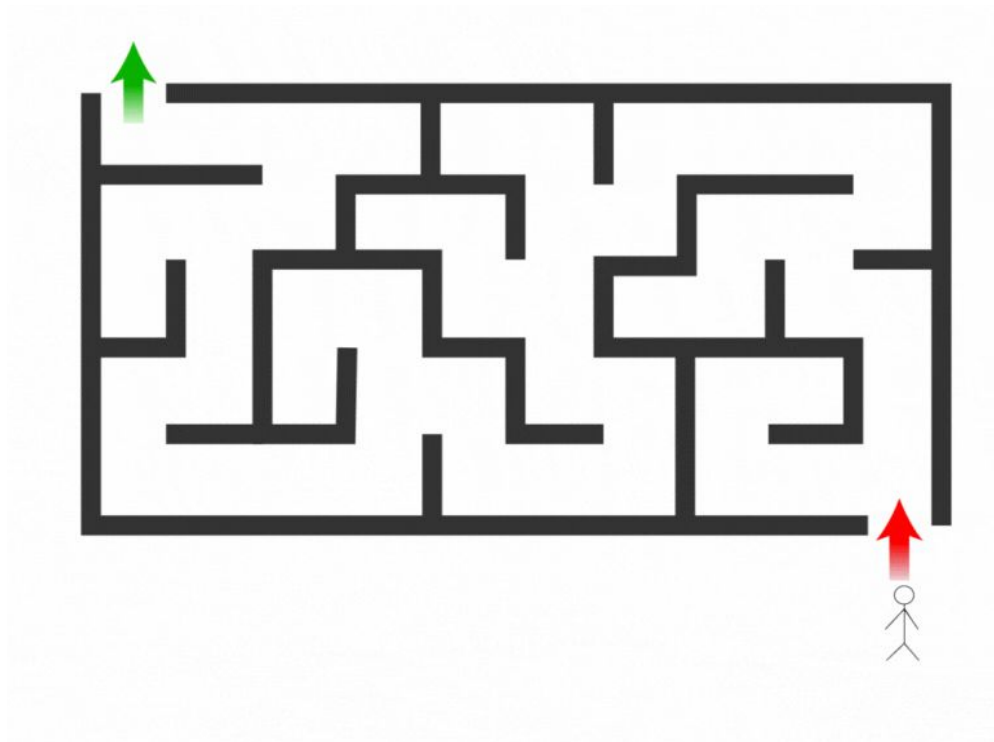






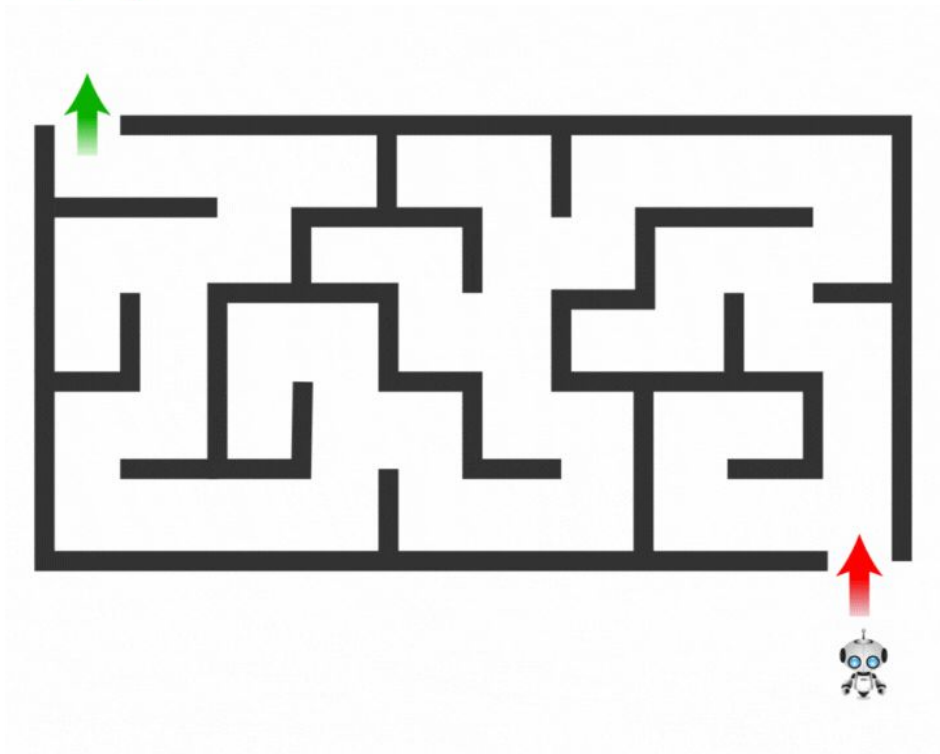
### Instrucciones

1. Entrar al laberinto
2. Posicionarse siempre a la pared derecha
3. Avanzar dando giros a la derecha cuando sea necesario
4. Repetir desde #2 hasta encontrar la meta



### Instrucciones

1. Entrar al laberinto
2. Posicionarse siempre a la pared derecha
3. Avanzar dando giros a la derecha cuando sea necesario
4. Repetir desde #2 hasta encontrar la meta



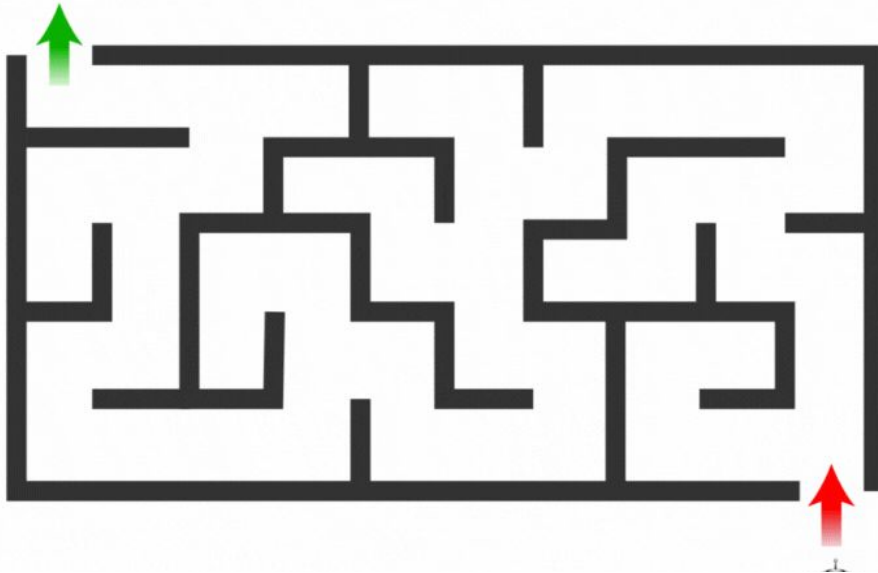
