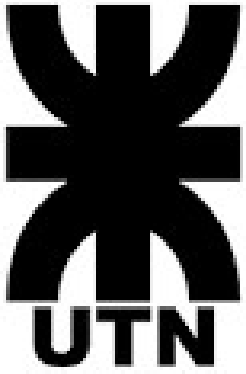
TP TEÓRICO

Materia: SSL

Comparación de Lenguajes: Haskell y Python



Fecha de entrega: 24/08/2023

Integrantes Grupo 25:

-CACACE, Guillermo Federico

-CALÓ, Ignacio

-GOMEZ PEREYRA, Manuel Francisco

- MAJER, Cecilia Alejandra

-TROSSERO, Agustín Francisco

Grupo 25 2023

Python vs. Haskell

Historia del Haskell

Haskell es un lenguaje de programación puramente funcional, cuya primera versión fue lanzada en 1990. Su nombre proviene del matemático Haskell Brooks Curry, que sentó las bases de los lenguajes funcionales con su trabajo sobre lógica combinatoria entre 1920 y 1960. Haskell está basado en el cálculo lambda, modelo que permite trabajar con funciones como objetos de primera clase. El logotipo del lenguaje contiene el símbolo de esta letra griega.

El objetivo de la creación del lenguaje fue unificar las características más importantes de los lenguajes funcionales, como las funciones de orden superior, evaluación perezosa, inferencia estática de tipos, tipos de datos definidos por el usuario, encaje de patrones y listas por comprensión. Es una excelente opción para minimizar las posibilidades de cometer errores y efectos secundarios al programar.

Este lenguaje es recomendable en tareas complejas de sectores como seguridad, tecnología financiera, blockchain y big data.

Entre sus características se destacan que es: funcional, polimórfico, de tipado estático y perezoso.

Tipado estático: Cada expresión cuenta con un tipo que es determinado durante la compilación. Los tipos compuestos deben coincidir, si no, el programa se rechazará por el compilador. Los tipos se convierten en un lenguaje que expresa la construcción de los desarrollos.

Funcional: A diferencia de otros lenguajes que tienen declaraciones e instrucciones, en Haskell solo existen las expresiones y no pueden variar.

Inferencia de tipos: Los tipos son inferidos y se unifican de manera bidireccional.

Concurrente: Por el manejo de los efectos, la programación es de este tipo.

Perezoso: Las funciones no evalúan los argumentos y los programas pueden componerse escribiendo funciones normales. Esto brinda pureza al código y la posibilidad de fusionar cadenas de función.

Los programas escritos en Haskell se representan como funciones matemáticas. Cada función devuelve un mismo resultado, por lo cual no habrá cambios en el programa. El valor de una expresión dependerá de los parámetros que se le indican.

Entre las nociones más utilizadas se encuentran:

Tipos: Son usados para detectar los posibles errores en las expresiones y definiciones. Los valores se organizan en tipos, y cada uno tiene una serie de operaciones que no es entendible para otros tipos (son únicas). Es posible asociar un tipo único a toda una expresión.
Existen varios tipos predefinidos:
Básicos: son los valores primitivos (enteros, flotantes o booleano) y **Compuestos**, que se construyen usando otros tipos (listas, funciones y tuplas).
Funciones: Están definidas por un grupo de ecuaciones o declaraciones. Son objetos de primera clase y pueden ser argumentos, resultados de otras funciones o componentes de estructuras. Permiten simular en una sola función, varias funciones con muchos argumentos.
Entorno HUGS: También llamado entorno Haskell, sigue el modelo de las calculadoras y se establece una sesión entre el equipo y el usuario.
Listas: representa elementos que tienen un valor en concreto. Estas se construyen de dos formas: vacía representada por [] y no vacía que se construyen, de forma explícita, a través de sus elementos.

Muchos de los programadores debaten si vale la pena o no aprender Haskell. Mientras unos argumentan que no, otros sugieren aprenderlo para conocer uno de los lenguajes puramente funcionales. Actualmente, se mantiene como uno de los lenguajes menos empleados según datos estadísticos

Al diseñar el lenguaje se observó que no existía un tratamiento sistemático de la sobrecarga con lo cual se construyó una nueva solución conocida como las clases de tipos.

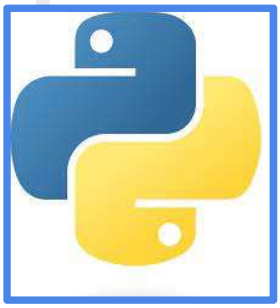
El lenguaje incorporaba, además, Entrada/Salida puramente funcional y, como ya se mencionó más arriba, definición de arrays por comprensión. Durante casi 10 años aparecieron varias versiones del lenguaje Haskell, hasta que en 1998 se decidió proporcionar una versión estable del lenguaje, que se denominó Haskell 98, a la vez

que se continuaba la investigación de nuevas características y nuevas extensiones al lenguaje.

