Implementación de métodos computacionales

(Gpo 601)

Actividad Integradora 2 - Resaltador de sintaxis en programación funcional

Profesor: Jesús Guillermo Falcón Cardona

Alumnos:

Daniel Antonio Sánchez Cantú A00839072

Axel Patricio De Gyves Garcia A01352329

Fecha: 06/06/2025

Introducción

El analizador léxico, programado en *Racket*, para *Python* descompone el código por partes, clasificándolos en categorías como palabras clave (if, def), operadores (+, **=), números (123, 3.14E-2), strings ("hola") y delimitadores ((, :). Esto mediante un autómata finito determinista (DFA por sus siglas en inglés), sigue reglas como longest match para operadores y distingue contextos (ej: - como operador o parte de un número negativo), generando al final un archivo HTML+CSS después de analizar el código.

Diccionario de categorías léxicas, indicando todos los elementos que estén dentro de cada categoría.

1. **WHITESPACE (Ignorado en tokens finales)**

   Elementos:

   i. Espacio ( )
   ii. Tab (\t)
   iii. Nueva línea (\n)
   iv. Retorno de carro (\r)

2. **COMMENT**

   Elementos:

   i. Todo texto desde # hasta el final de la línea.
   Ejemplo: # Esto es una funcion

3. **STRING**

   Elementos:

   i. Cadenas entre comillas simples ('...') o dobles ("...").
   ii. Admiten escapes con \ (ej: \", \n).
   Ejemplo: "Python_Lexer_Test", 'cadena\n'

4. **KEYWORD (Palabras reservadas)**

   Elementos:

```python
"False", "None", "True", "and", "as", "assert",
"async", "await", "break",
"class", "continue", "def", "del", "elif",
"else", "except", "finally",
"for", "from", "global", "if", "import", "in",
"is", "lambda", "nonlocal",
"not", "or", "pass", "raise", "return", "try",
"while", "with", "yield"
```

5. **IDENTIFIER (Identificadores)**

   Elementos:

   i. Secuencias que empiezan con letra o _, seguidas de letras, dígitos o _.

   ii. No incluye keywords.

   Ejemplo: my_func, x, res, variable_1

6. **INTEGER (Enteros)**

   Elementos:

   i. Secuencias de dígitos (0-9).

   ii. Pueden tener signo negativo (-).

   Ejemplo: 100, -50, 0

7. **FLOAT (Flotantes)**

   a. Elementos:

      i. Formato 1: [dígitos].[dígitos][exponente?]

      Ejemplo: 2.0, 10.5, .5, .007

      ii. Formato 2: [dígitos][e|E][+|-]?[dígitos]

Ejemplo: 3.14E-2, 1e10, 23.

iii. Pueden tener signo negativo (-).

8. OPERATOR (Operadores)

Elementos (ordenados por longitud):

```python
# 3 caracteres:
"**=", "//=", ">>=", "<<="
# 2 caracteres:
"+=", "-=", "*=", "/=", "%=", "&=", "|=", "^=",
"**", "//", "==", "!=", "<=", ">=", "->"
# 1 carácter:
"+", "-", "*", "/", "%", "&", "|", "^", "~", "<",
">", "=", "@", ":"
```

9. DELIMITER (Delimitadores)

Elementos:

```python
";", "(", ")", "[", "]", "{", "}", ",", "."
```

Nota: El . solo es delimitador si no es parte de un float (ej: en a.b el . es delimitador).

