# Python para Hackers

CloudLamb
E-Learning

CloudLamb E-Learning

# Python para Hackers

## Contenido del curso

**Capítulo 1.** Introducción

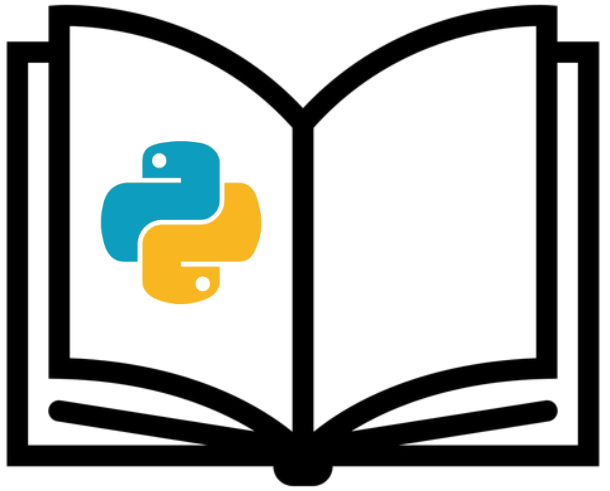
**Capítulo 2.** Primeros pasos

**Capítulo 3.** **Python** *Tema de hoy*

Tipos de datos y variables | Operadores | Cadenas | Condicionales | Bucles | Funciones | Clases y Objetos | Módulos | Archivos | Sockets

**Capítulo 4.** Hands-On

# Capítulo 3. Python

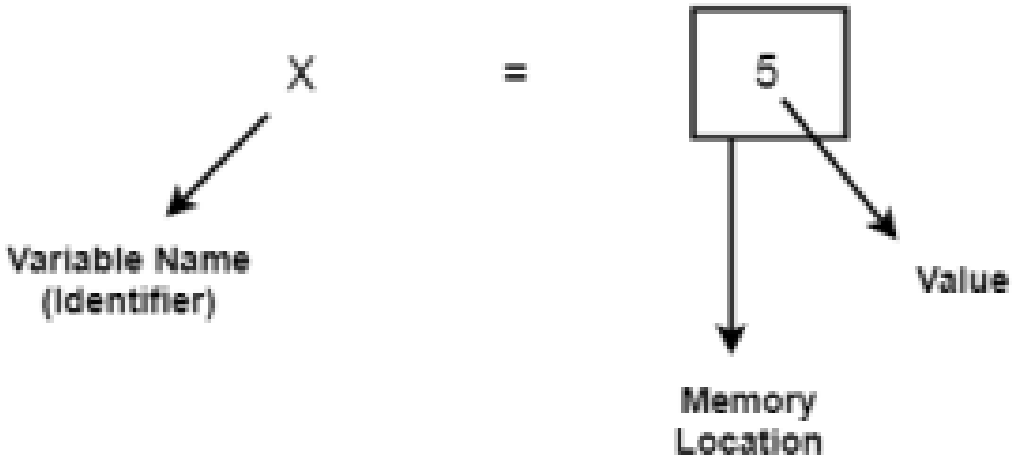


CloudLamb E-Learning

CloudLamb E-Learning

# Variables y tipos de datos

# Variables




Una variable es un **espacio reservado de memoria donde se guardan datos**.

CloudLamb E-Learning

# Variables

## Asignación y uso

En Python **no se tienen que declarar las variables** antes de usarlas, sino solamente asignarles un valor. Por ejemplo:

```
nombre_variable = valor
```

```
a = 3
cadena = “Hola mundo”
b = [1,2,3,4]
```

# Variables

## Convención

El **PEP8*** (Python Enhancement Proposal) es un documento que provee una guía y mejores prácticas para escribir código en Python. Su objetivo es mejorar la legibilidad y consistencia del código en Python.

