

UNIDAD 2:

VARIABLES Y TIPOS DE DATOS

CONTENIDOS:

2.1 Tipos de datos primitivos.

2.2 Definición y asignación de variables.

2.3 Operadores y expresiones matemáticas, lógicas y relacionales.

2.4 Conversiones entre tipos de datos.

OBJETIVOS ESPECÍFICOS

- Seleccionar los tipos de datos y los operadores lógicos y relacionales apropiados para escribir expresiones válidas en un lenguaje de programación.
- Utilizar sentencias de entrada y salida de datos con formato para la creación de programas sencillos.
- Aplicar la precedencia de los operadores, el operador de asignación y su uso, la lógica usada en las operaciones booleanas y los tipos de datos para escribir expresiones válidas en un lenguaje de programación.

TIPOS DE DATOS PRIMITIVOS

- Numéricos:

Tipo	Nombre	Descripción	Ejemplo
Enteros	int	Números sin parte fraccionaria	52 0 -318
Reales o de punto flotante	float	Números con parte fraccionaria o expresados en notación de potencias de 10	6.37 -0.089 4.1e-3
Complejos	complex	Números con un componente real y uno imaginario	(9-3j) (2.5+6.4j)

TIPOS DE DATOS PRIMITIVOS

- Lógicos:

Tipo	Nombre	Descripción	Ejemplo
Booleano	bool	Representación de los valores lógicos Verdadero o Falso.	TRUE FALSE

- Cadenas de Caracteres:

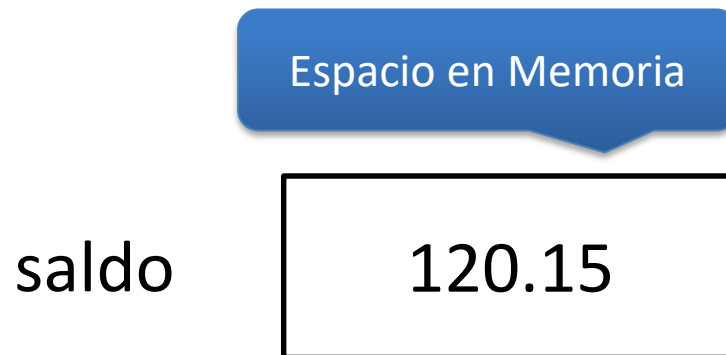
Tipo	Nombre	Descripción	Ejemplo
Cadenas	str	Expresiones (texto) formadas por caracteres. Se pueden representar con comillas simples o dobles.	'Hola' "Mundo"

REPASO TIPOS DE DATOS

- Identificar el tipo de dato que utilizaría en el lenguaje Python para los siguientes casos, e indicar un ejemplo:
 1. El kilometraje de un carro.
 2. El estado civil de una persona
 3. ¿Tiene multa un conductor?
 4. La cantidad de hijos de una pareja
 5. El color de un pantalón
 6. ¿Tiene beca un estudiante?
 7. La matrícula de un estudiante.
 8. El costo de un producto.
 9. ¿Aprobarás Fundamentos de Programación?

DEFINICIÓN DE VARIABLES

- Una **variable** es un dato cuyo valor puede cambiar durante un cálculo o en la resolución de un problema.
- A través de las variables se puede almacenar, organizar y manipular la información en la memoria (RAM).