10. ERROR (Caracteres no válidos)

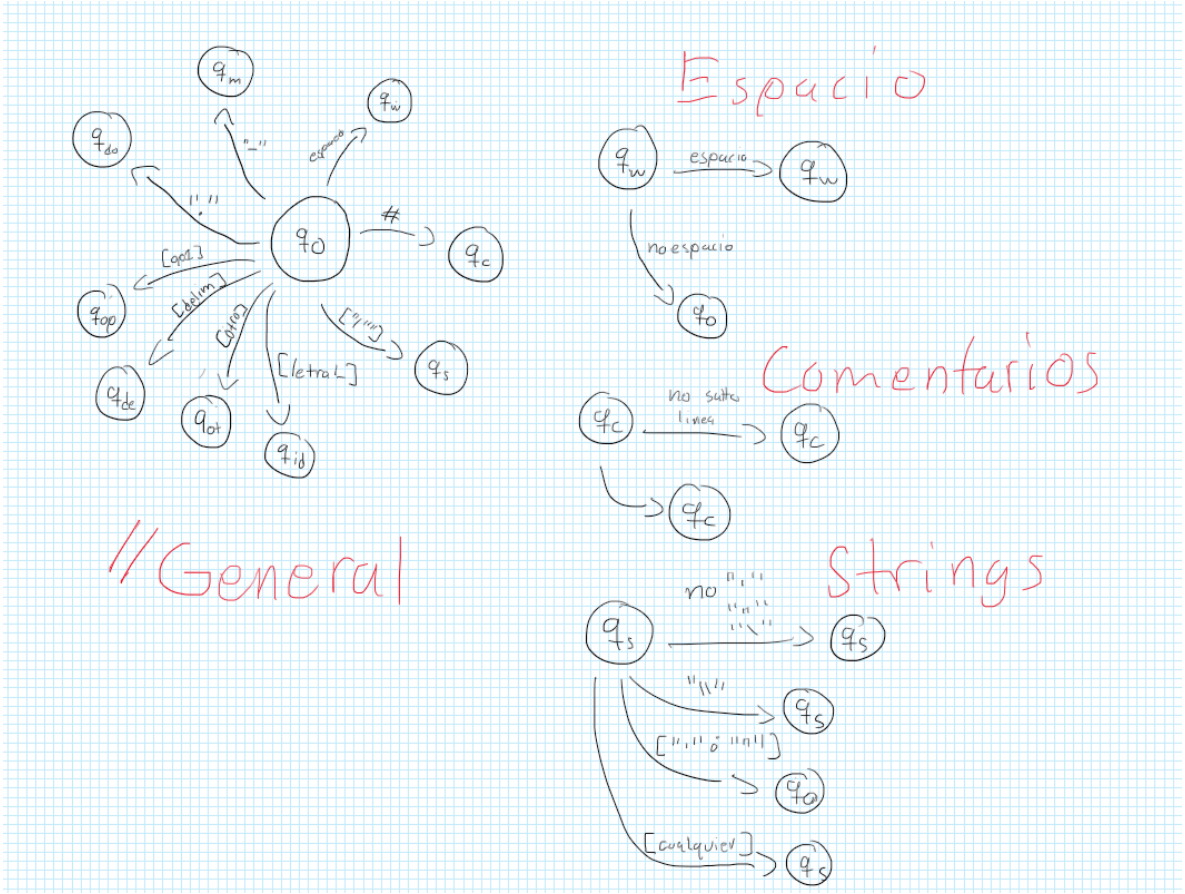
Elementos:

Cualquier carácter no reconocido por las categorías anteriores.
Ejemplo: $, ¿, ¡ en contexto inesperado.