### Instrucciones

1. Entrar al laberinto
2. Posicionarse siempre a la pared derecha
3. Avanzar dando giros a la derecha cuando sea necesario
4. Repetir desde #2 hasta encontrar la meta



# ¿ Qué es programar?

- Proporcionar un conjunto o set de instrucciones a una computadora.
- Generalmente con el objetivo de resolver un problema.
- Las instrucciones son proporcionadas a través de lenguajes de programación.

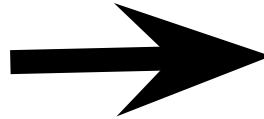
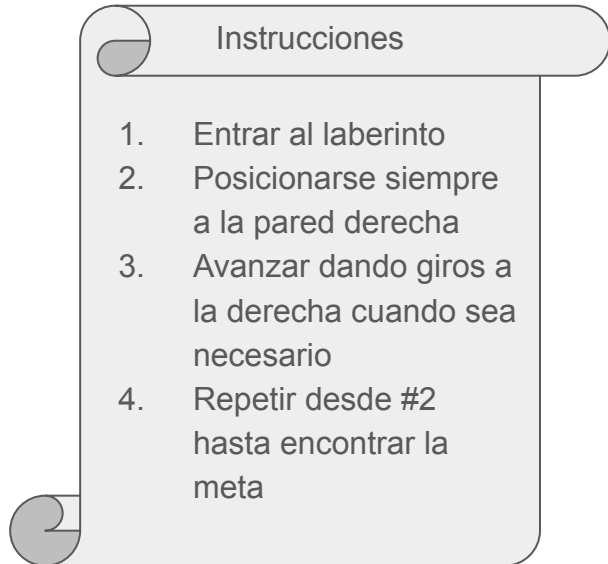