DEFINICIÓN DE VARIABLES

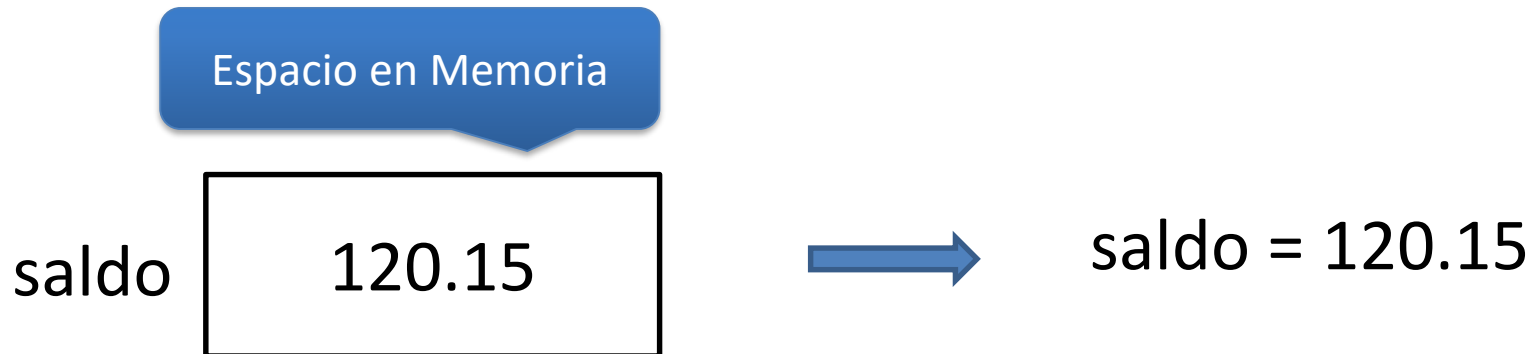
- El nombre de una variable en Python debe seguir ciertas reglas:
 - Sólo puede contener números, letras o el caracter _
 - No puede iniciar con un número.
 - No debe coincidir con una palabra reservada del lenguaje.
- Buenas prácticas de programación:
 - Elegir un nombre significativo que tenga relación con el dato que representará.
 - No utilizar nombres demasiado largos.

DEFINICIÓN DE VARIABLES

No correcto	Correcto
variable	edad
A B C	deposito retiro saldo
1numero 2numero	numero1 numero2
caso-1 caso-2	caso_1 caso_2
input	entrada

ASIGNACIÓN DE VARIABLES

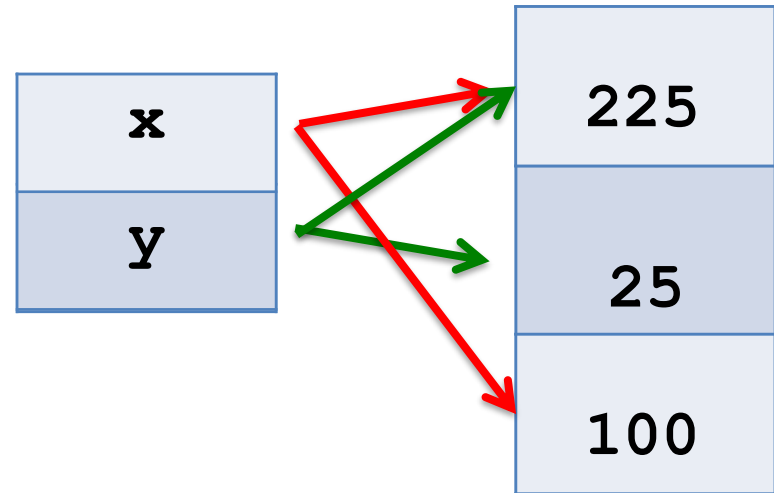
- La operación de asignación se utiliza para definir variables y dar un valor a su contenido.
- Se efectúa de derecha a izquierda. Si hay operaciones, éstas se calculan, luego se asigna el resultado a la variable.
- Cualquier valor que haya tenido la variable antes de la asignación, se pierde y es sobrescrito con el nuevo valor.



ASIGNACIÓN DE VARIABLES

```
>>> x = 15
>>> y = 25

>>> x = 100
>>> y = 225
```



TIP:

Se puede cambiar el valor de una variable en una instrucción posterior

ASIGNACIÓN DE VARIABLES

- **Asignación en la misma línea:**

```
x = 5; y = 9; z = 12
```

- **Asignación múltiple:**

```
day, month, year = "miércoles", "mayo", 2016
```

- **Asignación del mismo valor:**

```
largo = ancho = 4
```

- **Asignación de intercambio:**

```
base = 15; altura = 30
```

```
base, altura = altura, base
```

OPERADORES ARITMÉTICOS

- Permiten realizar operaciones aritméticas utilizando directamente símbolos del teclado.

Símbolo	Operación	Ejemplo	Resultado
+	Suma	$2 + 4$	6
-	Resta	$8 - 5$	3
*	Multiplicación	$6 * 2$	12
/	División	$9 / 2$	4.5
//	División (Entera)	$9 // 2$	4
%	Módulo	$9 \% 2$	1
**	Potenciación	$2 ** 3$	8

OPERADORES RELACIONALES

- Se utilizan para evaluar condicionales; al operarlos se obtiene como resultado valores booleanos.

