FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACIÓN

LENGUAJES DE PROGRAMACIÓN

PROYECTO DE SEGUNDO PARCIAL: PYTHON

ANALIZADOR SINTÁCTICO DE UN ARCHIVO XML

Integrantes:
Leonel Ramírez Gonzalez
José Vélez Gómez
Kevin Campuzano Castillo

Índice general

1. Objetivos **2**

2. Introducción **3**

3. Alcance del Proyecto **5**

4. Observaciones **8**

Capítulo 1

Objetivos

- Comprender, entender e implementar la sintaxis del Lenguaje de programacion Python.
- Ventajas y Desventajas del lenguaje de programacion Python referente a otros lenguajes.
- Implementar clases en Python.
- Sacar ventaja de este lenguaje que es multiparadigma.

Capítulo 2

Introducción

Python es un lenguaje de programación creado por Guido van Rossum a finales de los ochenta, y que gracias a sus características ha llegado a ser un lenguaje muy conocido en la actualidad. Python es un lenguaje muy simple, por lo que es muy fácil iniciarse en este lenguaje. El pseudo-código natural de Python es una de sus grandes fortalezas.

Usando el lenguaje Python se puede crear todo tipo de programas; programas de propósito general y también se pueden desarrollar páginas Web.

Python contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.
Las librerías pueden ayudar a hacer varias cosas como expresiones regulares, generación de documentos, evaluación de unidades, pruebas, procesos, bases de datos, navegadores web, CGI, ftp, correo electrónico, XML, XML-RPC, HTML, archivos WAV, criptografía, GUI, y también otras funciones dependientes del Sistema.

Python tiene una sintaxis muy visual, gracias a que maneja una sintaxis identada (con márgenes), que es de caracter obligatorio. Para separar los bloques de código en Python se debe tabular hacia dentro. Esto ayuda a que todos los programadores adopten las mismas notaciones y que los programas hechos en Python tengan un aspecto muy similar.

Capítulo 3

Alcance del Proyecto

Se investigo la manera de hacer parseo que es basicamente analizar una estructura de simbolos con el objetivo de terminar su estructura gramatica, como el caso de un archivo XML, generalmente un parseo, primero identifica los simbolos de entrada y luego lo va transformando a una estructura mas fácil de entender generalmente un arbol pero en nuestro caso se transforma en una lista de estructura de datos.

Para armar la estructura en Python, se investigo desde la pagina fuente, que es la pagina de la organización Python en la que nos demuestra que las estructuras son clases y se las implementa de la siguiente manera con sus respectivas funciones getter y setter.

Estructura de la clase Device

```
    def __init__(self, idD, user_agent, fall_back):
        self.idD = idD
        self.user_agent = user_agent
        self.fall_back = fall_back

    def setDevice(self, lista):
        idD, ug, fb, *listD = lista
        self.idD = idD
        self.user_agent = ug
        self.fall_back = fb
    def getIdDevice(self):
```

```
        return self.idD
    def setIdDevice(self, idD):
        self.idD = idD
    def getUserAgentDevice(self):
        return self.user_agent
    def setUserAgentDevice(self, user_agent):
        self.user_agent = user_agent
    def getFallBackDevice(self):
        return self.fall_back
    def setFallBackDevice(self, fall_back):
        self.fall_back = fall_back
```

Estructura de la clase Group

```
    def __init__(self, idG):
        self.idG = idG

    def setGroup(self, lista):
        idG, *list = lista
        self.idG = idG
    def getIdGroup(self):
        return self.idG
    def setIdGroup(self, idG):
        self.idG = idG
```

Estructura de la clase Capability

```
    def __init__(self, name, value):
        self.name = name
        self.value = value

    def setCapability(self, listaC):
        name, value,*list = listaC
        self.name = name
        self.value = value
    def getNameCapability(self):
        return self.name
    def setNameCapability(self, name):
        self.name = name
    def getValueCapability(self):
        return self.value
    def setValueCapability(self, value):
        self.value = value
```

Esto graficamente se representa como se lo muestra en la figura 1.1, en la cual nosotros decidimos separarlos cada uno por su estructura,en esta parte se tuvo problemas con la decision de como serian las estructuras de los grupos por ciertas correcciones que se hicieron con este proyecto en base a nuevas especificaciones de nuestro profesor. Se lo hizo de esta manera por que no vimos la necesidad de que estuviera asi ya que no lo almacenariamos en un archivo sino que solamente se lo procesaria en el mismo momento en que se forman su estructura.

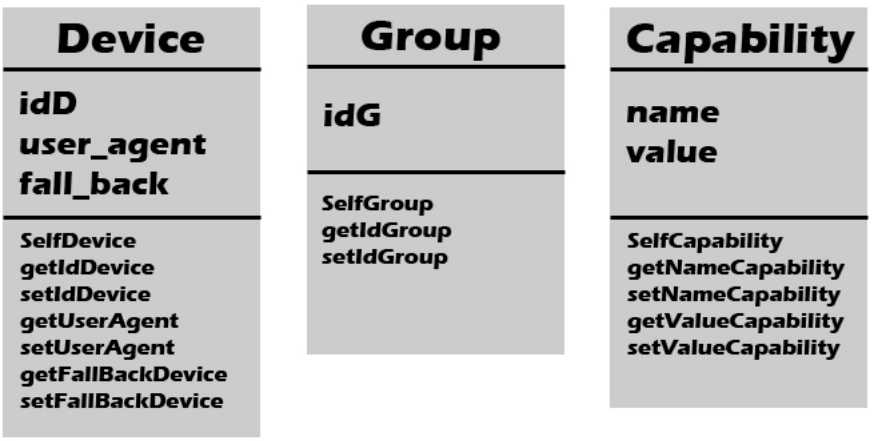

Figura 3.1: Estructura de datos

Al archivo Xml se le hizo algunos pasos para poder llegar al punto que deseamos. Traducimos las funciones que creamos en el programa de Haskell al lenguaje de programación Python, por una parte se hizo mas sencillo ya que Python es un lenguaje sencillo, limpio tuvimos que recurrir a otros metodos ya que ciertas funciones no se encontraban en Python.

Capítulo 4

Observaciones

- **Leonel Ramírez Gonzalez**

Ventajas: fácil de implementar variables, no se necesita declarar el tipo de variable. python se encarga de eso.

Desventajas: La desventaja que encontre es que no se puede usar recursividad con archivos demasiados grandes por que python utiliza colas y estas se pueden llenar a su limite por lo tanto votara un error.

- **José Vélez Gómez**

Ventajas: Existe gran cantidad de funciones y librerías. Sencillo y rápido de programar.

Desventajas: Oculta el uso de memoria.Los programas interpretados son mas lentos que los compilados.Sin embargo los programas interpretados suelen ser cortos, en los que la diferencia es inapreciable.

- **Kevin Campuzano Castillo**

Ventajas: Python es un lenguaje limpio para programar. Permite la creación de todo tipo de programas incluyendo sitios web.

Desventajas: lentitud por ser un lenguaje interpretado.