"Para nombrar variables usar letras individuales, palabra o palabras en minúsculas. Separar las palabras con guiones bajos para mejorar la legibilidad. (x, var, mi_variable)"

* https://www.python.org/dev/peps/pep-0008/

# Tipos de datos

```
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'''
```

En Python cada dato o valor que se almacena en una variable tiene un **tipo de dato**.

CloudLamb E-Learning

# Tipos de datos

## Python

| Tipo de dato | Descripción | Ejemplo |
|---|---|---|
| *int* | Entero | 10 |

# Tipos de datos

## Python

| Tipo de dato | Descripción | Ejemplo |
|---|---|---|
| *int* | Entero | 10 |
| *float* | Decimal | 3.55 |

# Tipos de datos

## Python

| Tipo de dato | Descripción | Ejemplo |
|---|---|---|
| *int* | Entero | 10 |
| *float* | Decimal | 3.55 |
| *str* | Cadena | “Hola mundo” |

# Tipos de datos

## Python

| Tipo de dato | Descripción | Ejemplo |
|---|---|---|
| *int* | Entero | 10 |
| *float* | Decimal | 3.55 |
| *str* | Cadena | “Hola mundo” |
| *bool* | Booleano | True \| False |

# Tipos de datos

## Python

| Tipo de dato | Descripción | Ejemplo |
|---|---|---|
| *int* | Entero | 10 |
| *float* | Decimal | 3.55 |
| *str* | Cadena | “Hola mundo” |
| *bool* | Booleano | True \| False |
| *list* | Lista | [1.0, “uno”, False] |

# Tipos de datos

## Python

| Tipo de dato | Descripción | Ejemplo |
|---|---|---|
| *int* | Entero | 10 |
| *float* | Decimal | 3.55 |
| *str* | Cadena | “Hola mundo” |
| *bool* | Booleano | True \| False |
| *list* | Lista | [1.0, “uno”, False] |
| *tuple* | Dupla | (1.5, False, 30) |

# Tipos de datos

## Python

| Tipo de dato | Descripción | Ejemplo |
|---|---|---|
| *int* | Entero | 10 |
| *float* | Decimal | 3.55 |
| *str* | Cadena | “Hola mundo” |
| *bool* | Booleano | True \| False |
| *list* | Lista | [1.0, “uno”, False] |
| *tuple* | Dupla | (1.5, False, 30) |
| *dict* | Diccionario | {id1:’1’, id2:’Martin’} |

# Laboratorio

## Variables y tipos de datos

**Escribir** el siguiente código en PyCharm:

prueba.py

```
1   #entero
2   a = 1
3   #decimal
4   d = 2.5
5   #cadena
6   s = "Hola mundo"
7   #booleano
8   b = False
9   #lista
10  l = ["uno", 1.0, 1]
11  #dupla
12  du = (2.0,"dos")
13  #diccionario
14  dic = {"nombre":"Juan","edad":36}
```

CloudLamb E-Learning

# Laboratorio

## Variables y tipos de datos

**Imprimir** el valor de las variables con print().

```
print(a)  # Imprime un numero entero
print(d)  # Imprime un numero decimal
print(s)  # Imprime una cadena
print(b)  # Imprime un boolean
print(l)  # Imprime una lista completa
print(l[2])  # Imprime el elemento No. 3 de la lista
print(du)  # Imprime una dupla
print(du[1])  # Imprime el elemento No. 2 de la dupla
print(dict)  # Imprime el diccionario completo
print(dict.get("nombre"))  # Imprime el elemento nombre
```