## Instrucciones

1. Entrar al laberinto
2. Posicionarse siempre a la pared derecha
3. Avanzar dando giros a la derecha cuando sea necesario
4. Repetir desde #2 hasta encontrar la meta

# ¿ Qué es programar?

- Proporcionar un conjunto o set de instrucciones a una computadora.
- Generalmente con el objetivo de resolver un problema.
- Las instrucciones son proporcionadas a través de lenguajes de programación.



# ¿ Qué es programar?

- Proporcionar un conjunto o set de instrucciones a una computadora.
- Generalmente con el objetivo de resolver un problema.
- Las instrucciones son proporcionadas a través de lenguajes de programación.



Solo sé binario

Hola ↔ 01101000 01101111 01101100 01100001

Sistema decimal = **0,1,2,3,4,5,6,7,8,9**

- Un dígito puede tener hasta 10 significados

cero = 0 0    nueve = 0 9    diez = 1 0

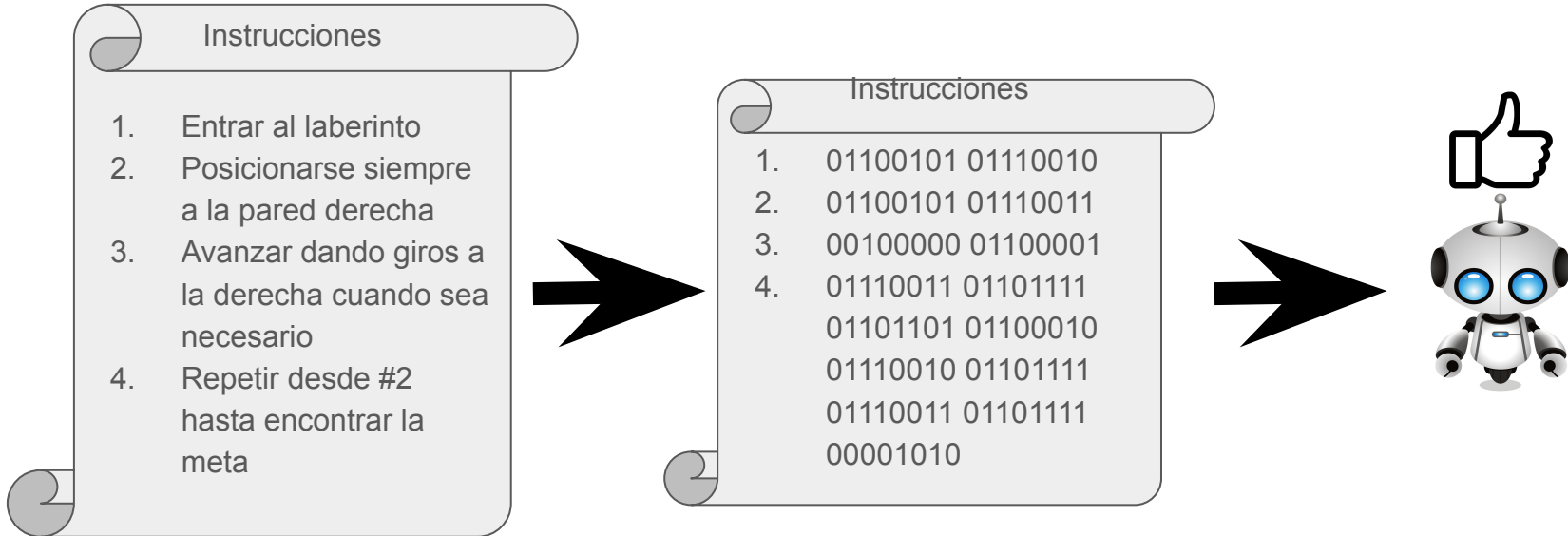
Sistema binario = **0, 1**

- Un dígito puede tener hasta 2 significados

cero = 0 0    uno = 0 1    dos = 1 0    tres = 1 1

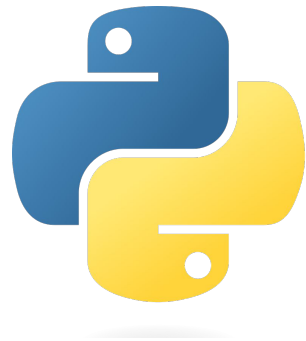
# ¿ Qué es programar?