A lo largo de la vida del lenguaje, como resultado de su investigación han sido implementadas algunas extensiones en algunos sistemas Haskell; por ejemplo el diseño de guardas, clases de tipos con multiparámetros, cuantificaciones locales y universales, entre otras. En la actualidad Haskell 2010 es el estándar oficial del lenguaje.

BNF HASKELL

https://www.haskell.org/onlinereport/syntax-iso.html
https://www.haskell.org/onlinereport/haskell2010/haskellch10.html



Python: Historia

La historia del lenguaje de programación Python se remonta hacia finales de los 80s y principio de los 90s, su implementación comenzó en diciembre de 1989 cuando en Navidad, Guido Van Rossum que trabajaba en el CWI (un centro de investigación holandés de carácter oficial que, entre otras cosas, actualmente alberga la oficina central del W3C) decidió empezar el proyecto como un pasatiempo dándole continuidad al lenguaje de programación ABC del que había formado parte del equipo de desarrollo en el CWI. En las imágenes se puede ver una comparativa entre un código en ABC y en Python.

Recuperar las palabras de un documento en ABC

```
HOW TO RETURN words document:
    PUT {} IN collection
    FOR line IN document:
        FOR word IN split line:
            IF word not.in collection:
                INSERT word IN
collection
    RETURN collection
```

Recuperar las palabras de un documento en Python

```
def words(document):
    collection = set()
    for line in document:
        for word in line.split():
            if word not in collection:
                collection.add(word)
    return collection
```

Grupo 25 SYS 2023

ABC fue desarrollado a principios de los 80s como alternativa a BASIC, fue pensado para principiantes por su facilidad de aprendizaje y uso. Su código era compacto pero legible. El proyecto no trascendió ya que el hardware disponible en la época hacía difícil su uso. Sin embargo, Van Rossum le dió una segunda vida creando Python. El lenguaje de programación Python fue originalmente desarrollado para el sistema operativo Amoeba.

Van Rossum es por tanto el autor principal de Python y continúa ejerciendo un rol central decidiendo la dirección del lenguaje. El nombre "Python" viene dado por la afición de Van Rossum al grupo humorístico Monty Python.

En 1991, Van Rossum publicó el código de la versión 0.9.0 en alt.sources. En esta versión ya hay disponibles clases con herencias, manejo de excepciones, funciones y los tipos modulares. Para este mismo año, Python llega a la versión 1.0 que incluyó herramientas de la programación funcional como lambda, reduce, filter y map.

Para Python 2.0 se incluyó la generación de listas, una de las características más importantes del lenguaje de programación funcional Haskell. Además incluyó un sistema de recolección de basura, que administra de forma automática la memoria, ya que es el encargado de liberar los objetos que ya no están en uso y que no serán usados en el futuro.

La última gran actualización de Python fue en 2008 con Python 3.0, diseñado para rectificar fallas fundamentales en el diseño del lenguaje. La filosofía de Python 3.0 es la misma de las versiones anteriores, sin embargo, Python como lenguaje ha acumulado nuevas y redundantes formas de programar una misma tarea.

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo, entre ellas: Instagram, Netflix, Spotify. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. **Es un lenguaje interpretado, dinámico y multiplataforma**.

Fuentes

Historia Python: https://platzi.com/blog/historia-python/

Grupo 25 SYS 2023

Historia Haskell: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-haskell/#:~:text=Haskell%20es%20un%20lenguaje%20de,(entre%201920%20y%201960).

https://labsys.frc.utn.edu.ar/ppr-2011/Unidad%20IV%20-%20Paradigma%20funcional/Unidad%20V%20-%20Paradigma%20Funcional.pdf

BNF PYTHON

https://pypi.org/project/bnf/

https://docs.python.org/3/reference/grammar.html

Grupo 25 SYS 2023

Rendimiento

● *Introducción*

Al momento de realizar el benchmark nos decidimos por realizar dos algoritmos que cumplan distintas tareas en un programa. El primero es Fibonacci que requiere tanto realizar cálculos como manejo de memoria dinámica al momento de trazar la lista resultante sin un límite específico. El otro algoritmo es el de ordenamiento por burbujeo (BubbleSort) que es crítico en cuanto al manejo de orden de una lista y de variables.