Símbolo	Operación	Ejemplo	Resultado
==	Igual que	5 == 5	True
!=	Distinto que	8 != 5	True
>	Mayor que	6 > 9	False
<	Menor que	9 < 2	False
>=	Mayor o igual que	7 >= 3	True
<=	Menor o igual que	4 <= 2	False

OPERADORES LÓGICOS

- Permiten construir expresiones lógicas, obteniendo como resultado valores booleanos.

Símbolo	Operación	Ejemplo	Resultado
and	Conjunción	$2 > 1$ and $4 < 8$	True
or	Disyunción	$9 \neq 6$ or $7 \leq 3$	True
not	Negación	not True	False

OPERADORES DE INCREMENTO/DECREMENTO

Símbolo	Ejemplo	Equivalente a
<code>+=</code>	<code>a+=5</code>	<code>a=a+5</code>
<code>-=</code>	<code>a-=5</code>	<code>a=a-5</code>
<code>*=</code>	<code>a*=5</code>	<code>a=a*5</code>
<code>/=</code>	<code>a/=5</code>	<code>a=a/5</code>
<code>%=</code>	<code>a%=5</code>	<code>a=a%5</code>

VERIFICAR TIPOS DE DATOS

Se utiliza type() para conocer el tipo de dato de una variable

```
>>> 30 == 40
```

```
False
```

```
>>> 25 > 12
```

```
True
```

```
>>> type(58)
```

```
<class 'int'>
```

```
>>> type(-4)
```

```
<class 'int'>
```

```
>>> type(9.8)
```

```
<class 'float'>
```

```
>>> type(-0.69)
```

```
<class 'float'>
```

```
>>> type("Fundamentos de Programación")
```

```
<class 'str'>
```

```
>>> type('2016')
```

```
<class 'str'>
```


OPERACIONES ARITMÉTICAS

- $a=2;$ $(a+2)^3$

- $a=4; b=3;$ $\frac{a+5}{b-1}$

```
>>> a=2; (a+2)**3
```

```
>>> a=4; b=3; (a+5)/(b-1)
```

PRECEDENCIA DE OPERADORES

```
x = 1 + 2 * 3 - 4 / 5 ** 6
```

```
x = 6.999744
```

Paréntesis

Potencia

Multiplicación y división

Suma y resta

Operadores de igual precedencia se evalúan de derecha a izquierda

Si una expresión contiene operadores de diferente tipo, se evalúan primero las operaciones aritméticas, luego las relacionales, y finalmente las lógicas.



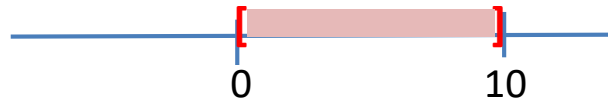
EXPRESIONES

- Una expresión es una secuencia de valores unidos por operadores, que al ser evaluada se simplifica en otro valor.
- Se puede utilizar paréntesis para indicar la precedencia de los operadores.

((3+4*x) > 10*(y-5)) and ((a+b)/c != 9*(4/a + (9+b)/c))

EXPRESIONES

- Número x en el intervalo entre $[0$ y $10]$



$$x \geq 0 \text{ and } x \leq 10$$

- Número x fuera del intervalo $[0, 10]$



$$\text{not } (x \geq 0 \text{ and } x \leq 10)$$

o también $x < 0 \text{ or } x > 10$

EJERCICIO: EXPRESIONES

- Crear un programa con 3 variables con números enteros y mostrar por pantalla si los tres números cumplen las siguientes restricciones (resultado TRUE o FALSE):

El número 1 elevado al cuadrado debe ser diferente de la resta del número 2 con el número 3.

El número 2 debe ser un número entre 20 y 40.

El número 3 debe ser un número entre -15 y -30.

El producto del número 2 y el número 3 debe ser divisible para 2.

