



**Proyecto de Lenguajes de Programacion.**

Nombre de Grupo: Python 1

**Integrantes del Grupo**

* Henry Steven Siavichay Plúas
  + - * Cristhian Andrés López Fuertes

**Profesor**: Rodrigo Saraguro.

1. Introducción.

Python es un lenguaje de programación multiparadigma, de mucho agrado para los programadores gracias a su versatilidad y facilidad de entendimiento de código, siendo este último factor con el cual ha ganado popularidad [2]. Este lenguaje ofrece una gran variedad de librería que facilitan el diseño de programas, ya que posee una consistencia sobre los demás lenguajes [1].

Python es uno de los pocos lenguajes que permite crear aplicaciones de servidor o incluso páginas web. Además de ser un lenguaje interpretado, es decir, que no necesita ser compilado su código para ejecutarlo [3]. Generalmente se lo utiliza para back-end cuando de páginas web se habla, junto con el Framework Django que es simple de entender y manejar el back-end [4].

Existen aplicaciones desarrolladas por Python que tienen un gran nivel Gimp es una aplicación similar Adobe Photoshop, otra aplicación es Blender que sirve para el modelado en 3D demostrando que Python maneja muy bien los recursos [].

1. Objetivos.

Desarrollar un analizador léxico y sintáctico para el lenguaje de programación Python, con la finalidad de aprender cómo se maneja la parte léxica y sintáctica de dicho lenguaje.

Objetivos Específicos.

* Aprender el uso de la librería PLY
* Aprender diferentes formas de escribir código en Python.
* Definir una gramática para las estructuras principales del lenguaje.
* Definir los tokens que maneja el lenguaje Python.

1. Documentación.

Alcance.

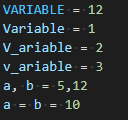
Nuestro proyecto analizará la forma de la declaración de la variable y estructuras de control, además se manejará de una manera adecuada el vocabulario de la sintaxis del lenguaje de programación en este caso el lenguaje Python.

* 1. **Definir la escritura correcta de variables y su inicialización.**

Las variables pueden ser escritas con cualquier carácter alfanumérico, y el carácter especial “\_”. Las variables tienen que empezar con un carácter alfabético o el carácter especial “\_”.

Y respecto a su inicialización debe estar precedida de su dato, no se aceptan variables sin su valor correspondiente.

Ejemplo:



* 1. **Tipos de datos primitivos: enteros, flotantes, booleanos**

|  |  |
| --- | --- |
| Tipo de Dato | Ejemplo |
| Enteros | Variable = 1 |
| Flotantes | Variable = 1.2 |
| Booleanos | Variable = False ó True |

* 1. **Tipos de datos estructurados: str, listas, tuplas, conjuntos, etc.**

|  |  |
| --- | --- |
| Tipo de Dato Estructurado | Ejemplo |
| str(String) | 1. Cadena = “string” 2. Cadena = ‘string’ |
| Lista | 1. Lista = [ 1, 2, 3 ] 2. lista = [“A”,”B”,”C”] 3. lista = [2.1 , 3.5, 2.3] |
| Tupla | 1. tupla = (2,3) 2. tupla = (2,5,3,…,) |
| Conjunto | 1. Conjunto = {1, 2, 3} 2. conjunto = {“A”, ”B”, ”C”} 3. conjunto = {2.1 , 3.5, 2.3} |
| Diccionario | Diccionario = { “Clave1” : “Valor1”, “Clave2” : “Valor2”} |

* 1. **Definir las operaciones matemáticas y condiciones de acuerdo a los tipos de datos mencionados.**

|  |  |
| --- | --- |
| Operación | Ejemplo |
| Adición. | a = 4 + 5 |
| Sustracción. | d = 4 – 5 |
| Multiplicación. | c = 4 \* 5 |
| División. | b = 4 / 5  b = 4//5 división con resultado entero |
| Exponencial. | e = 4 \*\* 5 |
| Mod | f = 4 % 5 |

* 1. **Métodos impresión, lectura de datos.**

**3.5.1 Imprimir**

Se utiliza la función “print()”, la cual recibe cualquier objeto y lo convierte en “string”, y muestra por pantalla.