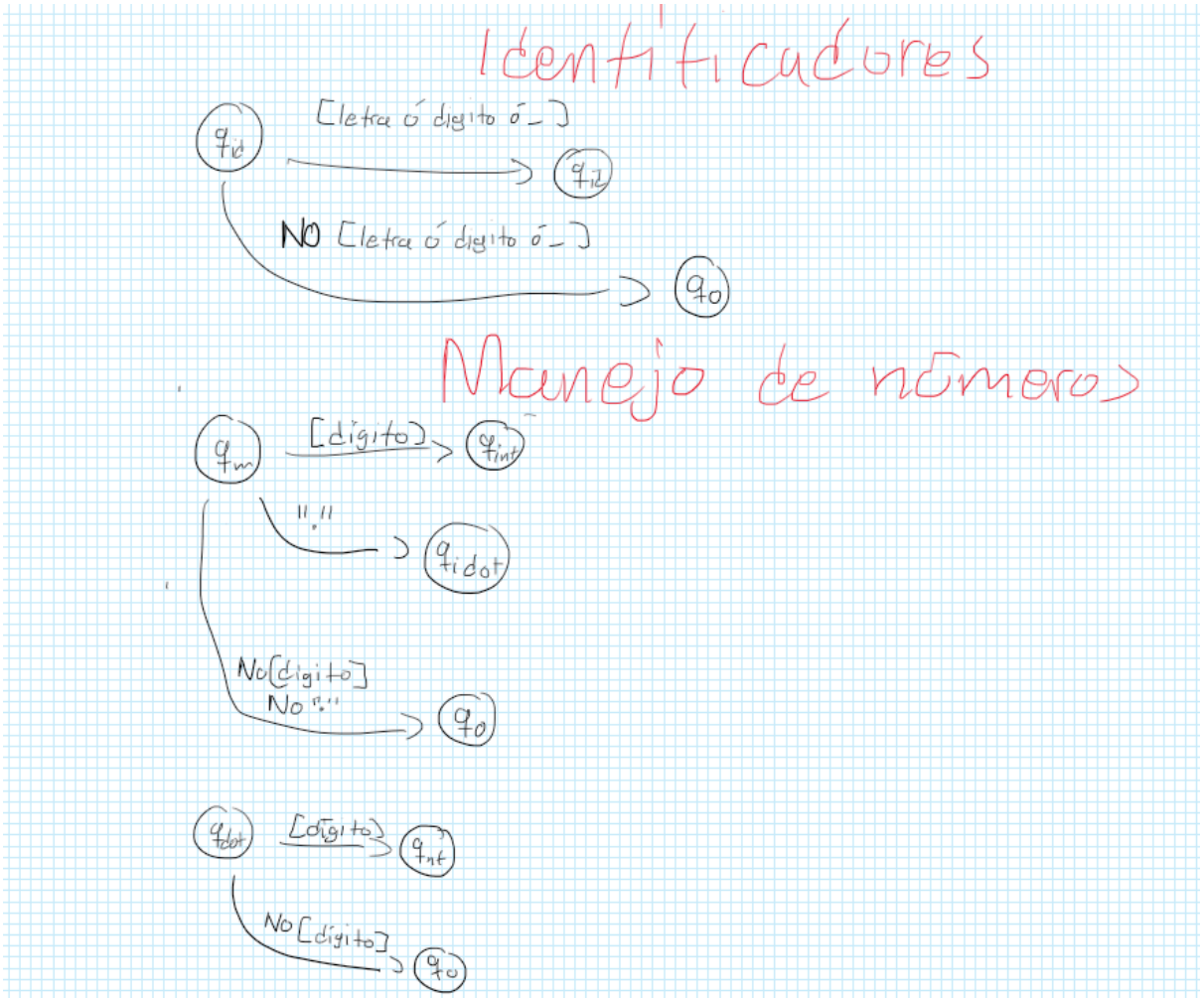
Diseño de autómatas

DFAs que se encargan de procesar cada categoría léxica y el DFA general que procesa todas las subcategorías léxicas.

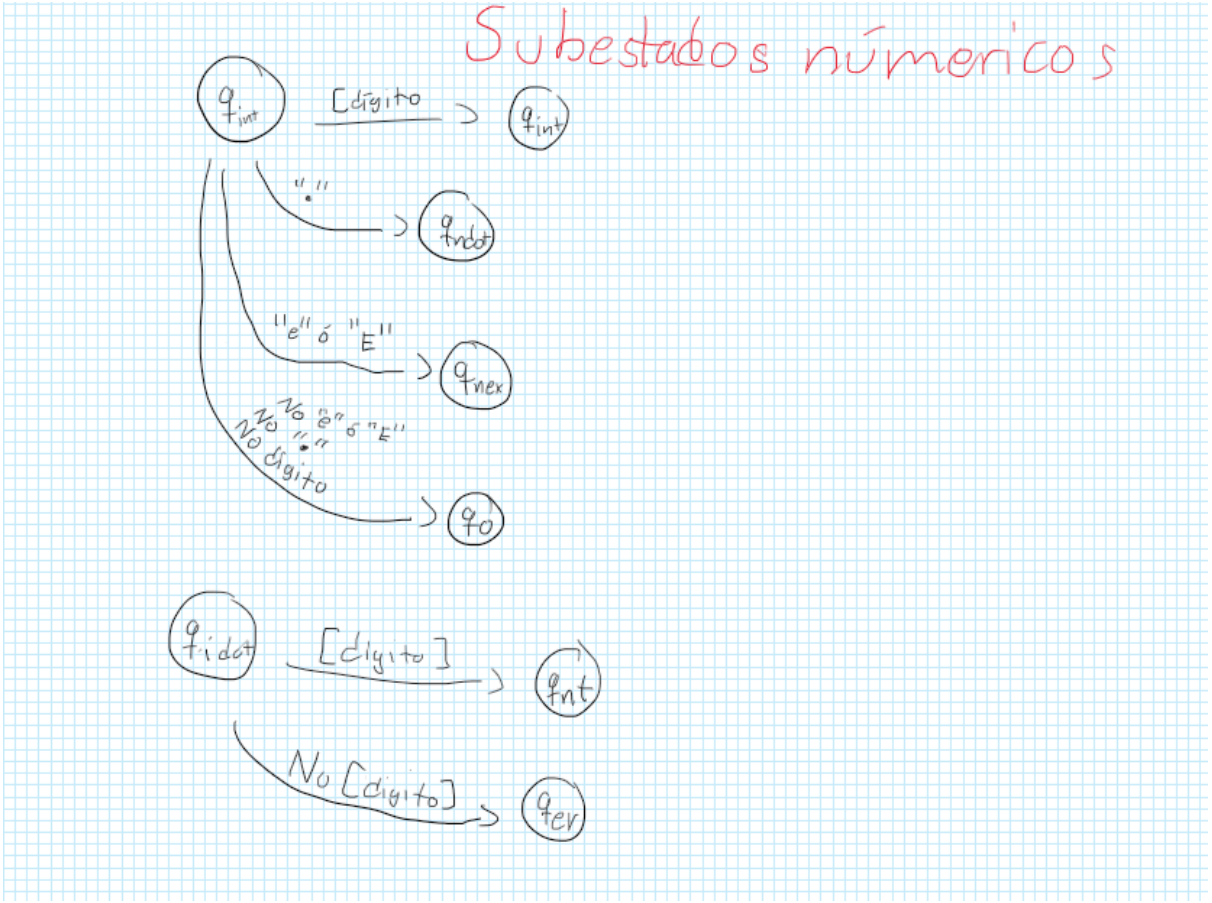
General/Espacio/Comentarios/Strings.

