Desarrollo de un intérprete para Python adaptado a Niveles de Aprendizaje

Resumen

Motivación o Justificación

¿Porqué considera importante el desarrollo del proyecto, o crear esta herramienta?

**Niveles**

Justificar el orden de los niveles, y dar una descripción en cada nivel. Puede tomar como referencia el paper: http://www.dfists.ua.es/~gil/niveles.pdf

1. **Sintaxis básica parte I**

Este nivel representa los primeros pasos del usuario en manejar archivos, o incluso, en usar Python como lenguaje de programación. Es por esto que este nivel servirá para introducirlo al lenguaje y darle nociones básicas sobre archivos. Al escribir una simple línea como:

f = open(“archivo.txt”)

y presionar la Tecla intro, el usuario podrá captar el sentido de asignación de variables, lo que signifca la palabra open, la cual es intuitiva pues está “abriendo un archivo”, y podrá relacionarlo con el uso de variables en ciencias exactas, lo cual ya ha experimentado hasta este punto el usuario.

1. **Sintaxis básica parte II**

Aquí, adentraremos un poco al usuario a lo que implica el manejo de archivos en Python, y en general en cualquier otro lenguajes, lo cual es el permiso que se tiene sobre un archivo. El usuario aprenderá la notación usada para cada uno de los permisos existentes, y lo que hace. En este punto, ya podrá abrir un archivo para cualquier propósito, y como prueba para asegurarnos que entendió lo leído, se le pedirá que abra un archivo con permiso de escritura:

f = open(“prueba.txt”,”w”)

1. **Leyendo un archivo parte I**

La operación básica sobre un fichero es leer sus líneas. En Python, esto se realiza con un lazo for. Hasta este punto ,el usuario puede o no estar familiarizado con esta estructura iterativa, por lo que el usarla para leer todas las líneas de un archivo puede serle intuitive y darle la noción básica de lo que hace:

for line in archivo:

print line

Estas son las líneas que el usuario deberá de ingresar al pedirle que escriba las líneas de código que permitan imprimir cada línea de un archivo almacenado en la variable archivo.

1. **Leyendo un archivo parte II**

Una lectura de un archivo requiere de cierto nivel de conocimiento en manipulación de cadenas (strings). Puede que el usuario aún no haya realizado operaciones sobre este tipo de dato, por lo que en este nivel introduciremos un par de funciones sencillas, y de las más usadas al procesar líneas de archivos, estas funciones son strip() y split(). Se le da al usuario una breve descripción de la teoría de estas funciones y cómo usarlas. Se ilustra un simple ejemplo en el que se guardan datos que se encuentran separados por comas en un archivo. Luego, se le pide al usuario escribir un código que logre algo similar al ejemplo mostrado anteriormente:

f=open("archivo.txt")

for line in f:

sinespacios=line.strip()

datos=line.split("|")

Pero ahora, los datos están separados por el caracter “|”. Con este ejercicio, el usuario sabra que es un uso común el almacenar datos en archivos, separados por un caracter a decision del programador.

1. Escribiendo en un archivo
2. Dame un número y te diré qué línea es
3. Renombrando archivos
4. Buffers
5. Un mini programa
6. Buscando en otros archivos

Analizador Léxico

Mencionar herramientas exploradas y la seleccionada

Analizador Sintáctico

Mencionar herramientas exploradas y la seleccionada

Analizador Semántico

Mencionar herramientas exploradas y la seleccionada

Resultados

Captura de alguna simulación, nivel superado, error léxico o sintáctico.

Conclusiones y recomendaciones

Trabajo Futuro

¿Qué se podría mejorar?

¿Se podría mejorar el intérprete de Python actual?

Referencias

Nathan Sobo(2015).Treetop: Bringing the simplicity of ruby to syntactic analysis. Retrieved from <http://treetop.rubyforge.org/>.

Treetop documentation: <http://treetop.rubyforge.org/semantic_interpretation.html>