Sintaxis y Semántica de los Lenguajes

GRUPO N° 5	CURSO: K2053

PROFESOR: Roxana Leituz

ASISTE LOS DÍAS: Sábado

EN EL TURNO: Mañana

TRABAJO PRÁCTICO N°: 1

TÍTULO: C# vs Python

INTEGRANTES	
Dimitri Isakow (164.780-5)	
Nahuel Alejandro Garcia (208.997-0)	
Martín Gonzalo Larrart (209.042-9)	
Pedro Jose Nicolas Alvarez (177.232-6)	

C#

Es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET).
Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Historia:

En abril de 1999, Anders Hejlsberg formó un equipo con la misión de desarrollar un nuevo lenguaje orientado a objetos.
En 2002 Microsoft lanzó su framework de desarrollo .NET. En aquel momento la mayoría de la comunidad de desarrolladores bajo las tecnologías de Microsoft desarrollan aplicaciones para ordenadores de escritorio en la plataforma Visual Basic y al aparecer .NET se introdujeron varios lenguajes para la plataforma, siendo los primeros en aparecer VB.NET y C#. Responde a la necesidad de crear un lenguaje con una sintaxis similar a C, como la mayoría de los lenguajes populares, entre ellos uno de sus grandes competidores: Java. Actualmente es uno de los lenguajes más usados en la industria del software.

Características

- ➔ Multiplataforma, ejecutable en los sistemas más comunes como Windows, MacOs, Linux
- ➔ Sintaxis similar a C, C++, Java y otros
- ➔ Lenguaje de paradigma de programación orientada a objetos, con expresiones de control heredadas de la programación estructurada
- ➔ Fuertemente tipado (tipado estático)
- ➔ Lenguaje moderno con actualizaciones de mejoras frecuentes
- ➔ Dispone de un nutrido conjunto de librerías
- ➔ Orientado a componentes
- ➔ POO (**P**rogramación **O**rientada a **O**bjetos): modelo de programación en el que el diseño de software se organiza alrededor de datos u objetos, en vez de usar funciones y lógica.

BNF Usado en el código C# Benchmark

tokens: [
	INCREMENTAR = ‘++’
	RESTA= ‘-’
]
<palabrasReservadas> ::= for | new | return | this
<operadores> ::- INCREMENTAR | RESTA
<letra> ::- a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<digito> ::- 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<accesosModificadores> ::- public | private
<tiposModificadores> ::- static | new | readonly
<tiposPrimitivos> ::- int | bool | byte | var | double
<tiposGenericos> ::- List
<tipos> ::- <tiposModificadores> | <tiposPrimitivos> | <tiposGenericos>

Python

Es un lenguaje de programación ampliamente utilizado en desarrollo de software, ciencia de datos y machine learning. Emplea facilidad de uso y permite ser ejecutado en varias plataformas.

Historia

Guido Van Rossum, un programador de computación de los Países Bajos, creó Python. Comenzó en 1989 en el Centrum Wiskunde & Informatica (CWI), en principio como un proyecto de afición para mantenerse ocupado durante las vacaciones de Navidad. El nombre del lenguaje se inspiró en el programa de televisión de la BBC “Monty Python’s Flying Circus”, ya que era un gran aficionado del programa.
La primera versión del código (versión 0.9.0) se publicó en 1991. Incluía buenas características, como algunos tipos de datos y funciones para la gestión de errores.
La versión 1.0 se lanzó en 1994 con nuevas funciones para procesar fácilmente una lista de datos, como la asignación, el filtrado y la reducción.
Luego la versión 2.0 se lanzó el 16 de octubre de 2000, con nuevas características útiles para los programadores, como la compatibilidad con los caracteres Unicode y una forma más corta de recorrer una lista.
El 3 de diciembre de 2008, se lanzó la versión 3.0. Incluía características como la función de impresión y más soporte para la división de números y la gestión de errores.

Características

- ➔ Interpretado: Ejecuta directamente el código línea por línea. Si existen errores en el código del programa, su ejecución se detiene. Lo que implica detectar errores en el código con rapidez.
- ➔ Fácil de usar: utiliza palabras similares a las del inglés. Los bloques de funciones o condiciones no usan corchetes como otros lenguajes habitualmente, en este caso usa indentaciones.
- ➔ Tipado dinámico: No necesita definir tipo de datos explícitamente, el interpretador asigna el tipo de datos de las variables en tiempo de ejecución.
- ➔ Alto nivel: Se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, como C#, PHP, Java, etc. Los programadores no necesitan preocuparse sobre las funciones bajo capot.
- ➔ POO (**P**rogramación **O**rientada a **O**bjetos): Al igual que C# este lenguaje también es Programación Orientada a Objetos.

