Universidad de Guadalajara
Centro universitario de Ciencias Exactas e Ingenierías

.

Nombre del alumnos:
Andre Maximiliano Calderón Acero -215661275
Diego Andres Hernandez Rodriguez -216028371
Juan Antonio Pérez Juárez - 215660996

Traductores de lenguajes 2
Práctica 1.

Actividad

Dado unas reglas de producción que sirven para construir una gramática en especifico, elaborar un analizador sintáctico descendente con la gramática vista en clase.

Reglas de producción de la gramática

Lista-> digito | Resto_lista
Resto_lista -> + digito
Resto_lista -> - digito
Resto_lista -> ∈
digito -> 0|1|2|3|4|5|6|7|8|9

Código:

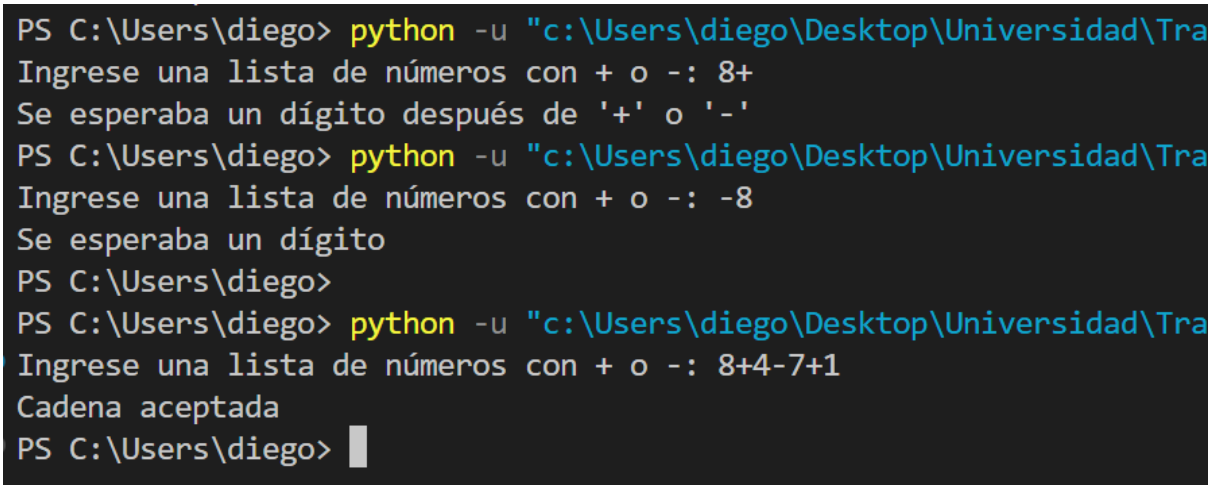
Python

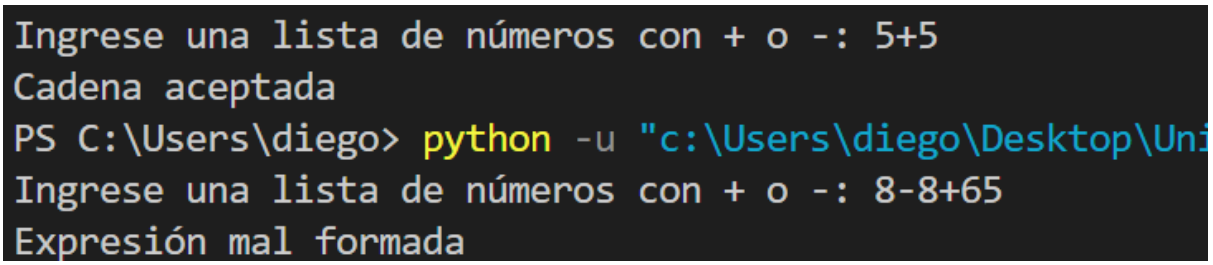
```
class Parser:
    def __init__(self, expression):
        self.expression = expression
        self.index = 0

    def parse(self):
        self.index = 0
        try:
            result = self.lista()
            if self.index == len(self.expression):
                return "Cadena aceptada"
            else:
                raise SyntaxError("Expresión mal formada")
        except SyntaxError as e:
            return str(e)

    def lista(self):
        if self.index < len(self.expression) and
self.expression[self.index].isdigit():
            self.index += 1  # Acepta un solo dígito
            self.resto_lista()
            return True
        else:
            raise SyntaxError("Se esperaba un dígito")

    def resto_lista(self):
        if self.index < len(self.expression) and self.expression[self.index]
in ('+', '-'):
```

```
            self.index += 1  # Avanza sobre '+' o '-'
            if self.index < len(self.expression) and
self.expression[self.index].isdigit():
                self.index += 1  # Acepta el dígito después del operador
                self.resto_lista()  # Llamada recursiva para más elementos
            else:
                raise SyntaxError("Se esperaba un dígito después de '+' o
'-'")
        # Si no hay '+' o '-', se permite la cadena vacía (epsilon)
        return True

# Ejemplo de uso
if __name__ == "__main__":
    expression = input("Ingrese una lista de números con + o -: ")
    parser = Parser(expression.replace(" ", ""))  # Elimina espacios en
blanco
    result = parser.parse()
    print(result)
```

O también puede acceder al repositorio de gitHub en el Siguiente link:
https://github.com/JuanAntonioPerezJuarez/TraductoresLenguajesII

Capturas de pantalla:





```
PS C:\Users\diego> python -u "c:\Users\diego\Desktop\Universidad\T
Ingrese una lista de números con + o -: (8+3)+5+4+8
Se esperaba un dígito
```

Conclusión:

Esta es una pequeña práctica que nos servirá para más adelante poder realizar más prácticas relacionadas con el reconocimiento de caracteres y cadenas relacionadas a los entornos de compiladores encargados de reconocer expresiones y convertirlas en sentencias de lenguajes de programación de alto nivel.

Aunque parezca algo sencillo, el poder reconocer si a una expresión le falta algo o está bien formulada es una tarea esencial por parte de los compiladores para poder hacer expresiones matemáticas en un orden correcto para no provocar ambigüedades.