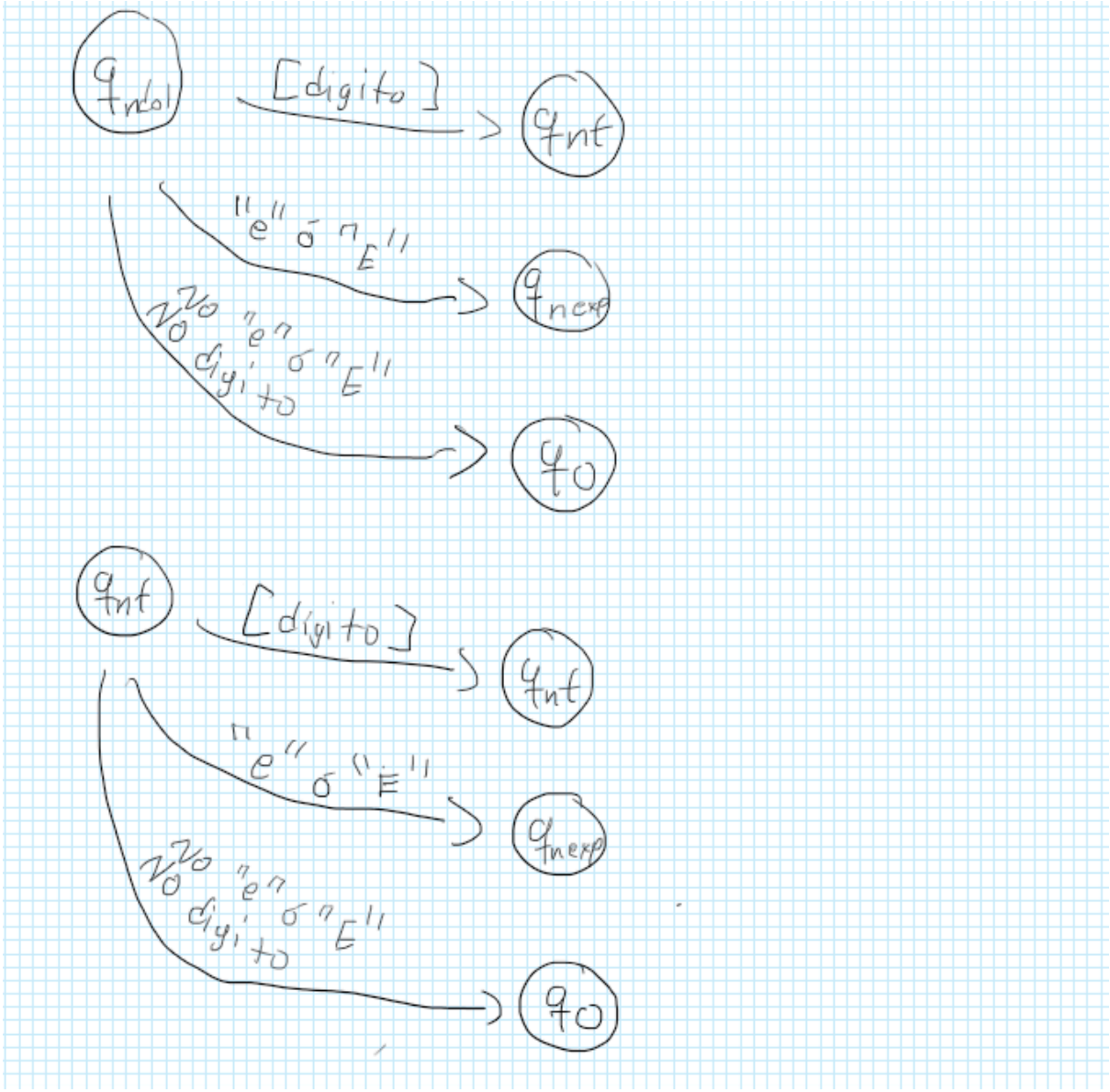