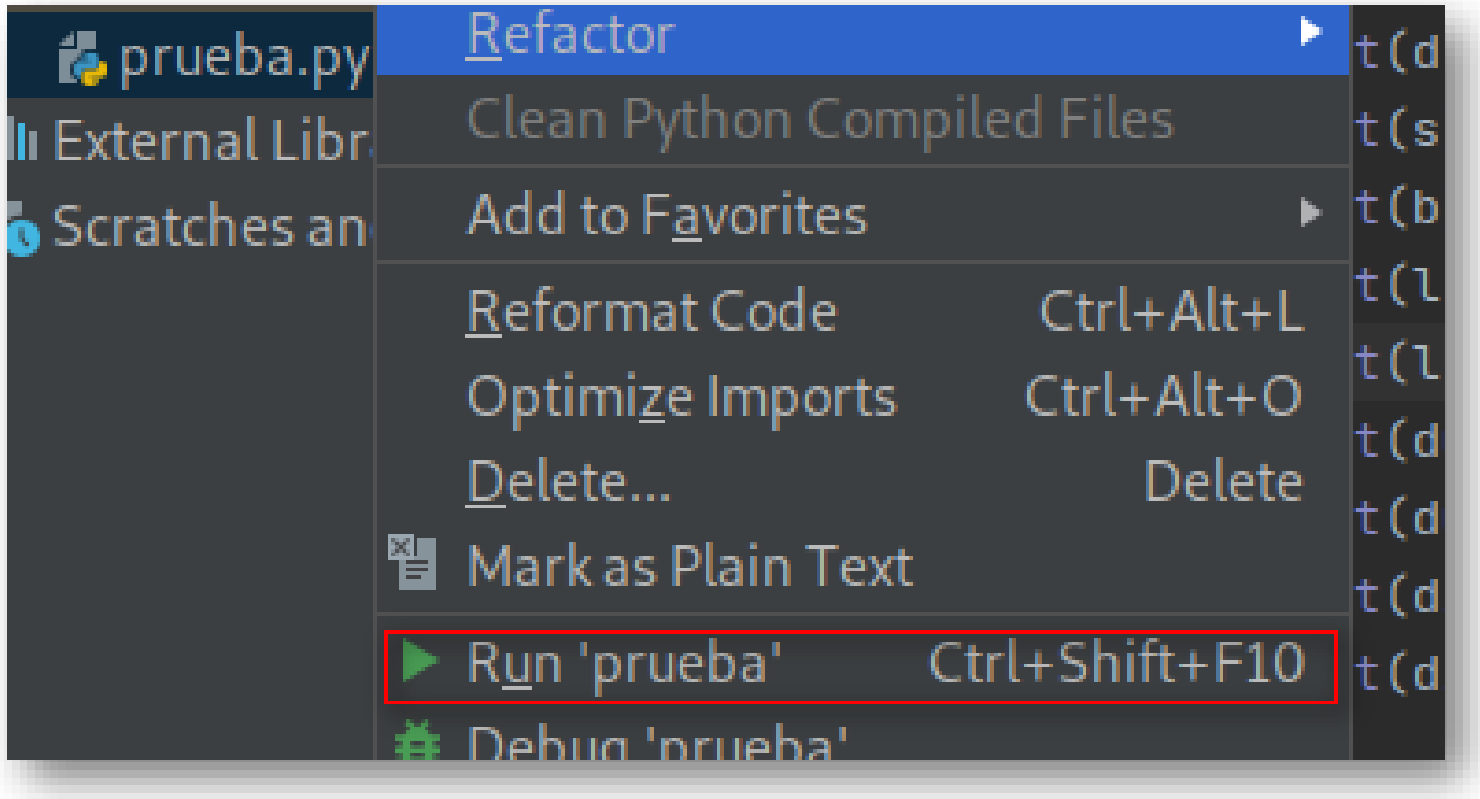
# Laboratorio

## Variables y tipos de datos

**Ejecutar** el script.

**Clic** derecho sobre el nombre del script.

**Seleccionar** “Run ‘nombre’”



# Laboratorio

## Variables y tipos de datos

**Resultado** de la ejecución del script:

```
/home/kali/PycharmProjects/PyHack/venv/bin/python /home/kali/PycharmProjects/PyHack/prueba.py
1
2.5
Hola mundo
False
['uno', 1.0, 1]
1
(2.0, 'dos')
dos
{'nombre': 'Juan', 'edad': 36}
Juan
```


CloudLamb
E-Learning

# Operadores


Los operadores se utilizan para realizar **operaciones sobre variables y valores**.

