



Unidad 3

Implementación de algoritmos en Python.

Conversión de tipos

Otro ejemplo de conversión de tipos es convertir la entrada del usuario (que es una **cadena**) a números (**enteros** o **flotantes**)

```
>>> float(input("Entre un numero: ")) + float(input("Entre otro numero: "))
Entre un numero: 40
Entre otro numero: 2
42.0
>>> int(input('Ingrese un numero: ')) + int(input('Ingrese otro numero'))
Ingrese un numero: 25
Ingrese otro numero: 6
31
... .
```

Conversión de tipos

¿Cuál es la salida de este código?.

```
>>> float ( "210" * int ( input ( "Ingrese un número: " )))
```

Ingrese un número: 2

- ☐ "420.0"
- ☐ "210210"
- ☒ 210210.0
- ☐ 420

Variables

Una variable es la que permite almacenar un valor al asignarlo a un nombre el cual puede ser utilizado para referirse al valor más adelante en el programa.

Para asignar una variable, se utiliza el signo **igual** (=). A diferencia de lo que se ha visto hasta el momento, esta instrucción de asignación no genera ninguna salida en la consola de Python.

```
>>> x=7
>>> print(x)
7
>>> print(x+3)
10
>>> print(x)
7
>>> |
```

Variables

¿Cuál es la salida de este código?.

```
>>> spam = "eggs"
```

```
>>> print( spam * 3 )
```



eggseggseggs



spamspamspam



"spamspamspam"

Variables

Las variables pueden ser reasignadas tantas veces como se quiera para cambiar su valor.

En Python las variables no tienen un tipo específico, es decir, a una variable se puede asignar un cadena y luego un entero.

```
>>> x=123.456
>>> print(x)
123.456
>>> x="Esto es una cadena"
>>> print(x+"!")
Esto es una cadena!
>>> |
```

Variables

¿Cuál es el valor del espacio en blanco para que dé la respuesta que se muestra?

```
>>> x = 5
```

```
>>> y = 7
```

```
>>> print ( x + y )
```

12

Nombre de variables

Los nombre de las variables tienen algunas restricciones;

1. Los únicos caracteres que permite Python para nombre variables son letras, números y guiones bajos.
2. Las variables no pueden empezar con número.

```
>>> este_es_un_nombre_normal = 7
>>> print(este_es_un_nombre_normal)
7
>>> 123abc = 3
SyntaxError: invalid syntax
>>> variable con espacios = 8
SyntaxError: invalid syntax
>>> |
```


Nombre de variables

Python es un lenguaje sensible a las mayúsculas y minúsculas.

Por ejemplo, **Nombre** y **nombre** son variables diferentes.

```
>>> Nombre = "Andres"
>>> nombre = "Felipe"
>>> print (Nombre + nombre)
AndresFelipe
>>> |
```

Variables

Intentar referenciar una variable que no se haya asignado, producirá un **ERROR**.

Se puede utilizar el comando **del** para eliminar una variable.

Intentar utilizar una variable luego de usar **del**, producirá un **ERROR**.

```
>>> var = "Hola mundo"
>>> var
'Hola mundo'
>>> abc
```

```
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <modul
    abc
NameError: name 'abc' is not defined
>>> del var
>>> var
```

```
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <modul
    var
NameError: name 'var' is not defined
>>> |
```

Variables

También se puede capturar el valor de una variable desde una entrada de un usuario.

```
>>> var = input ("Ingrese un numero: ")
Ingrese un numero: 123456
>>> print (var)
123456
>>> |
```

Variables

¿Cuál es la salida de estas instrucciones?

```
>>> spam = 2
```

```
>>> eggs = 3
```

```
>>> del spam
```

```
>>> eggs = 4
```

```
>>> spam = 5
```

```
>>> print (spam * eggs)
```

Operadores de asignación

Los operadores de asignación permiten escribir códigos como “`x = x + 3`”, que en forma abreviada se escribiría “`x += 3`”.