=> Hello World

* + 1. **Lectura de datos.**

1. **Inputs del usuario.**

Se utiliza la función “input()”, la cual recibe input del usuario y lo guarda en un “string”. Esta función puede recibir como parámetro un mensaje que se le mostrara al usuario.



1. **Archivos.**

Para leer archivos, se tiene que almacenar el archivo con

f = open(“nombrearchivo”, “r”)



* 1. **Definir 3 estructuras de control básicas.**
     1. **Estructura “IF”**



En condición puede ir cualquier dato o función que pueda devolver un tipo de dato booleano.

Y en Orden puede ir cualquier cosa incluso otra condición o otra estructura.

Ejemplo:



* + 1. **Estructura “WHILE”**



En condición puede ir cualquier dato o función que pueda devolver un tipo de dato booleano.

Y en Orden puede ir cualquier cosa incluso otra condición o otra estructura Ejemplo:



* + 1. **Estructura “FOR”**



Elemento es una variable.

Puede contener una lista o tipo de dato estructurado.

Sentencia puede existir otra estructura**.**

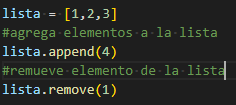
Ejemplo:



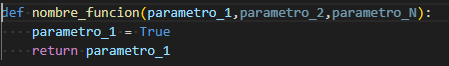


* + **Para datos estructurados definir al menos 2 métodos.**

Para listas.



* + **Creación de funciones.**



Nombre de la Función puede ser cualquiera siempre y cuando cumpla estándares de variables al igual que los parámetros.

Y la sentencia puede retornar un valor o solo ejecutar estructuras de control o imprimir.

Ejemplo:



**Operaciones Lógicas**

|  |  |
| --- | --- |
| Operación Lógica | Ejemplo |
| and(“y”) | Variable = True  Variable2= False  Variable and Variable2 |
| or(“o”) | Variable = True  Variable2= False  Variable or Variable2 |
| not(“no”) | Variable = False  not Variable |

OPERADORES DE COMPARACION

|  |  |
| --- | --- |
| Operación Lógica | Ejemplo |
| = =  (“igual a”) | Variable = 1  Variable2= 1  Variable = = Variable2 |
| ! =  (“no igual a”) | Variable = True  Variable2= False  Variable ! = Variable2 |
| > =  (“mayor igual que”) | Variable = 1  Variable2= 2  Variable >= Variable2 |
| < =  (“menor igual que”) | Variable = 1  Variable2= 2  Variable <= Variable2 |
| >  (“mayor que”) | **Variable = 1**  **Variable2= 2**  **Variable > Variable2** |
| < =  (“menor que”) | Variable = 1  Variable2= 2  Variable < Variable2 |

