COMPARACIÓN ENTRE HASKELL Y PYTHON

GRUPO 25

SINTAXIS Y SEMÁNTICA DE LOS LENGUAJES

K2055

HASKELL VS PYTHON

Lenguaje de programación puramente FUNCIONAL

Año de lanzamiento: 1990

Basado en: cálculo lambda, funciones.

Objetivo: unificar las características más importantes de los lenguajes funcionales.

Nombrado en honor a Haskell Curry.

Lenguaje de alto nivel de programación con gran legibilidad de su código

Año de lanzamiento: 1989

Lenguaje interpretado, dinámico y multiplataforma.

Objetivo: lenguaje sencillo y compacto.

Creador: Guido Van Rossum, Países Bajos.

CARACTERISTICAS

Haskell

- Funciones de orden superior
- Evaluación perezosa
- Inferencia estática de tipos
- Tipos de datos definidos por el usuario
- Encaje de patrones
- Listas(arrays) por comprensión
- Clases de tipos: tratamiento sistemático de la sobrecarga
- Entrada/Salida puramente funcional
- Diseño de guardas

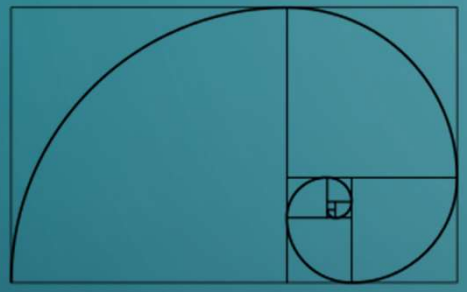
Python

- Clases con herencias
- Manejo de excepciones
- Funciones y los tipos modulares
- Generación de listas
- Sistema de recolección de basura
- Multiparadigma

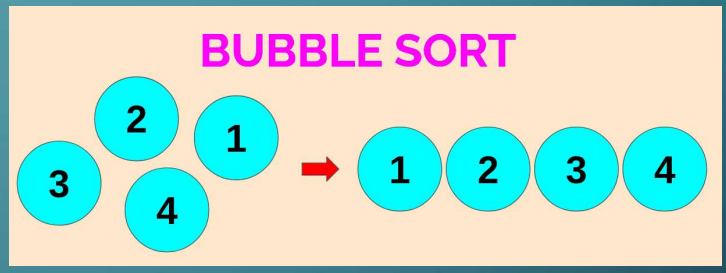
¿EN QUE SECTORES SE UTILIZAN?

ALGORITMOS ELEGIDOS

Suc. de Fibonacci: Manejo de memoria + cálculos



BubbleSort (Ordenamiento de una lista): Manejo de memoria



Medición de Tiempo y memoria:

- ghci> :set +s (HASKELL)
- import time (tiempo) (PYTHON)
- import tracemalloc (memoria) (PYTHON)

BENCHMARK FIBONACCI

Obtuvimos los primeros 150 números de la sucesión de Fibonacci.

HASKELL

```haskell
{-Función fibonacci-}
fibonacci :: Number -> [Number]
fibonacci n
    --Casos base
    | n <= 0 = []
    | n == 1 = [0]
    | n == 2 = [0, 1]
    --Caso recursivo
    | otherwise = fibonacci' [0, 1] (n - 2)
    where fibonacci' cola conteo
            | conteo <= 0 = cola --Corta recursividad y devuelve la lista como está
            | otherwise = fibonacci' (cola ++ [sum $ drop (length cola - 2) cola]) (conteo - 1)
            --Si se puede seguir: calcula el siguiente número, actualiza la lista con el nuevo
            --número calculado y llama a la función de vuelta con el conteo y lista actualizada
```

PYTHON

```python
#Algoritmo que realiza la función de fibonacci de un número n.
def fibonacci(n):
    #Casos base
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]
    else:
        #Caso recursivo
        secuencia_fibonacci = [0, 1]
        for i in range(2, n):
            #se agrega a la lista el cálculo del número siguiente de la secuencia
            secuencia_fibonacci.append(secuencia_fibonacci[i - 1] + secuencia_fibonacci[i - 2])
        return secuencia_fibonacci
```

Comparativa

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

BENCHMARK BUBBLESORT

Dada una lista de números del 1 al 99, se ordena esta lista mediante la técnica del "burbujeo".

HASKELL

```haskell
{-Función bubbleSort-}
bubbleSort :: Ord a => [a] -> [a]
--Caso base
bubbleSort [] = []
--Caso recursivo      You, 23 hours ago • Comentarios añadidos
bubbleSort cola = bubbleSort (init ordenado) ++ [last ordenado]
    --Aplicamos la función bubble a la cola mediante foldr
    where ordenado = foldr bubble [] cola
    --Definiciones de la función bubble
          bubble x [] = [x] --Si está vacía agrega directamente
          bubble x (y:ys) --Dado un elemento a incrustar
              | x <= y = x:y:ys --si es menor va primero
              | otherwise = y : bubble x ys --Si va luego, se pone detrás y se vuelve a llamar a la función
```

PYTHON

```python
#Función de bubble_sort, que lo que hace es ordenar una lista de n elementos por orden numérico
def bubble_sort(lista):
    #n es el largo de la lista
    n = len(lista)
    for i in range(n):
        #lo hacemos tantas veces como lugares en la lista - 1      You, 22 minutes ago • Uncommi
        for j in range(0, n - i - 1):
            #Si el número anterior es mayor al siguiente, intercambiamos el orden
            if lista[j] > lista[j + 1]:
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
```

Comparativa

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

CONCLUSIÓN COMPARATIVA

PREGUNTAS

1. ¿Cuáles son las principales características de Haskell?
2. ¿Qué función tiene el llamado “recolector de basura” en Python?
3. ¿Por qué Haskell se limita a sus funciones mientras que Python utiliza variables globales y main?
4. ¿Cuáles son las razones por las que Python aventaja a Haskell en velocidad de ejecución al correr un programa que obtenga los primeros 150 números de Fibonacci?
5. ¿A qué se le adjudica la diferencia que existe entre la memoria que utiliza cada lenguaje en ambos programas?