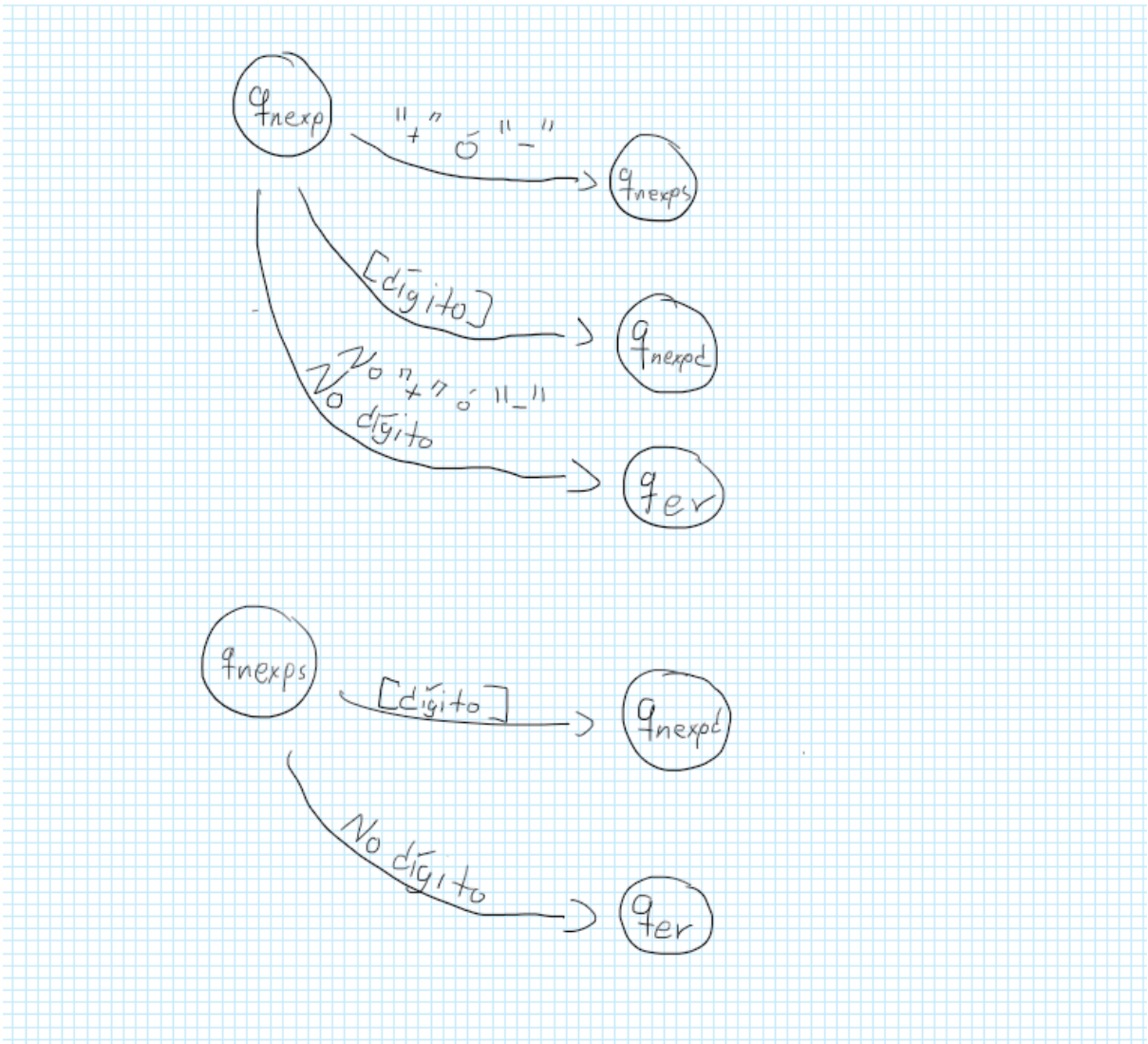
Identificadores/Manejo de números.