Esto mismo se puede hacer con otros operadores como: `-`, `*`, `/` y `%`.

```
>>> x = 2
>>> print (x)
2
>>> x += 3
>>> print (x)
5
>>> |
```

Operadores de asignación

¿Cuál es la salida de estas instrucciones?

```
>>> x = 4
```

```
>>> x *= 3
```

```
>>> x %= 5
```

```
>>> print ( x )
```

2

Operadores de asignación

Estos operadores pueden ser utilizados en otros tipos de datos además de números, como **cadenas**.

```
>>> x = "spam"
```

```
>>> print ( x )
```

```
spam
```

```
>>> x += "eggs"
```

```
>>> print ( x )
```

```
spameggs
```

Operadores de asignación

¿Cuál es la salida de estas instrucciones?

```
>>> x = "z"
```

```
>>> x *= 3
```

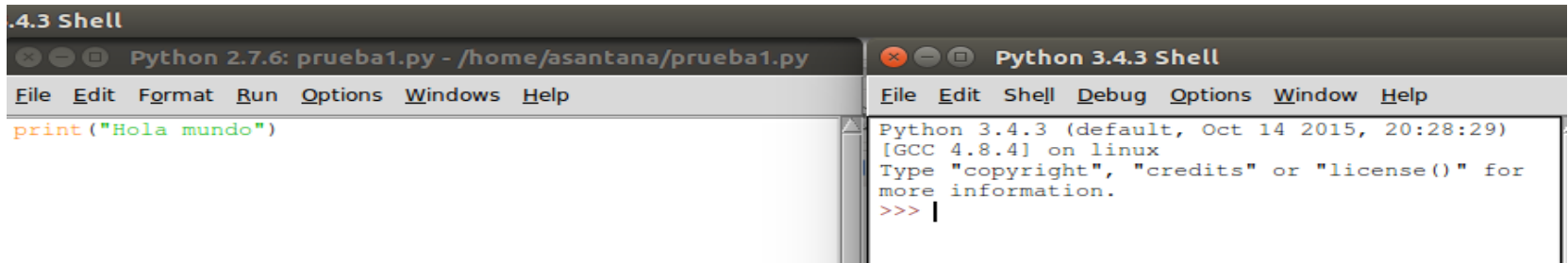
```
>>> print ( x )
```

zzz

Utilizando el Editor

Los programas reales se hacen con muchas líneas de código que se escriben en un archivo y luego se ejecutan en el interpretador de Python.

En el IDLE, esto se hace creando un archivo nuevo, en el cual se introduce en el código, se guarda el archivo y se ejecuta.



