1

Una sentencia compuesta es un conjunto de sentencias agrupadas delimitadas con algún delimitador, como begin end en Pascal o {} en Javascript.

2

C define la sentencia de asignación como una expresión con efectos laterales, ya que devuelven valores.

Ejemplo:

a=b=c=0;

if (i=30) printf(“Es verdadero”)

En la teoría está esta pregunta, lo dejo acá por las dudas:

¿C permite cualquier expresión que denote un l-valor? No.

3

Si, por ejemplo:

int main()

{

int a = 1;

int b = 2;

int c = 3;

a = b = c = 0;

printf("%d\n", a);

printf("%d\n", b);

printf("%d\n", c);

}

Si se lee de derecha a izquierda los valores de salida son:

0

0

0

Si se lee de izquierda a derecha los valores de salida son:

3

2

0

4

Circuito corto: al evaluar una condición, cuando un elemento da un resultado determinante deja de evaluar el resto.

Circuito largo: al evaluar una condición, todos los elementos se evalúan sin importar el resultado parcial.

Ejemplo:

bool = False;

a = 1;

b = 0;

if (bool and (a/b == 2)):

print(‘Entró al if’)

Con circuito corto:

Al evaluar el and del if, bool es falso, por lo que no entra en el if e ignora el resto de la evaluación.

Con circuito largo:

Al evaluar el if, bool es falso, sigue evaluando y a/b da un error (división por cero).

5

### Delphi

If sin else:

IF CONDICIÓN then INSTRUCCIÓN;

If con else:

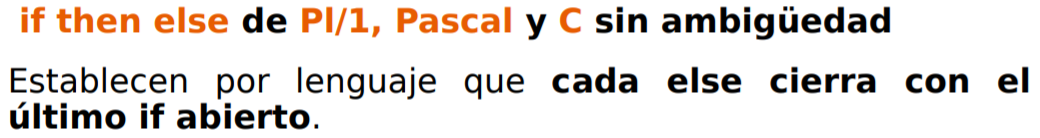
IF CONDICIÓN then INSTRUCCIÓN Else INSTRUCCIÓN;

o

IF CONDICIÓN then begin INSTRUCCIÓN 1; INSTRUCCIÓN ...; INSTRUCCIÓN n; end Else begin INSTRUCCIÓN 1; INSTRUCCIÓN ...; INSTRUCCIÓN n; end;

Se saca el ; al final del bloque then y se pone else, y se pone en el bloque del else.

### C



Ejemplo:

if (test expression) {

// statements to be executed if the test expression is true

}

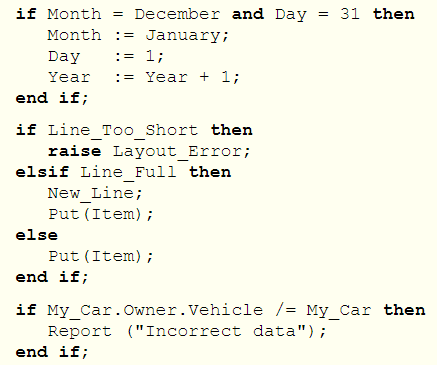
else {

// statements to be executed if the test expression is false

}

### Ada

Cualquier if termina con un end if, un else tiene que estar dentro del if - end if.



### Python

En python, los bloques if se definen

if (sentencia):

#bloque

El else es de la misma manera, pero debe ir siempre a continuación de un bloque if o elif, como:

if (sentencia):

#bloque

else:

#otro bloque

Ejemplo con elif=

if hora <= 6 and hora >=12:

print ‘Buenos días!!’

elif hora >12 and hora < 20 :

print ‘Buenas tardes!!’

else :

print ‘Buenas noches!!’

