1 - Una sentencia simple es aquella en donde hay una sola instruccion, mientras que una sentencia compuesta puede agrupar varias ejecuciones mediante el uso de delimitadores.

2 - C define la sentencia de asignación como una expresión con efectos colaterales. Evalua de derecha a izquierda. Requiere del lado izquierdo de la asignacion un l-valor.

3 - Si, puede suceder cuando una expresion tiene efecto sobre algo mas que solo calcular un valor y asignarlo a una variable. Ejs:

1. Expresiones con incremento o decremento:

1 int a = 5;

2 int b = ++a;

En este ejemplo, la expresión ++a incrementa el valor de a en 1 antes de asignarlo a b. El efecto lateral es que el valor final de b será 6, no 5.

2. Expresiones con operaciones de asignación compuestas:

1 int c = 10;

2 c += 5;

En este caso, la expresión c += 5 incrementa el valor de c en 5 y lo asigna de nuevo a c. El efecto lateral es que el valor final de c será 15.

3. Asignación en condición:

1 if (i = 30) {...}

Si se usa el operador de asignación (=) en la condición del if, la expresión se evaluará como verdadera siempre que el valor asignado no sea cero. El efecto lateral en este caso es que el valor de i se modifica a 30 como resultado de la asignación. Además la condición del if siempre se evaluará como verdadera.

4 - Que un lenguaje utilice circuito corto para la evaluacion de una expresion significa que la evaluacion terminara cuando con el resultado de solo una expresion alcance para determinar el resultado de toda la expresion. Mientras que si utiliza circuito largo evaluara todas las expresiones, pudiendo haber modificado algo mas que un circuito corto.

Ej circuito corto: Operador "y" (&&) >>> Si la primera parte de la expresión es falsa, el resultado final será falso sin importar el valor de la segunda parte. Por lo tanto, en un circuito corto, si la primera parte es falsa, la segunda parte no se evalúa.

Ej circuito largo: Operador "y" (&&) >>> En un circuito largo, ambas partes de la expresión se evalúan independientemente del valor de la primera parte. Esto asegura que se realicen todas las comprobaciones necesarias.

5 - En lenguajes como Delphi, ADA y C, la regla de asociación del else se basa en la coincidencia de las llaves o palabras clave correspondientes al if más cercano sin un else.

Python utiliza una indentación especial para delimitar bloques de código en lugar de llaves o palabras clave como begin y end. En Python, la asociación del else con el if correspondiente se basa en la indentación. El else se asocia con el if anterior que tenga la misma indentación.

6 - En C se utiliza el constructor Switch seguido de expresión

. Cada rama (Case) es etiquetada por uno o más valores constantes (enteros o char)

. Si coincide con una etiqueta del Switch se ejecutan las sentencias asociadas y se continúa con las sentencias de las otras entradas. (salvo exista un break)

. Existe la sentencia break, que provoca la salida de cada rama, sino continúa.

. Existe opción default que sirve para los casos que el valor no coincida con ninguna de las opciones establecidas, es opcional

Pascal y ADA tienen una construcción switch-case para expresar múltiples selecciones similar a C, Python utiliza estructuras if-elif-else y, a partir de la versión 3.10, ofrece la opción de usar match-case, aunque con diferencias significativas respecto al switch-case de C.

7 -

1. En ADA y Pascal estándar no se puede modificar manualmente los valores de las variables de los iteradores. Por lo que habría error.
2. En ADA y Pascal no estándar se pueden modificar los valores sin que salte un error pero sin embargo el código tendría muchos problemas.

8 - Como el bloque else es opcional en Pascal y en este caso no esta, podría haber un error si el valor de puntos no coincide con ninguna de las sentencias en la lista.

En C se utilizaria la clausula Switch, mientras que en ADA seria mas parecido a Pascal utilizando Switch-Case.

9 - La diferencia principal entre el generador yield de Python y el return de una función es que yield permite generar una secuencia de valores en lugar de retornar un solo valor como lo hace return.

Un ejemplo útil de uso de yield sería una función que genere una secuencia infinita de números pares. En lugar de calcular todos los números pares hasta un límite y almacenarlos en una lista, podemos utilizar un generador con yield para obtener los números pares uno a uno, evitando así almacenar todos los números en memoria.

10 - La función map en JavaScript es una función de orden superior que se utiliza para transformar los elementos de un arreglo original y generar un nuevo arreglo con los resultados de aplicar una función a cada elemento del arreglo original.

Similar a stream en Java.

11 - Un espacio de nombres es un contenedor lógico que se utiliza para agrupar y organizar elementos, como variables, funciones, clases u otros identificadores, dentro de un programa. Su propósito principal es evitar colisiones de nombres y proporcionar un contexto separado para los elementos con el mismo nombre. Un espacio de nombres permite que múltiples elementos con el mismo nombre coexistan en un programa sin conflicto, ya que cada elemento se encuentra dentro de un espacio de nombres único. Al utilizar espacios de nombres, se puede acceder a los elementos dentro de ellos mediante una notación de calificación, utilizando el nombre del espacio de nombres seguido de un separador (.) y luego el nombre del elemento específico. Esto ayuda a evitar ambigüedades y asegura que se esté haciendo referencia al elemento correcto.

En PHP, por ejemplo, se puede definir y utilizar espacios de nombres utilizando la palabra clave namespace y se pueden importar elementos utilizando use. Además, los espacios de nombres pueden estar organizados en una estructura de directorios que refleja la estructura de los archivos fuente.