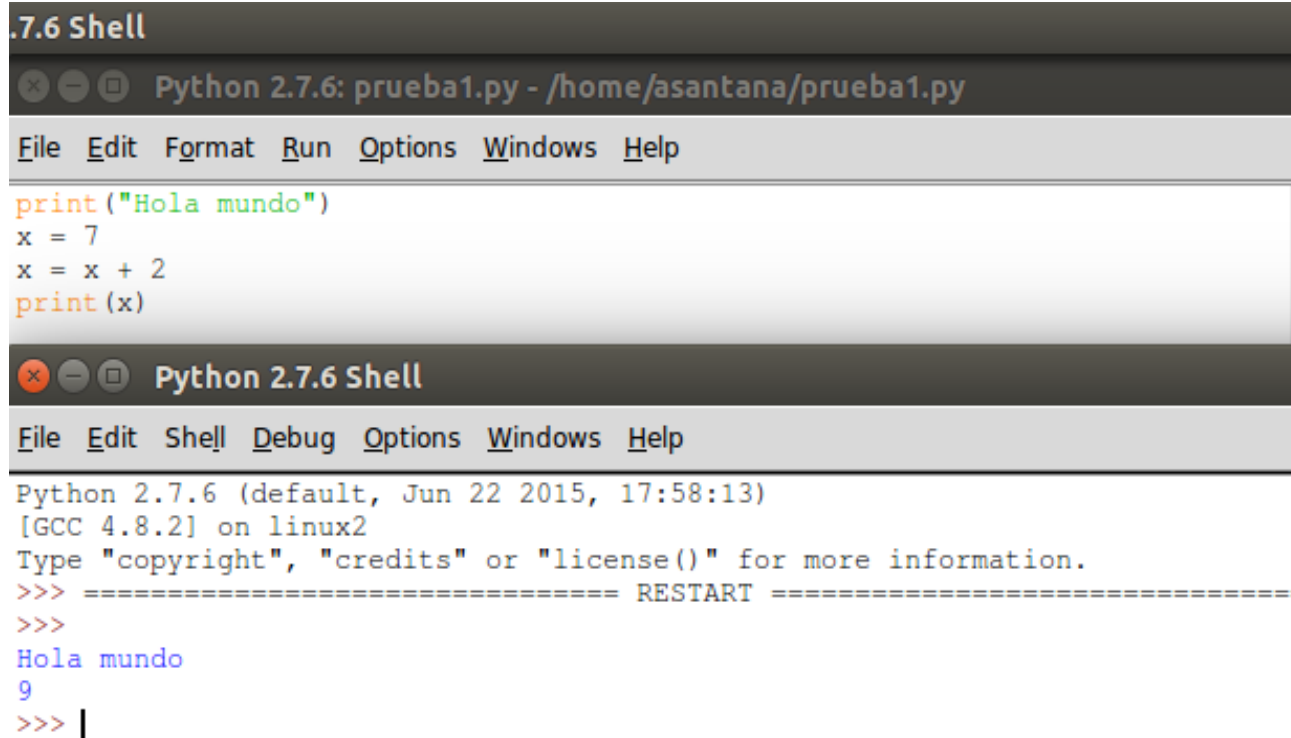
Utilizando el Editor

La creación del archivo nuevo se hace dando clic en el menú de la consola **File > New File** o utilizando en atajo **Ctrl+N**.

Existen otro atajos (*Shortcuts*) útiles para trabajar en el editor:

- **Ctrl+S** para guardar, se recomienda guardar frecuentemente los cambios que se vayan haciendo.
- **F5** que se utiliza para ejecutar el programa y que este se visualice en la consola.

Utilizando el Editor



The screenshot displays two windows from a Python IDE. The top window, titled 'Python 2.7.6: prueba1.py - /home/asantana/prueba1.py', contains a Python script with the following code:

```
print("Hola mundo")
x = 7
x = x + 2
print(x)
```

The bottom window, titled 'Python 2.7.6 Shell', shows the output of running the script. It includes the Python version (2.7.6), GCC version (4.8.2), and the system (linux2). After a restart, it displays the output of the script:

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Hola mundo
9
>>> |
```

Los archivos fuente de Python tienen la extensión **.py**

Repaso

¿Cuál es la salida de las siguientes instrucciones?

```
>>> spam = "7"  
>>> spam = spam + "0"  
>>> eggs = int(spam) + 3  
>>> print(float(eggs))
```

☒ 73.0

☐ 10.0

☐ 703

Repaso

¿Cuál es la salida de las siguientes instrucciones?

```
>>> word = input("Ingrese una palabra: ")  
Ingrese una palabra: "Hola"  
>>> print(word + ' mundo')
```



Hola mundo



"Holamundo"



'Holamundo'

Repaso

¿Cuál es la salida de las siguientes instrucciones?

```
>>> x = 5  
>>> y = x + 3  
>>> y = int(str(y) + "2")  
>>> print(y)
```

Repaso

¿Cuál es la salida de las siguientes instrucciones?

```
>>> x = 3
```

```
>>> num = 17
```

```
>>> print(num % x)
```

2

Booleanos

Hay dos tipos de valores booleanos: **True** (*verdadero*) y **False** (*falso*).

Pueden ser creados. Por ejemplo, al utilizar un operador de equivalencia **==**.

```
>>> var_booleana = True
>>> var_booleana
True
>>> 2 == 3
False
>>> "hola" == "hola"
True
>>> |
```


Booleanos

¿Cuáles son los dos tipos de valores booleanos en Python?



True y False



true y false



Verdadero y Falso

Comparación

Otro operador de comparación, el operador de desigualdad (**!=**), devuelve **True** si los elementos que están siendo comparados no son iguales y **False** si lo son.

```
>>> 1 != 1
False
>>> "once" != "doce"
True
>>> 2 != 10
True
>>> |
```

Comparación

También existe un operador para determinar si un número (decimal o entero) es **mayor** o **menor** que otro, estos operadores son **>** y **<** respectivamente.