CONVERSIONES ENTRE TIPOS DE DATOS

- Se puede realizar conversiones entre tipos de datos siempre que el contenido sea compatible.



```
>>> int(3.14)
>>> int(-3.999)
>>> int("2345")
>>> int(17)
>>> int("23 bottles")
```

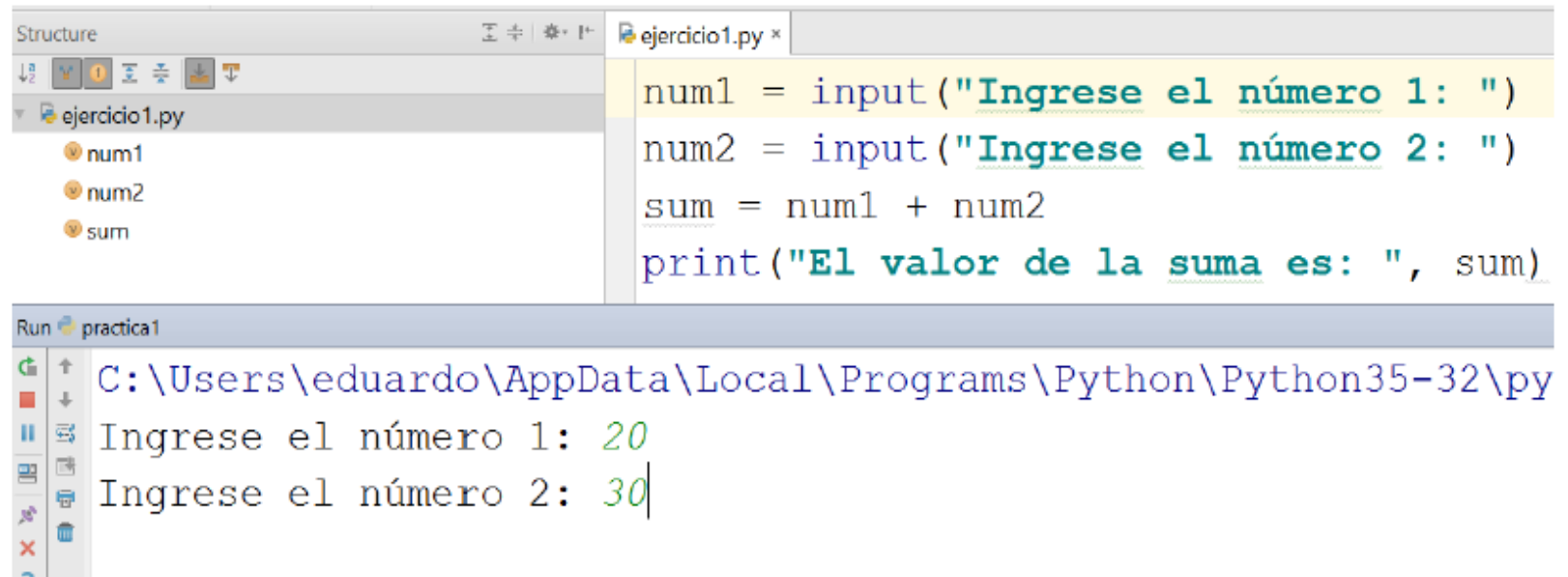
```
>>> float(17)
>>> float("123.45")
>>> str(17)
>>> str(123.45)
```

CONVERSIONES ENTRE TIPOS DE DATOS

Dato	Conversión	Resultado
saldo = 120	float (saldo)	120.0
saldo = 120	str (saldo)	'120'
saldo = '120'	int (saldo)	120
saldo = '120.0'	float (saldo)	120.0
saldo = '120.0'	int (saldo)	Error
saldo = 'x120'	int (saldo)	Error

EJERCICIO: CONVERSIONES

3.-Cuál es la salida de ejecutar el siguiente script, utilizando los datos de entradas mostrados en la siguiente imagen.



The screenshot displays a Python IDE interface. The top pane shows the file 'ejercicio1.py' with the following code:

```
num1 = input("Ingrese el número 1: ")
num2 = input("Ingrese el número 2: ")
sum = num1 + num2
print("El valor de la suma es: ", sum)
```

The left sidebar shows the file structure with 'ejercicio1.py' expanded, listing variables 'num1', 'num2', and 'sum'. The bottom pane, titled 'Run practica1', shows the execution output:

```
C:\Users\eduardo\AppData\Local\Programs\Python\Python35-32\py
Ingrese el número 1: 20
Ingrese el número 2: 30
```