- Proporcionar un conjunto o set de instrucciones a una computadora.
- Generalmente con el objetivo de resolver un problema.
- Las instrucciones son proporcionadas a través de lenguajes de programación.



# ¿ Qué es Python?

- Un lenguaje de programación de alto nivel
- Orientado a Objetos
- Se interpreta en lugar de compilarse



## PYPL PopularitY of Programming Language

Worldwide, Mar 2023 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.91 %	-0.6 %
2		Java	16.58 %	-1.6 %
3		JavaScript	9.67 %	+0.6 %
4		C/C++	6.93 %	-0.5 %

# ¿ Qué es Python?

- Un lenguaje de programación de alto nivel
- Orientado a Objetos
- **Se interpreta en lugar de compilarse**

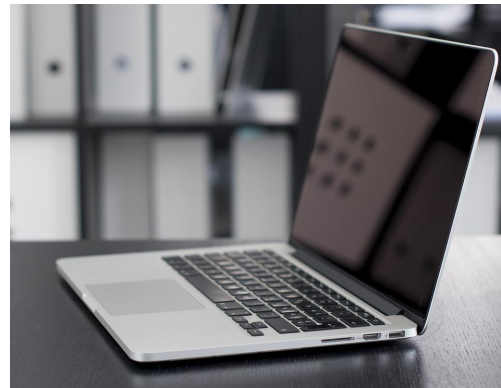
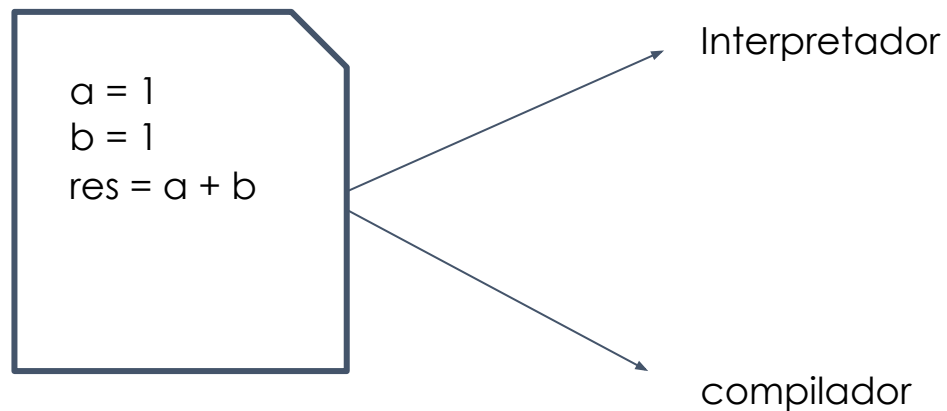


## PYPL PopularitY of Programming Language

Worldwide, Mar 2023 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.91 %	-0.6 %
2		Java	16.58 %	-1.6 %
3		JavaScript	9.67 %	+0.6 %
4		C/C++	6.93 %	-0.5 %