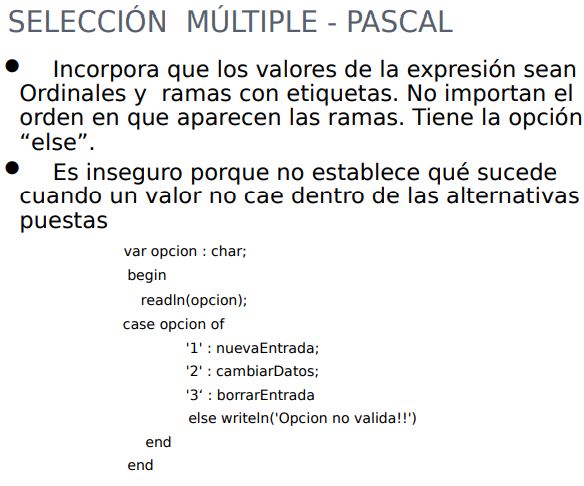
If then else de Python, sin Ambigüedad (significa que no establecía por lenguaje cómo se asociaban los else con los If abiertos) y legible

: es obligatorio al final del if, else y del elif ′

La indentación es obligatoria al colocar las sentencias correspondientes tanto al if, else y el elif

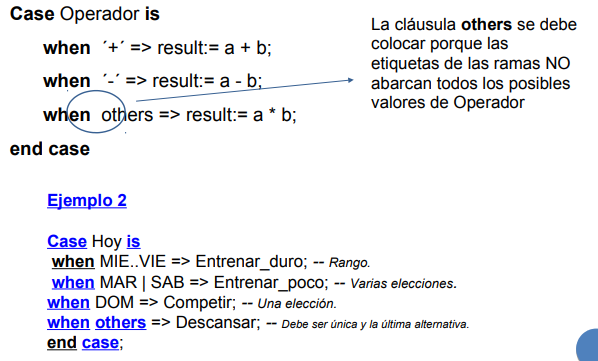
6

Pascal:



Ada:

Las expresiones pueden ser solamente de tipo entero o enumerativas. En las selecciones se debe estipular todos los valores posibles que puede tomar la expresión. Tiene la cláusula Others qué se puede utilizar para representar a aquellos valores que no se especificaron explícitamente. Si NO se coloca la rama para un posible valor o si NO aparece la opción Others en esos casos, no pasará la compilación.



Python: no existe un switch, se implementa con funciones y diccionarios de la siguiente manera

def switch\_demo(argument):

switcher = {

1: "January",

2: "February",

3: "March",

4: "April",

5: "May",

6: "June",

7: "July",

8: "August",

9: "September",

10: "October",

11: "November",

12: "December"

}

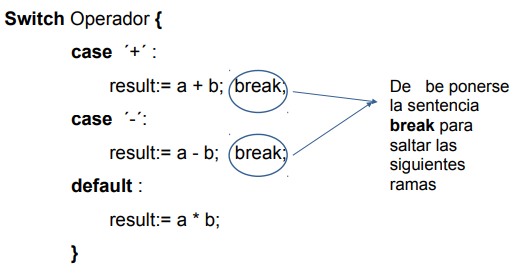
print switcher.get(argument, "Invalid month")

C:

Cada rama es etiquetada por uno o más valores constantes. Cuando la opción coincide con una etiqueta del Switch se ejecutan las sentencias asociadas y se continúa con las sentencias de las otras entradas.

Existe la sentencia break, que provoca la salida.

Tiene una cláusula default que sirve para los casos que el valor no coincida con ninguna de las opciones establecidas.



La diferencia de C con las demás implementaciones es que necesitamos la palabra break para impedir que se ejecuten las demás condiciones, cuando en los demás lenguajes una vez que se ejecuta el código referente a una condición se sale del bloque switch.

7

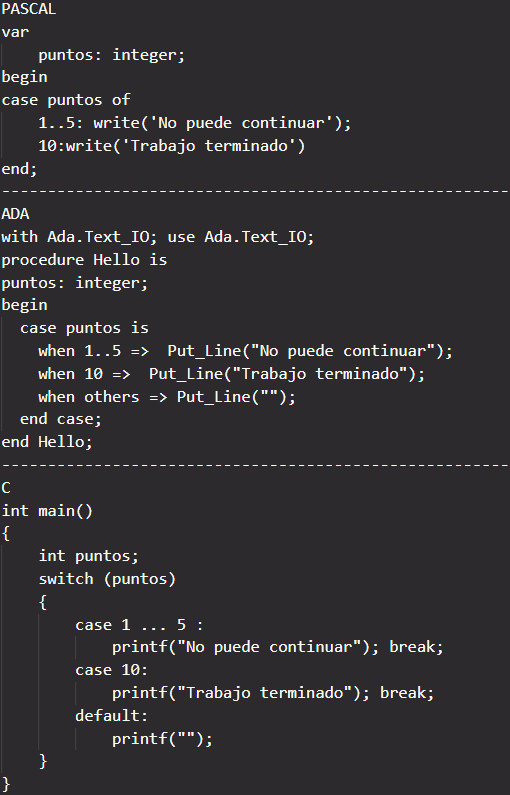
En pascal la variable del for se modifica por lo cual puede caer en loop infinito.