BNF Usado en el código Python

tokens: [
    INCREMENTAR = ‘++’
    RESTA= ‘-’
    DIVISION=’/’
    MULTIPLICACION=’*’
]
<palabrasReservadas> ::= for | in | return | import
<operadores> ::- DIVISION | RESTA | MULTIPLICACION
<letra> ::- a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<digito> ::- 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<accesosModificadores> ::- def
<tiposPrimitivos> ::- int | str
<tipos> ::- <tiposPrimitivos>

Benchmark

La prueba de rendimiento que realizamos es insertar 1 millón de elementos de clase, aplicamos hash criptográfico SHA-256 en campo string, luego repetimos el proceso 5 veces. La razón de este tipo de prueba es para realizar procesamiento más pesado, de esta forma se puede comparar bien el rendimiento. Las 5 iteraciones nos permiten tener resultados más precisos sobre nuestra prueba.
Para un resultado más correcto sin sesgo, realizamos la prueba con NET 5 y Python 3.9.7, ya que ambos se lanzaron en las mismas fechas.

Código C#

Instancia de SHA256

```
private static readonly SHA256 sha = SHA256.Create();
```

Clase

```
private class NombreNumero
{
  private string Name;
  private int Num;
  public NombreNumero(string name, int n)
  {
    this.Name = SHA256HexHashString(name);
    this.Num = n;
  }
}
```

Funciones esenciales

```
private static string ToHex(byte[] bytes)
{
   StringBuilder result = new StringBuilder(bytes.Length * 2);
   for (int i = 0; i < bytes.Length; i++)
      result.Append(bytes[i].ToString("x2"));
   return result.ToString();
}

private static string SHA256HexHashString(string str)
{
   return ToHex(sha.ComputeHash(Encoding.Default.GetBytes(str)));
}
```

Bloque Main

```
static void Main(string[] args)
{
   List<NombreNumero> lista = new List<NombreNumero>();
   List<double> miliseconds = new List<double>();
   for (int i = 0; i < 5; i++)
   {
      var start = Stopwatch.GetTimestamp();
      for(int j=0;j<1000000;j++)
         lista.Add(new NombreNumero(j.ToString(), i));
      var end = Stopwatch.GetTimestamp();
      var elapsed = new TimeSpan(end - start);
      Console.WriteLine($"Elapsed: {elapsed.TotalMilliseconds}ms");
      miliseconds.Add(elapsed.TotalMilliseconds);
      lista.Clear();
   }
   Console.WriteLine($"Avg. Miliseconds: {miliseconds.Average()}ms");
   Console.ReadKey();
}
```

Código Python

Librería y Funciones esenciales

```
import hashlib
import time
import datetime

m = hashlib.sha256()
def Average(lst):
    return sum(lst) / len(lst)
```

Clase

```
class NombreNumero:
    def ToHex(nomb:str):
        m.update(nomb.encode())
        hx = m.hexdigest()
        return hx
    def __init__(self, nomb: str, num: int):
        result = hashlib.sha256(nomb.encode())
        self.nombre = result.hexdigest()
        self.numero = num
```

Bloque “main”

```
unaLista = []
listAvg =[]
for i in range(5):
    start = datetime.datetime.now()
    for j in range(1000000):
        stra = str(j)
        unaLista.append(NombreNumero(stra,i))
    unaLista.clear()
    end = datetime.datetime.now()
    delta = end - start
    milliseconds = int(delta.total_seconds()*1000)
    listAvg.append(milliseconds)
    print(f"Elapsed: {milliseconds}")
print(Average(listAvg))
```

Resultado

	C#	Python	Diferencias
Tiempo de pruebas en milisegundos	1577 ms	1649 ms	72 ms
	1557 ms	1622 ms	65 ms
	1559 ms	1594 ms	35 ms
	1488 ms	1652 ms	164 ms
	1571 ms	1661 ms	90 ms
Promedio en milisegundos	**1550 ms**	**1635 ms**	***85.2 ms***

La diferencia de prueba de rendimiento entre estos 2 lenguajes es de 85.2 Milisegundos, esto significa que casi no hay suficiente diferencia para iniciar un debate en cuanto a cuál de los 2 lenguajes es mejor.

Conclusión

En conclusión en esta prueba C# resultó ser más rápido que python con una diferencia de 5.20%. Se debe principalmente a que C# Es un lenguaje de programación compilado, mientras que Python es un lenguaje de programación interpretado. Esto significa que cuando se ejecuta un programa escrito en C#, el código se compila en código de máquina, que es directamente ejecutado por el hardware de la computadora. Por otro lado, cuando se ejecuta un programa escrito en Python, el código es interpretado por el intérprete de Python, que luego genera una salida.