# ¿ Qué es Python?



# ¿ Qué es Python?

```
a = 1  
b = 1  
res = a + b
```

Interpretador

compilador

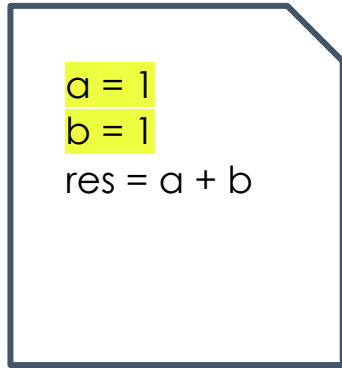
*instruccion*

0111110101





# ¿ Qué es Python?



Interpretador

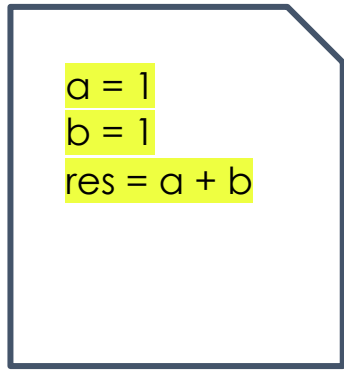
compilador

*instruccion*

*1101010101*



# ¿ Qué es Python?



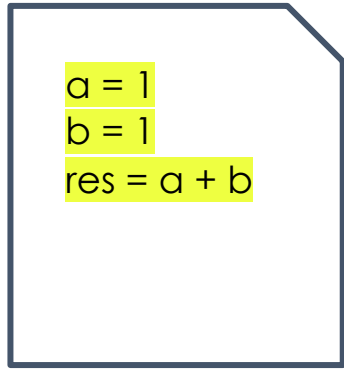
Interpretador

compilador

Instruccion  
0101010101



# ¿ Qué es Python?

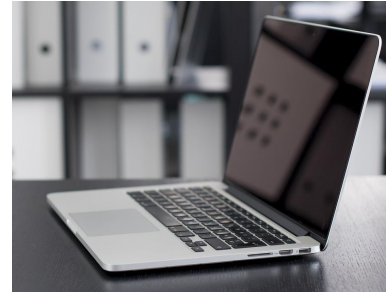


Interpretador

compilador

Conjunto de  
instrucciones

0101010101  
0111110101  
1101010101



# Sintaxis

- La sintaxis de Python define la combinación de palabras que son consideradas correctas o válidas.
- Python espera recibir las instrucciones de una manera predeterminada

El perro mordió al cartero

Al mordió cartero perro el

var = 10

Var tiene que ser 10

# Semántica

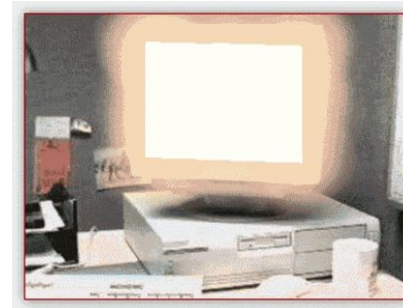
- En el lenguaje español, la semántica trata del significado de las expresiones.
- En programación, un programa semánticamente incorrecto realiza una tarea diferente a la esperada.

El perro mordió al cartero

El cartero mordió al perro

$x = 2 * (10 - 5)$

$x = 2 * 10 - 5$

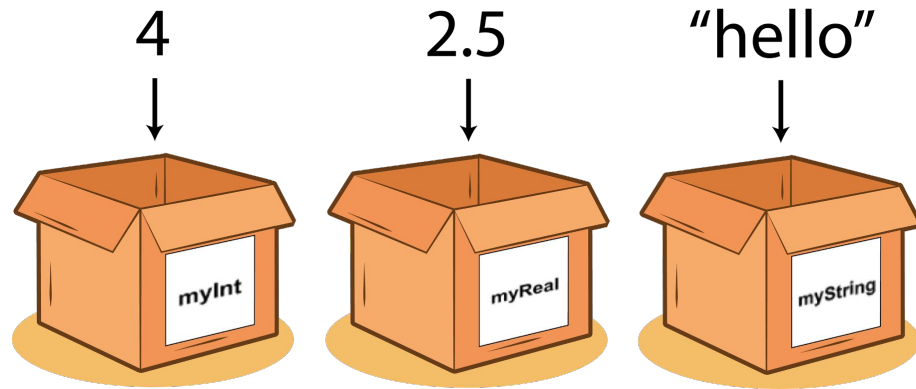


# Tipos de datos

- Strings, floats, booleans, e integers
- Datos numéricos pueden ser representados con integers: **-1, -2, 0, 1, 2**
- Representaciones decimales con floats: **-1.01, -2.001, 1.0001, 2.00001**
- Representación de valores condicionales con booleans: **True** o **False**
- Representación de palabras con strings: **“Hola”, “Mundo”, “Hola Mundo”**

# Variables

- Las variables nos permiten referirnos a un dato.
- Representación simbólica de un dato.