```
>>> 7 > 5
True
>>> 10 < 10
False
>>> |
```

Los operadores de **mayor o igual** y **menor o igual** son **>=** y **<=**

```
>>> 7 <= 8
True
>>> 9 >= 9.0
True
>>> |
```

Comparación

Operadores de mayor que y menor que pueden utilizarse también para comparar cadenas de caracteres. (el orden alfabético de las palabras está basado en el orden alfabético de sus letras que las componen)

```
>>> "a" < "b"  
True  
>>> "avion" > "arbol"  
True  
>>> "carro" > "carta"  
False  
>>> |
```

Sentencia if

Puede utilizar sentencia **if** para ejecutar un código si una condición de cumple.

Si la expresión es evaluada como **True**, algunas sentencias son llevadas a cabo, de lo contrario no.

if *expresión:*
sentencia

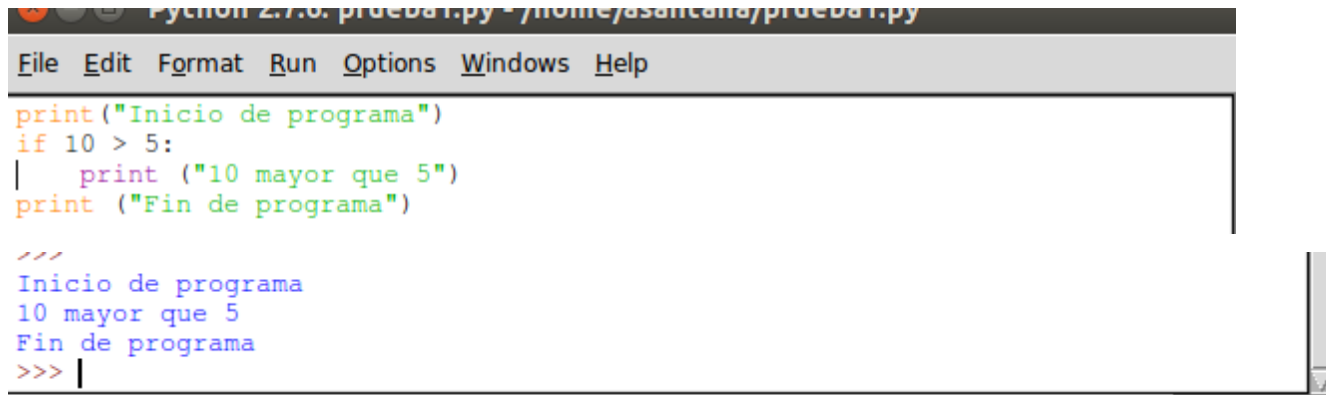
Sentencia if

Python usa **indentación** para delimitar bloques de códigos. Otros lenguajes como C, usan llaves para delimitar, pero en Python la indentación es obligatoria; los programas no funcionan sin esta. Como ve a continuación las sentencias en el **if** debe ser indentadas.

```
print("Hola mundo")
x = 7
y = 5
if x > 5:
    print ("Verdadero")
```

Sentencia if

La **indentación** que está dentro de la condición **if** se imprime al evaluar si 10 es mayor que 5. La siguiente línea no está indentada, lo que quiere decir que no pertenece a la condición **if**.



```
Python 2.7.6: prueba1.py - /home/asantana/prueba1.py
File Edit Format Run Options Windows Help

print("Inicio de programa")
if 10 > 5:
|   print ("10 mayor que 5")
print ("Fin de programa")

///
Inicio de programa
10 mayor que 5
Fin de programa
>>> |
```

Sentencia if

¿Cuál es la salida de las siguiente instrucciones?