- Operadores aritméticos
- Operadores de asignación
- Operadores de comparación
- Operadores lógicos
- Operadores bit a bit
- Etc.

CloudLamb E-Learning

# Operadores aritméticos

## Python

Son utilizados con valores numéricos para realizar **operaciones matemáticas**.

| Operador | Descripción | |
|---|---|---|
| + | Suma | `x = 4 + 6` |
| - | Resta | `x = 6 - 3` |
| * | Multiplicación | `x = 6 * 5` |
| ** | Exponente | `x = 6 ** 2` |
| / | División | `x = 6 / 3` |
| % | Módulo | `x = 6 % 4` |

CloudLamb E-Learning

# Laboratorio

## Operadores aritméticos

**Utilizar** los operadores aritméticos en PyCharm:

prueba.py

```
28  # Operadores aritmeticos
29  print("\nOPERADORES ARITMETICOS")
30  a = 3
31  b = 6
32
33  c = a + b
34  print(c)
35  c = b - a
36  print(c)
37  c = a * b
38  print(c)
39  c = b ** a
40  print(c)
41  c = b / a
42  print(c)
43  c = b % a
```

# Laboratorio

## Operadores aritméticos

**Ejecutar** el script y revisar el resultado.

```
OPERADORES ARITMETICOS
9
3
18
216
2.0
0
```

CloudLamb E-Learning

# Operadores de asignación

## Python

Son utilizados para **asignar valores a las variables**.

| Operador | Ejemplo | Igual que |
|---|---|---|
| = | x = 5 | x = 5 |
| += | x += 5 | x = x + 5 |
| -= | x -= 5 | x = x - 5 |
| *= | x *= 5 | x = x * 5 |
| **= | x **= 5 | x = x ** 5 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 2 | x = x % 2 |

CloudLamb E-Learning

# Laboratorio

## Operadores de asignación

**Utilizar** los operadores de asignación en PyCharm:

prueba.py

```
46  # Operadores de asignacion
47  print("\nOPERADORES DE ASIGNACION")
48  x = 5
49  x += 5
50  print(x)
51  x -= 2
52  print(x)
53  x *= 2
54  print(x)
55  x **= 2
56  print(x)
57  x /= 4
58  print(x)
59  x %= 3
60  print(x)
```

# Laboratorio

## Operadores de asignación

**Ejecutar** el script y revisar el resultado.

```
OPERADORES DE ASIGNACION
10
8
16
256
64.0
1.0
```

# Estructuras de control

## Python

Es un **bloque de código** que permite agrupar instrucciones de manera controlada.

En Python existen dos tipos de estructuras:

- Estructuras de control *condicionales*
- Estructuras de control *iterativas*

# Estructuras de control

## Identación

En Python la **identación** es obligatoria, ya que la estructura de un programa depende de ella. Esta indica que las instrucciones identadas forman parte de una misma estructura de control.

```
Estructura de control
      expresiones
```

PEP8: Usar 4 espacios por cada nivel. Los espacios son recomendados en vez de utilizar tabs.

# Estructuras de control

## Condicionales

Estructura que permite **controlar condiciones** y que el programa se comporte de cierta manera dependiendo del resultado de la condición.

```
if (si), elif (sino, si), else (sino)
```

Las estructuras de control condicionales utilizan:

- Operadores *relacionales*
- Operadores *lógicos*

# Operadores relacionales

## Python

Son utilizados para **comparar entre dos valores.**

| Operador | Descripción |
|---|---|
| == | Igual |
| != | No igual |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |

# Operadores lógicos

## Python

Son utilizados para **evaluar más de una condición simultáneamente.**

| Operador | Ejemplo | Explicación | Resultado |
|---|---|---|---|
| and | 5 == 6 and 6 > 12 | False and False | False |
| and | 9 < 10 and 5 == 5 | True and True | True |
| and | 9 < 7 and 4 > 3 | False and True | False |