Luego, para medir el tiempo y memoria, en cuanto a Haskell ya disponemos de un comando que nos ofrece la consola: “ghci> :set +s”.

http://downloads.haskell.org/~ghc/7.2.1/docs/html/users_guide/ghci-set.html

En cuanto a python tuvimos que utilizar librerías que colocamos en el main acotando la función a analizar:

- **time:** para el tiempo de ejecución. https://docs.python.org/3/library/time.html
- **tracemalloc:** medir memoria durante la ejecución.

https://docs.python.org/es/3/library/tracemalloc.html

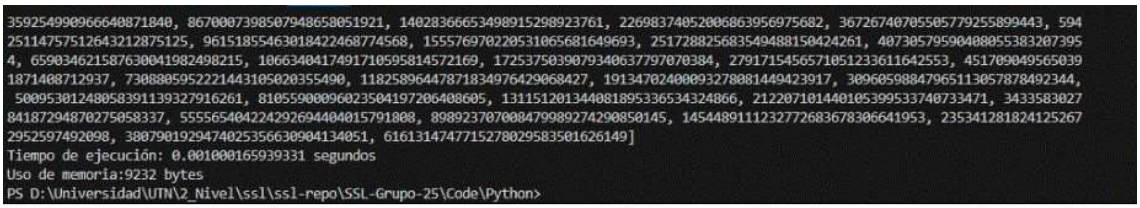
● *Fibonacci*

Obtenemos los primeros 150 números de la sucesión de Fibonacci.

Haskell

```
ghci> fibonacci 150
[0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,121393,196418,317811,514229,832040,1346269,2178309,
3524578,5702887,9227465,14930352,24157817,39088169,63245986,102334155,165580141,267914296,433494437,701408733,1134903170,1836311903,2971215073,48075269
76,7778742049,12586269025,20365011074,32951280099,53316291173,86267571272,139583862445,225851433717,365435296162,591286729879,956722026041,154800875592
0,2504730781961,4052739537881,6557470319842,10610209857723,17167680177565,27777890035288,44945570212853,72723460248141,117669030460994,190392490709135,
308061521170129,498454011879264,806515533049393,1304969544928657,2111485077978050,3416454622906707,5527939700884757,8944394323791464,14472334024676221,
23416728348467685,37889062373143906,61305790721611591,99194853094755497,160500643816367088,259695496911122585,420196140727489673,679891637638612258,110
0087778366101931,1779979416004714189,2880067194370816120,4660046610375530309,7540113804746346429,12200160415121876738,19740274219868223167,319404346349
90099905,516807088548583230 72,836211434898484229 77,1353018523447067 46049,2189229958345551690 26,3542248481792619150 75,5731478440138170841 01,927372692193
078999176,1500520536206896083277,2427893228399975082453,3928413764606871165730,6356306993006846248183,10284720757613717413913,16641027750620563662096,2
6925748508234281076009,4356677625885484473 8105,7049252476708912581 4114,1140593010259439705522 19,1845518257930330963663 33,2986111268189770669185 52,48316
2952612010163284885,7817740794309872302034 37,1264937032042997393488322,2046711111473984623691759,3311648143516982017180081,5358359254990966640871840,86
7000739850794865805 1921,1402836665349891529892376 1,2269837405200686395697 5682,3672674070550577925899443,5942511475751264321287 5125,9615185546301842246
8774568,1555769702205310656816496 93,2517288256835494881504242 61,4073057959040805538320 3954,6590346215876300419824 98215,1066340417491710595814572169,17
2537503907934063779707038 4,2791715456571051233611642553,4517090495650391871408712937,7308805952221443105020355490,1182589644787183497642906 8427,1913470
2400093278081449423917,3096059884796511305787 84932344,5009530124805839113932791 6261,8105590009602350419720 6408605,1311512013440818953365343 24866,2122071
0144010539953374073 3471,3433583027841872948702750583 37,5555654042242926944040157918 88,8989237070084799892742908501 45,1454489111232772683678306641953,23
5341281824125267295259749 2098,3807901929474025356630904134051,6161314747715278029583501626149]
(0.06 secs, 3,785,160 bytes)
```

Python



Comparativa

Característica / Lenguaje	Velocidad	Memoria Usada
Haskell	0.06 seg	3785160 bytes
Python	0.001 seg	9232 bytes

La **diferencia de memoria** es fácilmente adjudicable a la fuerte tipificación estática de Haskell, la cual no permite que una variable cambie de valor, sino que se crea y asigna, se crea y asigna, y ese valor no cambia. Por lo que para dar 150 números, debe utilizar espacio para esos 150 números, mientras que Python no (tipado dinámico). Otro factor es la evaluación perezosa de Haskell, que si bien puede tener un beneficio de rendimiento en algunos casos, puede generar una sobrecarga en términos de uso de memoria.
Además, Python utiliza un sistema de administración de memoria que incluye un recolector de basura para liberar automáticamente la memoria no utilizada.