```
spam = 7  
if spam > 5:  
    print("cinco")  
if spam > 8:  
    print("ocho")
```



nada



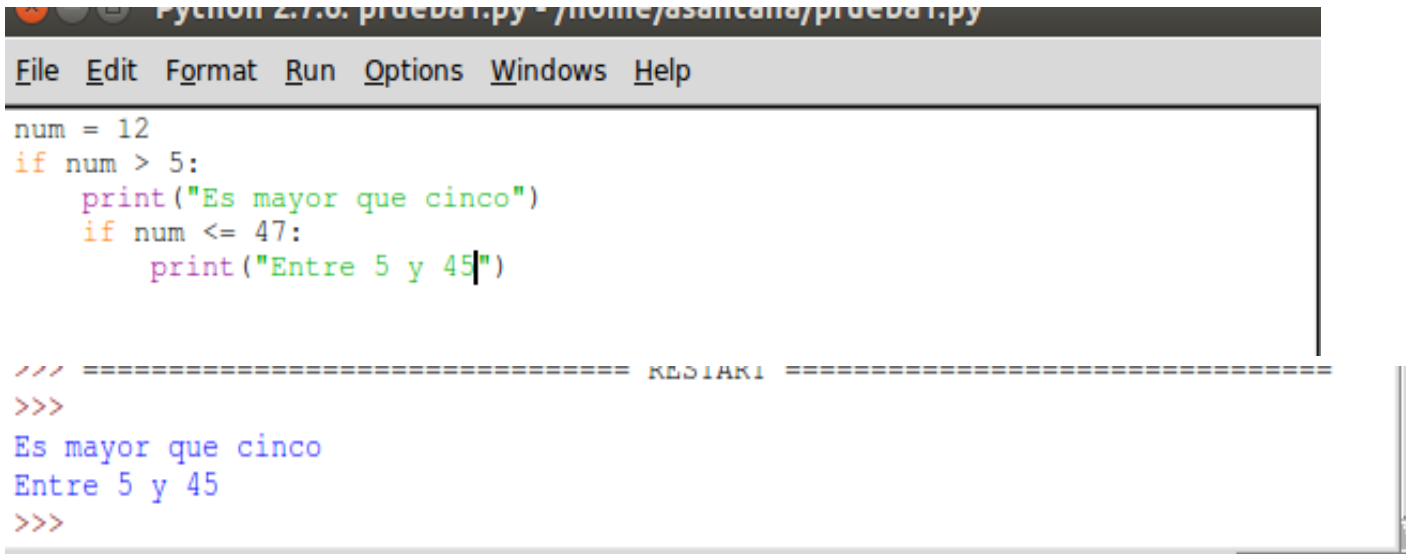
ocho



cinco

Sentencia if

Para llevar a cabo comparaciones más complejas, las sentencias **if** pueden ser **anidadas**.



```
Python 2.7.6: prueba1.py - /home/asantana/prueba1.py
File Edit Format Run Options Windows Help
num = 12
if num > 5:
    print("Es mayor que cinco")
    if num <= 47:
        print("Entre 5 y 45")

/// ===== RESIAR1 =====
>>>
Es mayor que cinco
Entre 5 y 45
>>>
```

Sentencia if

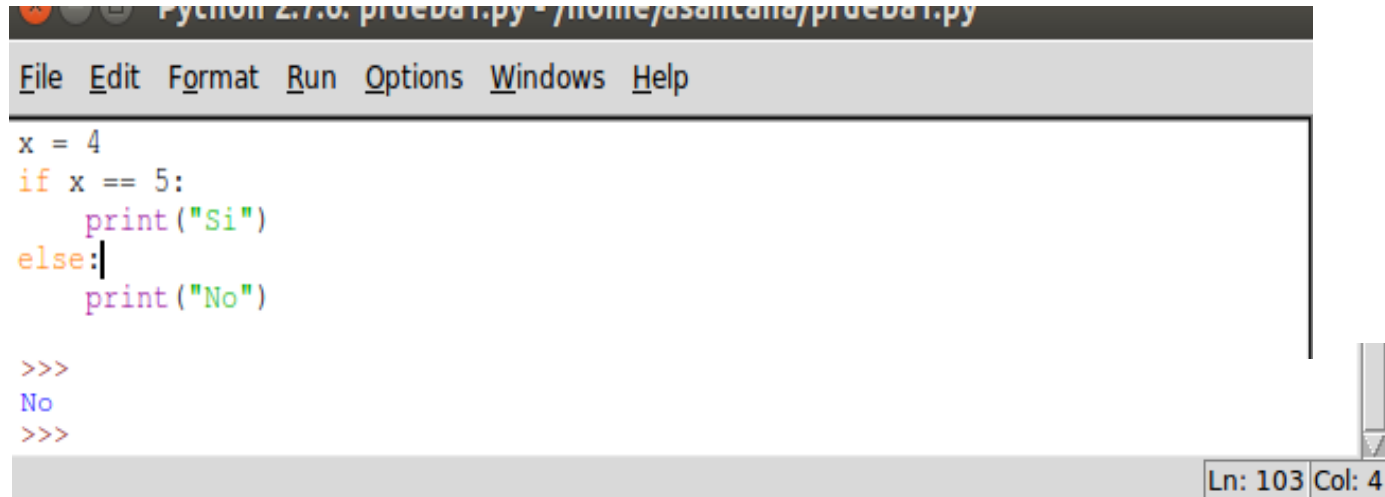
¿Cuál es la salida de las siguientes instrucciones?