**AND**

| | F | V |
|---|---|---|
| F | F | F |
| V | F | V |

# Operadores lógicos

## Python

Son utilizados para **evaluar más de una condición simultáneamente.**

| Operador | Ejemplo | Explicación | Resultado |
|---|---|---|---|
| and | 5 == 6 and 6 > 12 | False and False | False |
| and | 9 < 10 and 5 == 5 | True and True | True |
| and | 9 < 7 and 4 > 3 | False and True | False |
| or | 5 < 10 or 4 == 3 | True or False | True |
| or | 6 > 3 or 4 < 6 | True or True | True |

**OR**

| | F | V |
|---|---|---|
| F | F | V |
| V | V | V |

CloudLamb E-Learning

# Operadores lógicos

## Python

Son utilizados para **evaluar más de una condición simultáneamente.**

| Operador | Ejemplo | Explicación | Resultado |
|---|---|---|---|
| and | 5 == 6 and 6 > 12 | False and False | False |
| and | 9 < 10 and 5 == 5 | True and True | True |
| and | 9 < 7 and 4 > 3 | False and True | False |
| or | 5 < 10 or 4 == 3 | True or False | True |
| or | 6 > 3 or 4 < 6 | True or True | True |
| xor | 4 == 4 xor 7 > 3 | True xor True | False |
| xor | 3 == 3 or 3 > 6 | True xor False | True |

**XOR**

| | F | V |
|---|---|---|
| F | F | V |
| V | V | F |

# Estructuras de control

## Iterativas

Permiten **ejecutar un mismo código de manera repetida**, mientras la condición se cumpla. También son conocidas como estructuras *cíclicas* o *bucles*.

**while** (mientras que)
**for** (por cada)

También utilizan los operadores *relacionales* y *lógicos*.

# Laboratorio

## Estructuras de control

### IF *(si)*

```
if condición:
      código
```

**Código:**

```
# Estructuras de control
print("\nESTRUCTURAS DE CONTROL")
x = 3 + 4

if x == 7:
    print("X es igual a 7")

if x < 3:
    print("X es menor a 3")

if 10 > x > 5:
    print("X cumple la condicion")
```

**Ejecución:**

```
X es igual a 7
X cumple la condicion
```

# Laboratorio

## Estructuras de control

**ELSE** *(sino)*

```
if condición:
      código
else:
      código
```

**Nota:** ELSE siempre va al final de un IF o un ELIF.

**Código:**

```
print("\nELSE")
x = 65
if x % 2 == 0:
    print("No hay residuo")
else:
    print("La division tiene residuo")
```

**Ejecución:**

```
ELSE
La division tiene residuo
```

# Laboratorio

## Estructuras de control

**ELIF** *(sino, si)*

```
if condicion:
      código
elif condición:
      código
```

**Nota:** ELIF siempre va al final de un IF.

**Código:**

```
print("\nELIF")
x = 5
if x == 4:
    print("X es 4")
elif x != 4:
    print("X no es 4")
```

**Ejecución:**

```
ELIF
X no es 4
```

# Laboratorio

## Estructuras de control

### Operadores lógicos

```
if condicion1 and condicion2:
    código
```

**Código:**

```
print("\nOPERADORES LOGICOS")
x = 3
y = False
if x == 3 and y != True:
    print("Condicion cumplida")
```

**Ejecución:**

```
OPERADORES LOGICOS
Condicion cumplida
```

# Laboratorio

## Estructuras de control

### WHILE *(mientras)*

```
while condicion:
      código
```

**Código:**

```
print("\nWHILE")
x = 1
while x < 10:
    print("X:" + str(x))
    x += 1
```

**Nota:** *str* es una función que convierte el parámetro de entrada en un *string*.

**Ejecución:**

```
WHILE
X:1
X:2
X:3
X:4
X:5
X:6
X:7
X:8
X:9
```

# Próxima clase...

- **Capítulo 3:** Python (Parte II)


CloudLamb
E-Learning

CloudLamb E-Learning

CloudLamb

# ¡Muchas gracias por su atención!