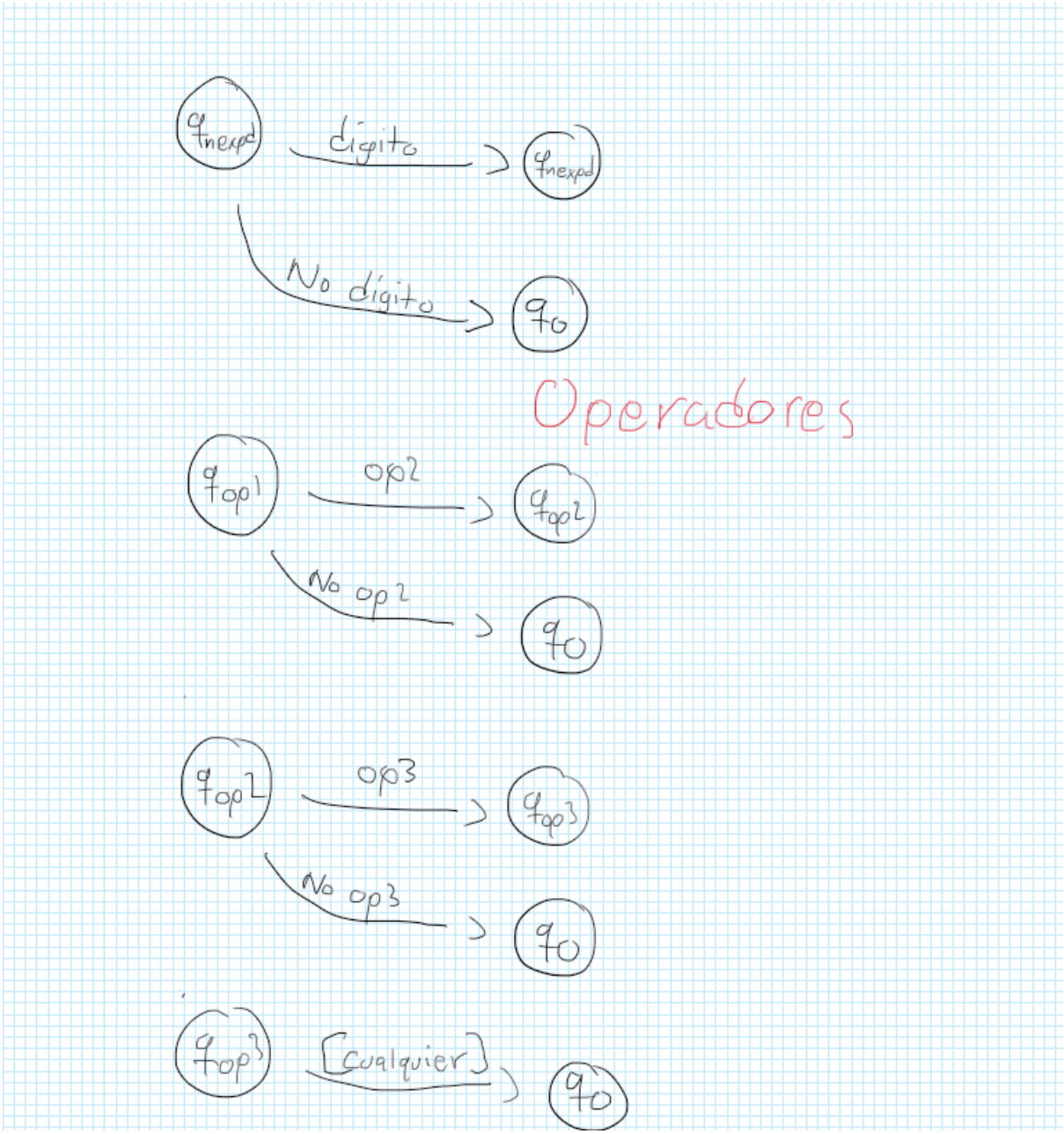
Estados Numéricos



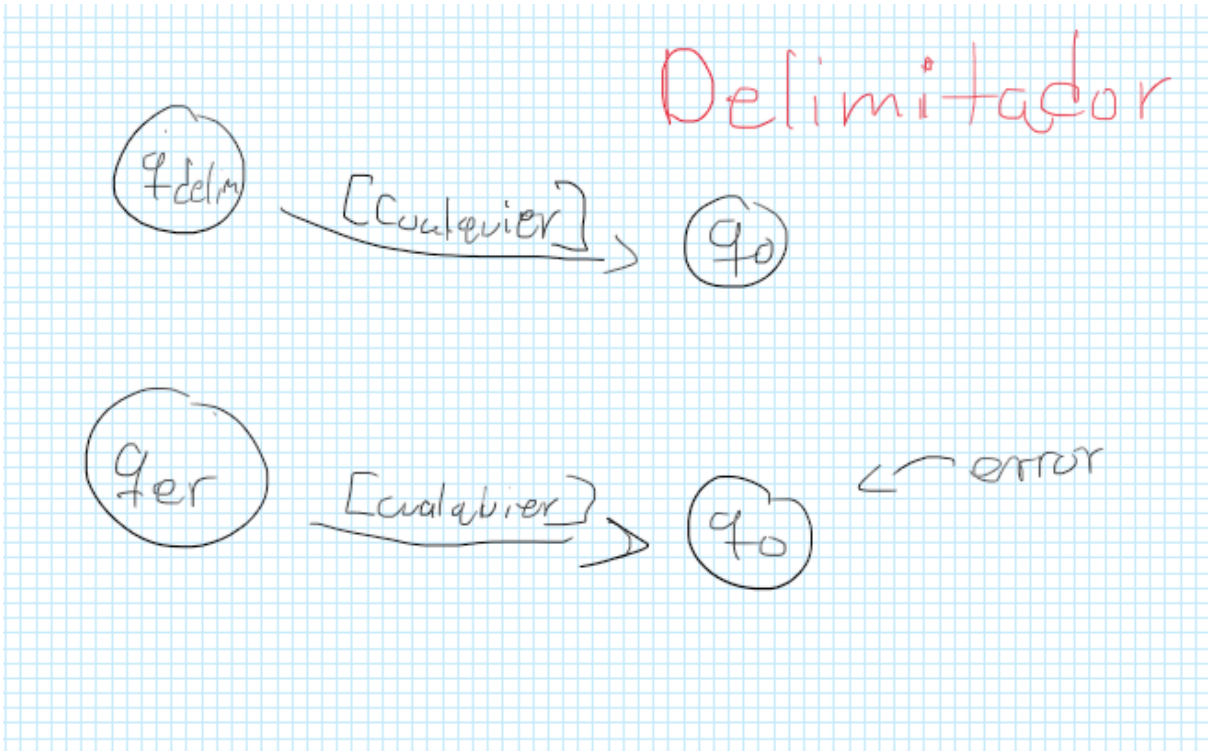

q_{ndol}
[dígito]
q_{nf}
"e" ó "E"
q_{nexp}
Vo "e" ó "E"
Vo dígito
q_0
q_{nf}
[dígito]
q_{nf}
"e" ó "E"
q_{nexp}
Vo "e" ó "E"
Vo dígito
q_0


q_{fnexp}
"+" "-"
q_{fnexps}
[dígito]
q_{fnexpd}
No "+" "-"
No dígito
q_{fer}
q_{fnexps}
[dígito]
q_{fnexpd}
No dígito
q_{fer}

Operadores


q_{nexpd}
dígito
q_{nexpd}
No dígito
q_0
Operadores
q_{op1}
op2
q_{op2}
No op2
q_0
q_{op2}
op3
q_{op3}
No op3
q_0
q_{op3}
[Cualquier]
q_0