En ada la variable del for es tomada como variable local, por lo cual en A no se modifica.

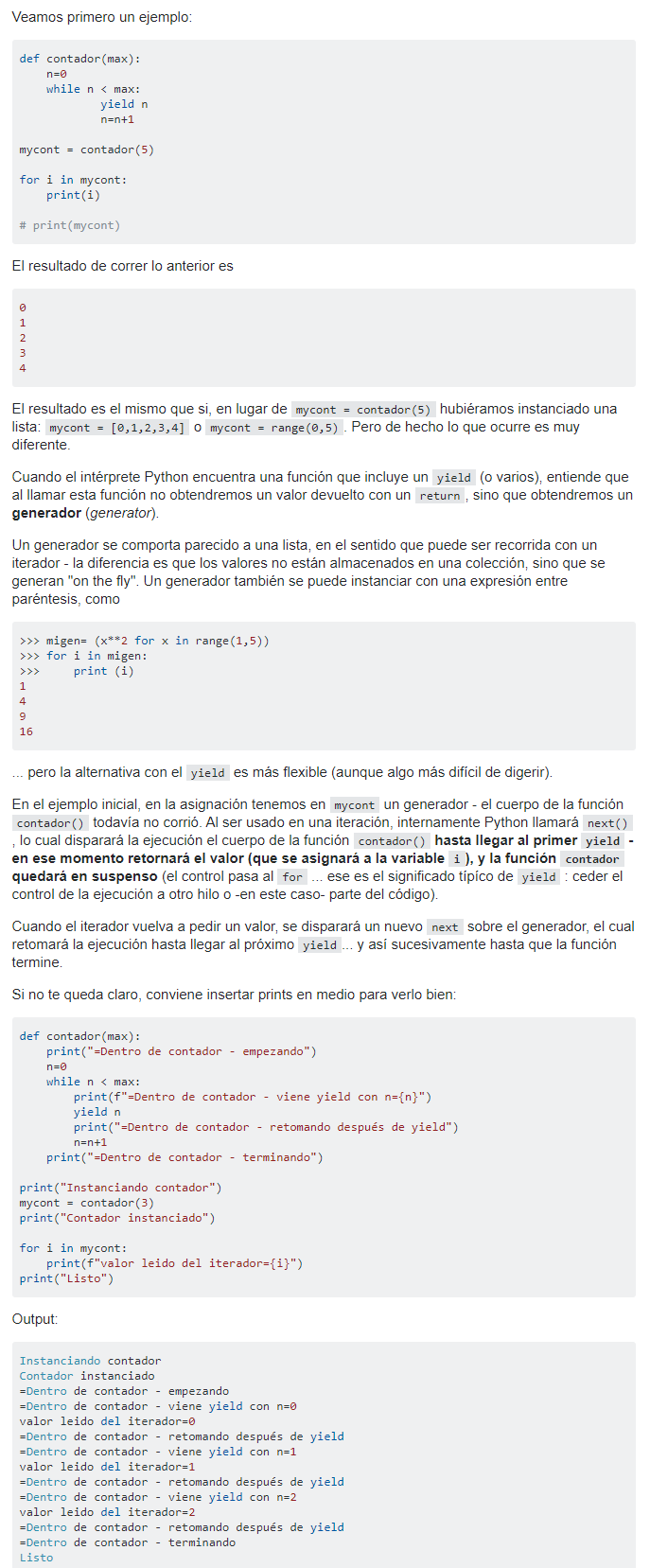
8

Ignorando el hecho de que el código de pascal está mal (necesita of en el case, y comillas simples):

En ADA puede dar un error si no especificamos la sentencia others, y en C puede funcionar mal si no ponemos el break en cada bloque o el default:



9



La principal diferencia es que (utilizando otro elemento de Python llamado generador) yield “pausa” la ejecución de una función, por lo que en la próxima invocación de la misma se continúa desde ese punto.