```
num = 7
if num > 3:
    print("3")
    if num < 5:
        print("7")
        if num == 7:
            print("7")
```

Sentencia else

Una sentencia **else** le sigue a una sentencia **if** y contiene un código que es ejecutado cuando la sentencia **if** se evalúa como **False**.

Al igual que la sentencia **if**, el código dentro del bloque debe ser indentado



```
Python 2.7.6: prueba1.py - /home/asancana/prueba1.py
File Edit Format Run Options Windows Help
x = 4
if x == 5:
    print("Si")
else:
    print("No")

>>>
No
>>>
```

Ln: 103 Col: 4

Sentencia else

¿Cuál es la salida de las siguiente instrucciones?

```
if 1 + 1 == 2:  
    if 2 * 2 == 8:  
        print("if")  
    else:  
        print("else")
```

☐

if

☐

No imprime nada

☒

else

Sentencia else

Se pueden anidar las sentencias **if** y **else**.

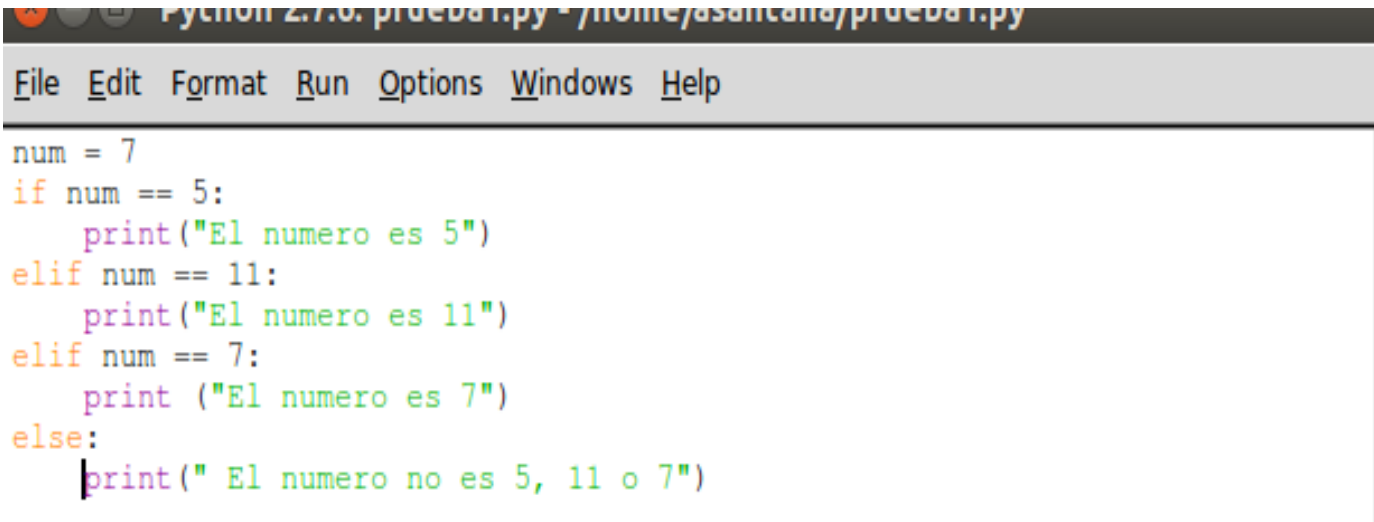
```
Python 2.7.6: prueba1.py - /home/asancana/prueba1.py
File Edit Format Run Options Windows Help

num = 7
if num == 5:
    print("El numero es 5")
else:
    if num == 11:
        print("El numero es 11")
    else:
        if num == 7:
            print("El numero es 7")
        else:
            print(" El numero no es 5, 11 o 7")

---
>>> ===== RESTART =====
>>>
El numero es 7
>>> |
```

Sentencia else

La sentencia **elif** (abreviatura de else if) es un atajo utilizado cuando se encadenan sentencias **if** con sentencias **else**.

A screenshot of a Python IDE window titled "Python 2.7.6: prueba1.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code in the editor is as follows:

```
num = 7
if num == 5:
    print("El numero es 5")
elif num == 11:
    print("El numero es 11")
elif num == 7:
    print ("El numero es 7")
else:
    print(" El numero no es 5, 11 o 7")
```

Lógica booleana

Se utiliza para realizar condiciones más complicadas para las sentencias if que dependen de más de una condición.

Los operadores booleanos de Python son **and**, **or** y **not**.

```
>>> 1 == 1 and 2 == 2
True
>>> 1 == 1 and 2 == 3
False
>>> 1 != 1 and 2 == 2
False
>>> 2 < 1 and 3 > 6
False
>>> |
```

```
>>> 1 == 1 or 2 == 2
True
>>> 1 == 1 or 2 == 3
True
>>> 1 != 1 or 2 == 2
True
>>> 2 < 1 or 3 > 6
False
>>> |
```

```
>>> not 1 == 1
False
>>> not 1 > 7
True
>>> |
```

NOTA: Python utiliza palabras para sus operadores booleanos, mientras que otros lenguajes usan símbolos como &&, || y !.

Lógica booleana

¿Cuál es el resultado de las siguientes instrucciones?

```
if not True:  
    print ("1")  
elif not (1 + 1 == 3):  
    print ("2")  
else:  
    Print("3")
```


Precedencia de operadores

Es la extensión de la idea matemática del orden de las operaciones para incluir otros operadores.

Por ejemplo, en el código siguiente, se ve que `==` tiene una mayor precedencia que `or`.

```
>>> False == False or True
True
>>> False == (False or True)
False
>>> (False == False) or True
True
... .
```

Precedencia de operadores

¿Cuál es el resultado de las siguientes instrucciones?

```
If 1 + 1 * 3 == 6:
```

```
    print ("Si")
```

```
else:
```

```
    Print("No")
```

No

Precedencia de operadores

la más baja.

Operador	Descripción
**	Exponenciación (elevar a la potencia)
~ + -	Complemento, más unario y menos unario (los nombres de método para los últimos dos son +@ y -@)
* / % //	Multipliación, división, módulo y división entera
+ -	Adición y sustracción
>> <<	Desplazamiento bit a bit a la derecha y a la izquierda
&	'AND' bit a bit
^	'OR' exclusivo bit a bit y 'OR' regular
<= < > >=	Operadores de comparación
<> == !=	Operadores de igualdad
= %= /= //=-= += *= **=	Operadores de asignación
is is not	Operadores de identidad
in not in	Operadores de pertenencia
not or and	Operadores lógicos

Precedencia de operadores

¿Cuál es el resultado de las siguientes instrucciones?

x = 4

y = 2

if not 1 + 1 == y or x == 4 and 7 == 8:

 print("Si")

elif x > y:

 print("No")

☐ Si No

☒ No

☐ Si

Ejercicios

1. Hacer un algoritmo que, dados dos valores numéricos A y B, escriba un mensaje diciendo si A es mayor, menor o igual a B.
2. Diseñe un algoritmo que permita obtener el promedio de 5 notas de un curso, valoradas de 0 a 5 y ponderadas con un 30, 15, 15, 20, 20%, y escriba aprobado si la calificación es mayor o igual a 3. Finalmente, independiente de si ganó o perdió, muestre el promedio.
3. Dados los datos A, B y C que representan números enteros diferentes, construir un algoritmo para escribir estos números en forma descendente.

Ejercicios

4. Cierta universidad para liquidar el pago de matrícula de un estudiante le exige los siguientes datos: Número de inscripción, Nombres, Patrimonio, Estrato.

La universidad cobra un valor constante para cada estudiante de \$50.000. Si el patrimonio es mayor que \$2'000.000 y el estrato superior a 3, se le incrementa un porcentaje del 3% sobre el patrimonio.

Hacer un algoritmo que muestre: Número de inscripción, Nombres, Pago de matrícula