# Variables - ejemplo

```
un_numero = 15
pi = 3.1415
un_string = 'hola'
otro_string = "como estas"
```

```
print('tipo de variable "un_numero" es', type(un_numero))
print('tipo de variable "pi" es', type(pi))
print('tipo de variable "un_string" es', type(un_string))
print('tipo de variable "otro_string" es', type(otro_string))
```

```
tipo de variable "un_numero" es <class 'int'>
tipo de variable "pi" es <class 'float'>
tipo de variable "un_string" es <class 'str'>
tipo de variable "otro_string" es <class 'str'>
```



# Convirtiendo variables a un tipo diferente

- Las variables pueden convertirse de un tipo a otro
- Sintaxis: float(), str(), int()

- **3** -> **3.0**

- **3.0** -> **"3.0"**

- **"3.0"** -> **3**

```
un_entero = 15
un_float = float(un_entero)
print('un_entero:', un_entero)
print('un_float:', un_float)
```

```
un_entero: 15
un_float: 15.0
```

# Operadores

- Operadores aritméticos

OPERADOR	DESCRIPCIÓN	USO
+	Realiza Adición entre los operandos	$12 + 3 = 15$
-	Realiza Substracción entre los operandos	$12 - 3 = 9$
*	Realiza Multiplicación entre los operandos	$12 * 3 = 36$
/	Realiza División entre los operandos	$12 / 3 = 4$
%	Realiza un módulo entre los operandos	$16 \% 3 = 1$
**	Realiza la potencia de los operandos	$12 ** 3 = 1728$
//	Realiza la división con resultado de número entero	$18 // 5 = 3$

- Operadores lógicos

Operador	Descripción	Uso
<b>and</b>	Regresa True si ambos operandos son verdaderos	a <b>and</b> b
<b>or</b>	Regresa True si cualquiera de los operando son verdaderos	a <b>or</b> b
<b>not</b>	Regresa True si el operando es Falso	<b>not</b> a

A	B	A (and) B	A (or) B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

# Operadores

- Operadores aritméticos

OPERADOR	DESCRIPCIÓN	USO
+	Realiza Adición entre los operandos	$12 + 3 = 15$
-	Realiza Substracción entre los operandos	$12 - 3 = 9$
*	Realiza Multiplicación entre los operandos	$12 * 3 = 36$
/	Realiza División entre los operandos	$12 / 3 = 4$
%	Realiza un módulo entre los operandos	$16 \% 3 = 1$
**	Realiza la potencia de los operandos	$12 ** 3 = 1728$
//	Realiza la división con resultado de número entero	$18 // 5 = 3$

```
primer_numero = 10
segundo_numero = 3
```

```
suma = primer_numero + segundo_numero
print('suma:', suma)
```

```
>> suma: 13
```

```
multiplicacion = primer_numero * segundo_numero
print('multiplicacion:', multiplicacion)
```

```
>> multiplicacion: 30
```

```
modulo = primer_numero % segundo_numero
print('modulo:', modulo)
```

```
>> modulo: 1
```

# Operadores

- Operadores lógicos

Operador	Descripción	Uso
<b>and</b>	Regresa True si ambos operandos son verdaderos	a <b>and</b> b
<b>or</b>	Regresa True si cualquiera de los operandos son verdaderos	a <b>or</b> b
<b>not</b>	Regresa True si el operando es Falso	<b>not</b> a

A	B	A (and) B	A (or) B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

# Operadores

- Operadores relacionales

Símbolo	Descripción	Ejemplo	Booleano
<code>==</code>	Igual que	<code>1 == 1</code>	<code>False</code>
<code>!=</code>	Distinto que	<code>"Rojo" != "Verde"</code>	<code>True</code>
<code>&lt;</code>	Menor que	<code>8 &lt; 12</code>	<code>True</code>
<code>&gt;</code>	Mayor que	<code>12 &gt; 7</code>	<code>True</code>
<code>&lt;=</code>	Menor o igual que	<code>16 &lt;= 17</code>	<code>True</code>
<code>&gt;=</code>	Mayor o igual que	<code>67 &gt;= 72</code>	<code>False</code>

- Operadores de pertenencia

Identificar un elemento dentro de una secuencia.