El yield podría ser útil cuando querés, por ejemplo, obtener valores de una lista por demanda. Supongamos que tenemos una lista enorme, para no tener que cargarla toda en memoria vas usando yield.

https://es.stackoverflow.com/questions/6048/cuál-es-el-funcionamiento-de-yield-en-python

10

El método **map()** crea un nuevo array con los resultados de la llamada a la función indicada aplicados a cada uno de sus elementos.

map llama a la función callback provista **una vez por elemento** de un array, en orden, y construye un nuevo array con los resultados. callback se invoca sólo para los índices del array que tienen valores asignados; no se invoca en los índices que han sido borrados o a los que no se ha asignado valor.

map no modifica el array original en el que es llamado (aunque callback, si es llamada, puede modificarlo).

Sintaxis:

*var nuevo\_array = arr.map(function callback(currentValue, index, array) {*

*// Elemento devuelto de nuevo\_array*

*}[, thisArg])*

Una alternativa a map, podría ser utilizar **forEach**.

Funciona parecido a map, pero en vez de esperar un return si o si del nuevo array, este solo recorre el array, se podría crear uno nuevo y agregarle los valores del array original modificados.

Otra alternativa podría ser **Array.from()**, el cual crea una nueva instancia de Array a partir de un objeto iterable. Se puede modificar cada valor del array de la siguiente forma:

*console.log(Array.from([1, 2, 3], x => x + x));*

*// expected output: Array [2, 4, 6]*

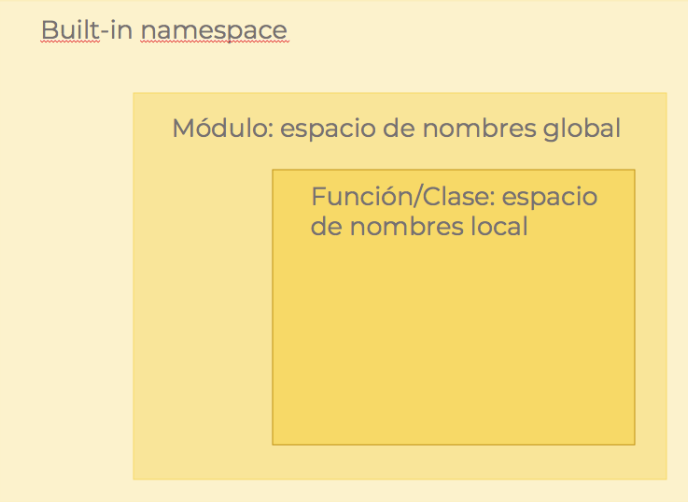
11

En Python todos son objetos, y un nombre o identificador es la forma que existe en Python de referenciar a un objeto concreto. Un espacio de nombres es una colección aislada de nombres (o identificadores) que referencian a objetos.

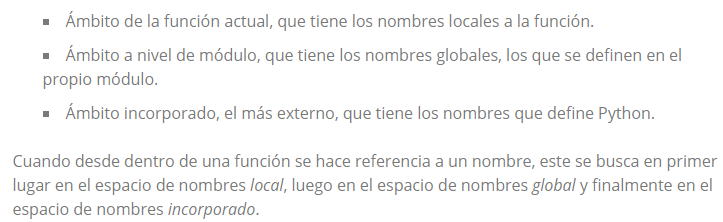
Cuando accedemos a un intérprete de Python o ejecutamos un programa, todos los identificadores que define el lenguaje son añadidos a un espacio de nombres al que es posible acceder desde cualquier punto de un script. Se lo llama built-in namespace.

Además, cada módulo en Python crea su propio espacio de nombres global. Como nombramos, los espacios de nombres están aislados. Esa es la razón por la que en diferentes módulos se pueden usar los mismos nombres y estos no interfieren entre sí.

A su vez, en un módulo existen funciones y clases. Cuando se invoca a una función se crea un espacio de nombres local asociado a dicha función que contiene todos los nombres definidos dentro de la misma (sucede algo similar para las clases).



Por lo tanto hay 3 ámbitos para una variable según el espacio de nombres que utiliza:



Y se pueden utilizar palabras como **nonlocal** o **global** para cambiar el espacio de nombres en el que se debe buscar la variable.

En PHP funciona igual, y acá están las formas de acceder a una variable:

