Python#	C#	JAVA
Comentarios en línea: #	Comentarios en línea: //[1]	Comentarios en línea: //
Comentarios en múltiples líneas comienzan y terminan con """	Comentarios en múltiples líneas comienzan con /* y terminan con */	Comentarios en múltiples líneas comienzan con /* y terminan con */
if…, if…else, if…elif	if…, if…else, switch…case…default	if…, if… else, if… else if… else, switch… case… default,
for i in range(0, 3)…	for (int i = 0; i <=3; i++)…, foreach…[2]	for(inicialización, finalización, incremento)
while (i < 3)…	while (i < 3)…	While(i<3)
and, or, not	& y &&[3], | y |[3]|, !	& y &&, | y |, !,
<, <=, >, >=, ==, !=	<, <=, >, >=, ==[4], !=	<, <=,>, >=, ==, !=
+, -, *, /, +=, -=, *=, /=	+, -, *, /, +=, -=, *=, /=[5]	+, -, *, /, +=, -=, *=, /=, %=
bool	bool, Boolean[6]	boolean
float	float, Single[7]	Float, double
int	int, Int32[8]	Byte, short, int, long
str	string, String[9]	Char, String

[1] Los comentarios en línea que comienzan con tres /// tienen un significado especial para generar documentación externa. Ver: https://docs.microsoft.com/en-us/dotnet/csharp/codedoc

[2] La cláusula foreach en C# es similar a la cláusula for en Python cuando se usa con iterables.

[3] La diferencia es si se evalúa el segundo operando. Ver https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/conditional-and-operator y https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/conditional-or-operator

[4] En C# el operador == compara si dos objetos son el mismo; excepto para las instancias de la clase String, donde compara el valor. El operador == puede ser sobrescrito.

[5] Mientras /= aplicado a objetos de tipo int en Python da como resultado un objeto de tipo float, en C# da un objeto de tipo int siempre. Ver https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/division-operator.

[6] En C# la palabra clave bool es un alias para el tipo System.Boolean.

[7] En C# la palabra clave float es un alias para el tipo System.Single.

[8] En C# la palabra clave int es un alias para el tipo System.Int32.

[9] En C# la palabra clave string es un alias para el tipo System.String.

Asignar el valor y a la variable x: `x = y`	Asignar el valor x a la variable y del tipo T: `T x = y`[10]	T x=y.
Crear una instancia de una clase C y asignarla a la variable x: `x = C()`	Crear una instancia de una clase C y asignarla a la variable x: `C x = new C();`	C x= new c();
`float(…)`	`Convert.ToSingle(…)`, `Single.Parse(…)`[11]	`float a= Float.valueOf(str)` `float.parseFloat(str)`
`int(…)`	`Convert.ToInt32(…)`, `Int32.Parse(…)`[11]	`Math.round(float a)`
`str(…)`	`Int32.ToString(…)`, `Single.ToString(…)`[11]	`String s = Satring.valueOf(f);`
Cuando `demo` es una variable que contiene una instancia de `string`, `demo[0]` referencia el primer carácter, `demo[-1]` el último, `demo[2:4]` referencia una porción del segundo al cuarto carácter, y `demo[:4]` una porción del primero al cuarto carácter	Cuando `demo` es una variable que contiene una instancia de `String`, `demo[0]` referencia el primer carácter; no hay un equivalente en C# para acceder al último carácter[12] o a una porción[13]	
`def…`	No hay un solo equivalente en C#; para los métodos la sintaxis depende de la visibilidad, si retorna un resultado o no, el tipo de datos del resultado, y otros factores.	

[10] Aquí se muestra la declaración de la variable x del tipo T y la asignación del valor y a la variable x en la misma sentencia; es equivalente a `T x;` y luego `x = y;`. Vean que en Python no se declara el tipo de la variable, mientras en C# sí. La variable x es del tipo de y en Python, no es que no tenga tipo. Algo similar ocurre en C# al usar la palabra clave `var` al declarar una variable: `var x = y;` es equivalente a `T x = y;` si T es el tipo de y.

[11] Mientras `float()`, `int()` y `str()` son cast -conversión de tipos- en Python, los mostrados como correspondientes en C# son métodos; el concepto de cast existe en C# pero no se usa aquí como en Python.

[12] El último carácter se accede con `demo[demo.Lenght - 1]`.

[13] Una porción se obtiene con `demo.Substring(2, 2)`; noten que el segundo argumento es la cantidad de caracteres y no el final de la Proción.

`class…`	`class…`	`Class nombre{`
`self`	`this`[14]	`this`
`@classmethod`	`static`	`Static`
`cls`	El nombre de la clase	El nombre de la clase
`pass`	Un bloque vacío `{}` o un `;` puede ser similar en algunos casos, pero no existe una equivalencia exacta	{}
`with…`	`using…`[15] es similar, pero no existe una equivalencia exacta	`import`
`print(…)`	`Console.WriteLine(…)`	`System.out.println(..)`
`input()`	`Console.ReadLine()`[16]	`Entrada= System.console().readLine()`
`import`	No hay un equivalente exacto en C# porque los ensamblados -análogos a los módulos o paquetes en Python- sólo pueden ser cargados dinámicamente mediante una API, a diferencia de Python que siempre son cargados dinámicamente. `using` es una directiva en C# que permite referenciar tipos en un espacio de nombres sin calificarlos.	import

[14] La primera variable de un método en Python referencia al objeto que recibe el mensaje que ocasiona la ejecución de ese método, y suele llamarse `self`. En C# esa referencia se obtiene mediante la palabra clave `this`, y no es un parámetro del método

[15] En C# `using` es tanto una directiva para importar un módulo como una sentencia para asegurar la ejecución de destructores.

[16] Mientras `input` en Python permite mostrar un mensaje además de leer un valor, `Console.ReadLine()` en C# sólo lee un valor; para mostrar un mensaje, puedes usar `Console.Write()` o `Console.WriteLine()`.