Delimitador/Error

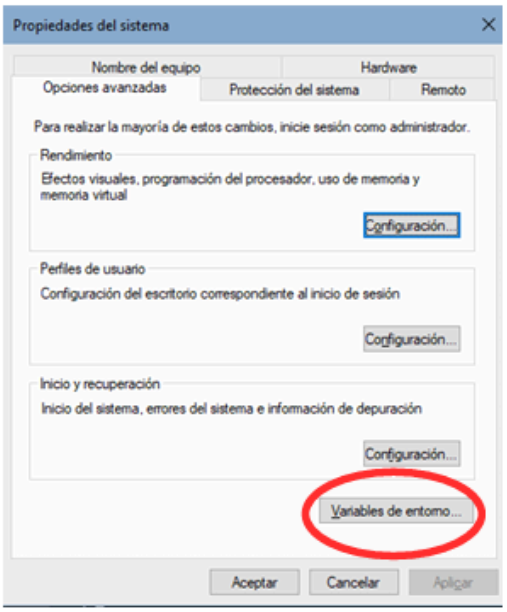


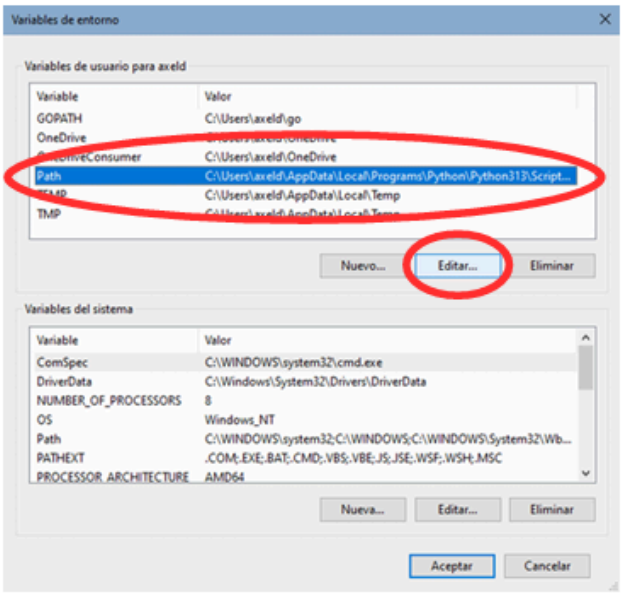
Manual de usuario

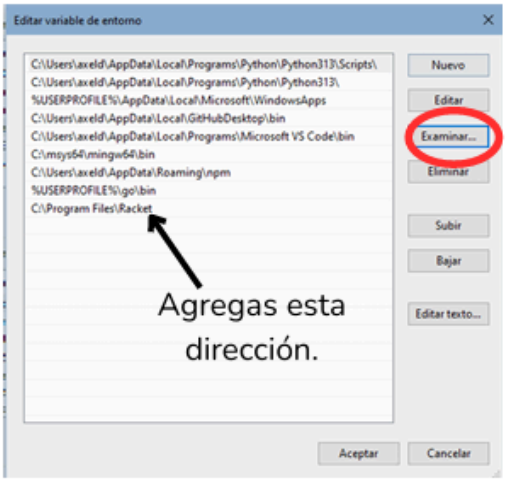
Utilizando Visual Studio Code

Para correr el código debes de inicializar Visual Studio Code con una carpeta que contenga los archivos “Integrador2.rkt” y “test_python.py”.

Agregar al Path una dirección a Racket abriendo “Editar las variables de entorno del sistema”.



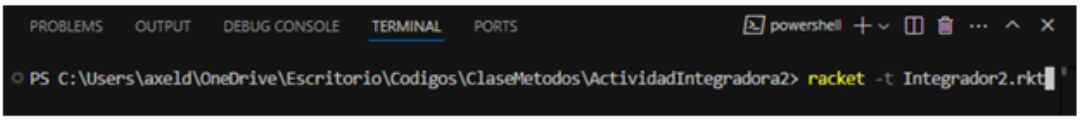




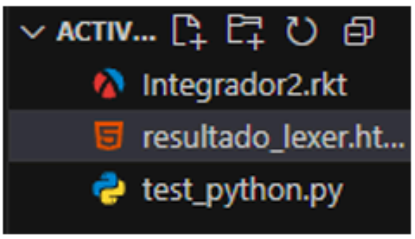
Agregas esta dirección.

Reinicias Visual Studio Code si lo tenias abierto

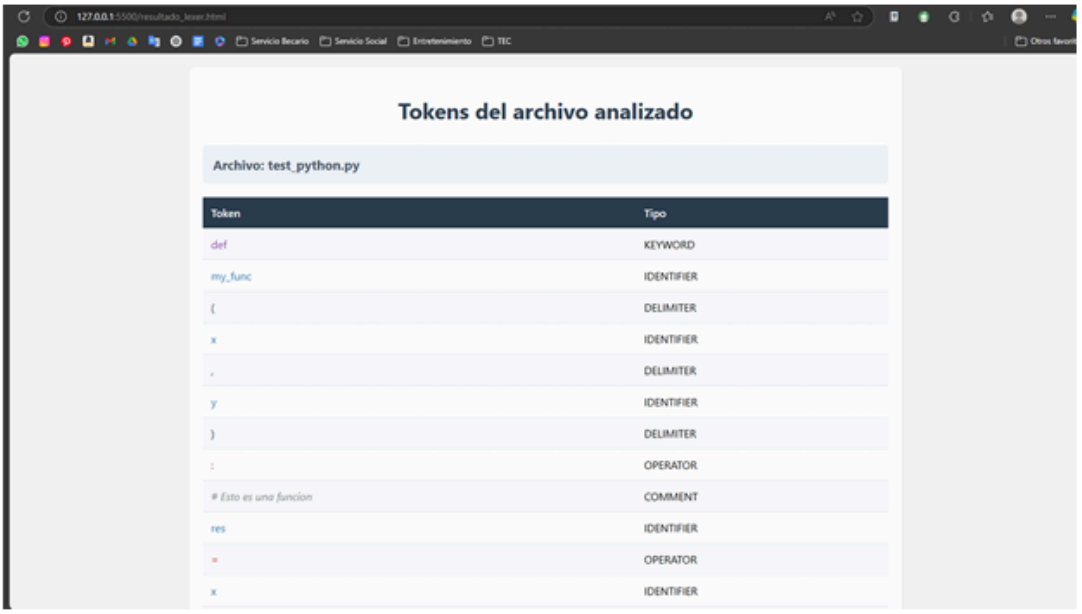
En la terminal pones el siguiente comando: racket -t Integrador2.rkt



Se te va a generar un archivo llamado “resultado_lexer.html”



Lo corres con la extensión de Live Preview o Live Server para visualizar el resultado



Si es que ya tienes un HTML hecho, puedes seguir usándolo para más pruebas, modificando únicamente el archivo “test_python.py”.

En el archivo de “test_python.py”, simplemente con modificarlo es suficiente para que al ejecutar de nuevo el archivo .rkt se puedan visualizar los cambios en el HTML.