.

La **diferencia de velocidad** en ejecución, se puede atribuir a varios factores:

- Gestión de memoria: La gestión de memoria también puede influir en el rendimiento. Haskell y Python tienen enfoques diferentes para la gestión de la memoria y la recolección de basura.
- Evaluación diferida (Haskell) vs ansiosa (Python): La evaluación diferida en Haskell puede hacer que algunos cálculos sean diferidos hasta que sean necesarios. Esto puede tener un impacto positivo en el rendimiento en ciertos programas, pero también puede tener implicaciones de rendimiento en otros.

Grupo 25 SYS 2023

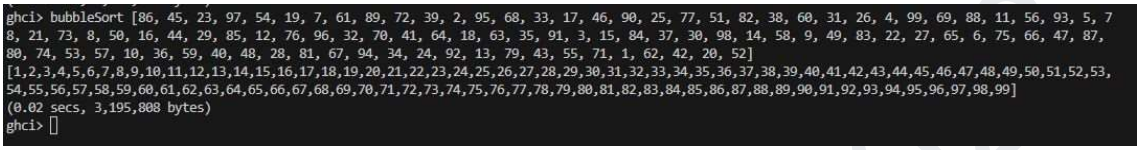
No es necesariamente cierto que Haskell sea más lento en la ejecución que Python en todos los casos, incluyendo el cálculo de los primeros N números de la sucesión de Fibonacci. La velocidad de ejecución de un programa en un lenguaje de programación está influenciada por varios factores, como la eficiencia de la implementación, las optimizaciones del compilador, la gestión de memoria y la elección de algoritmos y estructuras de datos.

Grupo 25 SYS 2023

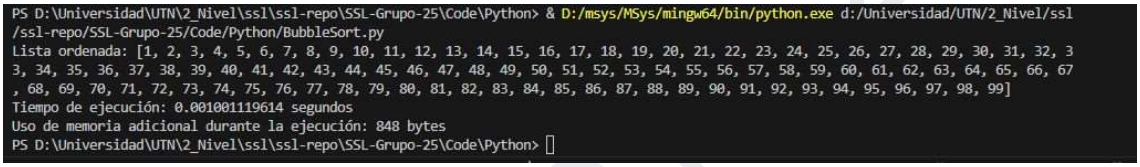
● *BubbleSort (Ordenamiento de burbuja)*

Dada una lista de números del 1 al 99, se debe ordenar esta lista mediante la técnica del “burbujeo”.

Haskell



Python



Comparativa

Característica / Lenguaje	Velocidad	Memoria Usada
Haskell	0.02 seg	3195808 bytes
Python	0.001 seg	848 bytes

En este caso, la **diferencia de velocidad** en ejecución es menor, y se lo podría atribuir a factores ajenos al lenguaje en sí, como la máquina en la que se testea, la forma en que se manejan las estructuras de datos en ambos lenguajes, optimización de ciertos compiladores o las diferencias mínimas en el código.

Sin embargo, la **diferencia de memoria** vuelve a ser considerable, pero las razones siguen siendo las mismas:

Tipado estático vs tipado dinámico: Haskell es un lenguaje de tipado estático, lo que significa que el tipo de datos de las variables se verifica en tiempo de compilación.

Esto puede llevar a una representación más robusta y precisa de los datos, pero también puede requerir más memoria para almacenar información sobre tipos.

Evaluación perezosa en Haskell: Haskell utiliza evaluación perezosa (lazy evaluation), lo que puede afectar la gestión de memoria en algunos casos. Las estructuras de datos en Haskell pueden ser representadas de manera más eficiente gracias a la evaluación perezosa, pero en ciertos algoritmos podría resultar en el almacenamiento temporal de más datos en memoria antes de que se resuelvan.

Eficiencia de implementación: La implementación del algoritmo de ordenamiento por burbujeo en Haskell podría ser menos eficiente en términos de memoria que una implementación en Python. Las optimizaciones de compilación y el uso de estructuras de datos pueden influir en la cantidad de memoria utilizada.

CONCLUSIÓN

En general, Haskell y Python son lenguajes con características diferentes, y pertenecientes a diferentes paradigmas de la programación (diferentes objetivos), que pueden afectar su rendimiento en diferentes escenarios.

Grupo 25 SYS 2023

Enlaces usados

https://labsys.frc.utn.edu.ar/ppr-2011/Unidad%20IV%20-%20Paradigma%20funcional/Unidad%20V%20-%20Paradigma%20Funcional.pdf

https://docs.python.org/es/3/tutorial/

Grupo 25 SYS 2023