*in / not in*

```
correo_recibido = "Pablo se encuentra a 10km de distancia  
y quiere conocerte."
```

```
es_spam = "quiere conocerte" in correo_recibido
```

# Operadores

- Operador a nivel de bits

&	Bitwise AND	$x \& y$
	Bitwise OR	$x   y$
~	Bitwise NOT	$\sim x$
^	Bitwise XOR	$x \wedge y$
>>	Bitwise right shift	$x >>$
<<	Bitwise left shift	$x <<$

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT Truth Table

A	B
0	1
1	0

```
a = 0b10110110 #182
b = 0b10001100 #140
```

```
a_and_b = a & b
print("a & b:", "{0:b}".format(a_and_b))
```

**Resultado: a & b: 10000100**

```
a_or_b = a | b
print("a | b:", "{0:b}".format(a_or_b))
```

**Resultado: a | b: 10111110**

```
a_desplazada_derecha = a >> 1
print("a >> 1:", "{0:b}".format(a_desplazada_derecha))
```

**Resultado: a >> 1: 01011011**

# Flujo del programa

- Hasta ahora no hemos podido especificar qué instrucciones se ejecuten
- Podemos controlar la ejecución de un programa con cláusulas condicionales
- Se requiere de una **condición** y una **cláusula** o bloque de código a ejecutar
- Condiciones siempre se evalúan a True o False
- Cláusulas pueden contener cláusulas dentro de sí mismo
- Las cláusulas necesitan estar **indentadas**

```
if condición:  
    _Cláusula  
elif condición:  
    _Cláusula  
else:  
    _Clausula
```

# Flujo del programa

```
edad_mama = 45
edad_papa = 50

if edad_mama == edad_papa:
    print('Mamá y papá tienen la misma edad')
elif edad_mama > edad_papa:
    print('Mamá es mayor que papá')
else:
    print('Papá es mayor que mamá')
```



# Conclusión

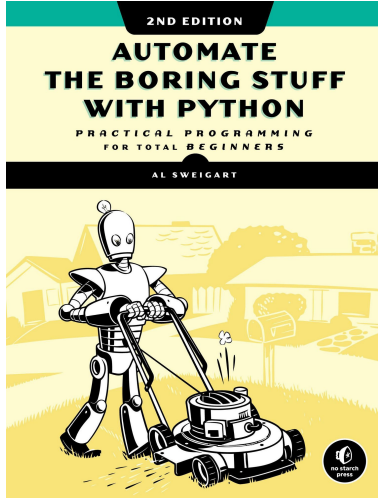
- Programar es escribir un conjunto de instrucciones con el fin de que se ejecuten en una computadora.
- La comunicación entre el programador y la computadora se hace a través de lenguajes de programación.
- Python es un lenguaje interpretado.
- Python contiene 4 principales tipos de datos: strings, floats, booleans e integers.
- Las variables son representaciones simbólicas usadas para hacer referencia a algún tipo de dato.
- Usando operadores podemos realizar operaciones aritméticas, lógicas e incluso comparar variables.
- El flujo del programa puede ser controlado usando cláusulas condicionales.

# Retroalimentación

- Para retroalimentación dirigirse al siguiente enlace <https://forms.gle/HeKEhanPpjeKsVve7> .
- Déjanos saber qué podemos hacer para mejorar el curso



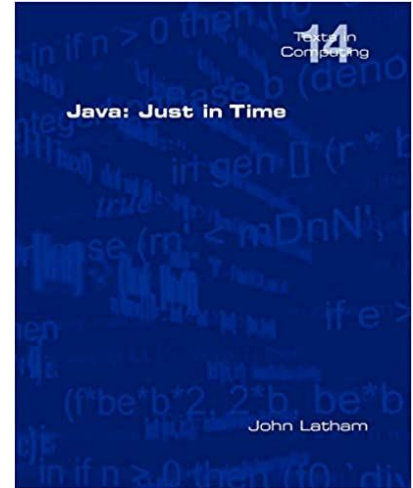
# Referencias



**Automate the Boring Stuff with Python**  
**Por Al Sweigart**



**Introduction To Computer Science And  
Programming In Python**



**Java: Just in Time**  
**Por John Latham**