1. Resultados

**Código de prueba.**



**Salida del analizador**





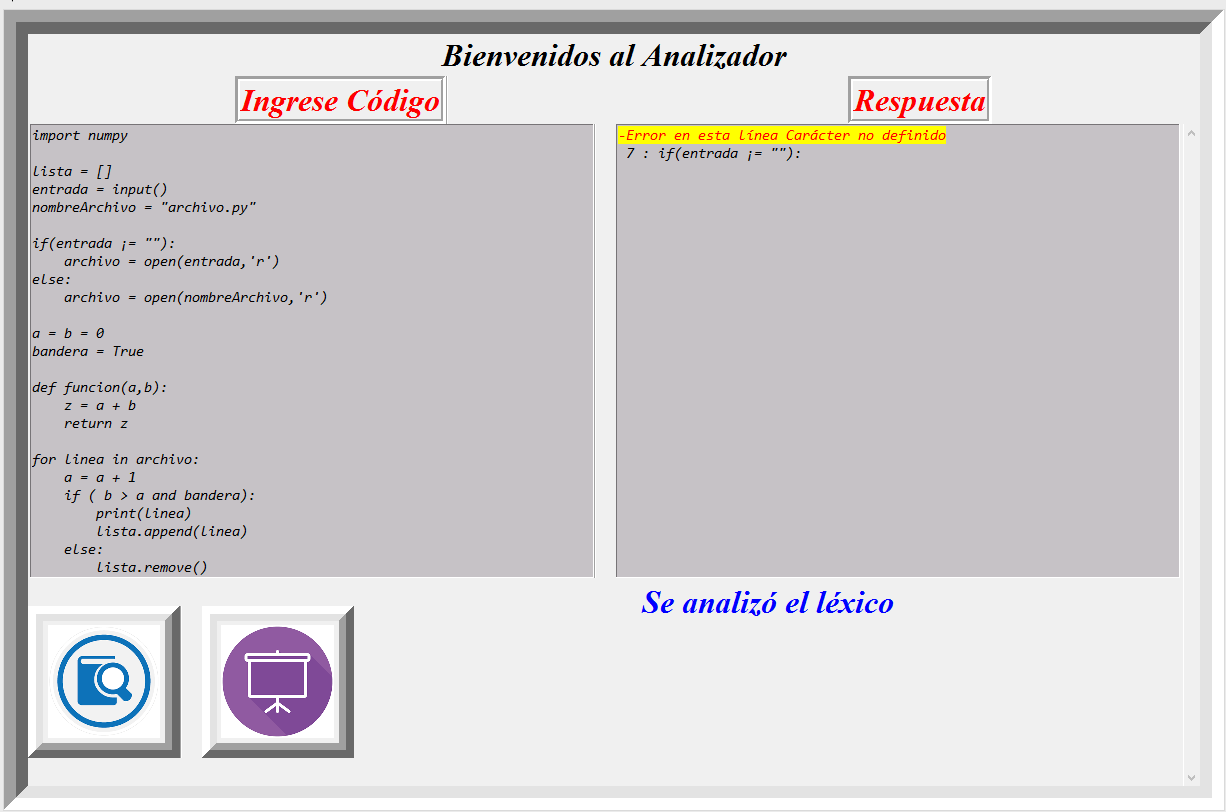
Cabe recalcar que se muestra un mensaje si recuadro de texto no tiene ningún carácter significa que está correcto.

**Se analiza con errores.**

Error Sintáctico.



Error Léxico.



1. Conclusiones.

Para concluir, hemos podido desarrollar una herramienta capaz de hacer análisis tanto léxico como sintáctico del código ingresado por el usuario. Nos pareció que Python es un buen lenguaje de emplear para poder aprender a utilizar los analizadores léxicos y sintácticos de la librería PLY, no solo porque es el primer lenguaje que aprendimos al entrar a la ESPOL, sino también gracias a su simplicidad y facilidad de construir sentencias, y sus reglas simples. El proyecto nos ayudo a poder analizar como el lenguaje funcionaba, y poder deducir las reglas de como servía este. Las herramientas a utilizar fueron muy simples, en parte gracias a la documentación existente, de tanto PLY, como Tkinter, y logramos desarrollar una buena aplicación de análisis gracias a esto.

1. Referencias.

[1] Challenger-Pérez, I., Díaz-Ricardo, Y., & Becerra-García, R. A. (2014). El lenguaje de programación Python. Ciencias Holguín, 20(2), 1-13.

[2] González Duque, R. (2014). Python para todos.

[3] Marzal, A., & Luengo, I. G. (2002). Introducción a la Programación con Python y C. Publicacions de la Universitat Jaume I.

[4] Ríos, J. R. M., Mora, N. M. L., Ordóñez, M. P. Z., & Sojos, E. L. L. (2016). Evaluación de los Frameworks en el Desarrollo de Aplicaciones Web con Python. Revista latinoamericana de Ingenieria de Software, 4(4), 201-207.