

# TRADUCCIÓN DE PSEUDOCÓDIGO A UN LENGUAJE DE PROGRAMACIÓN PARA LA ENSEÑANZA

A N T I A G O   A N D R E S   D E L G A D O   Q U I C E N O - 2 2 1 1 7 9 9  
A M M E R   R O N A L D O   M U Ñ O Z   H E R N Á N D E Z - 2 2 1 1 9 1 8  
A N D R É S   F E L I P E   J A I M E S   O I E D A - 2 2 1 1 2 3 6

# AUTÓMATAS Y LENGUAJES FORMALES

```
10  for(unsigned int i = 1; i <= len1; ++i)
11    for(unsigned int j = 1; j <= len2; ++j)
12      d[i][j] = std::min( std::min(d[i - 1][j] + 1, d[i][j - 1] + 1),
13                           d[i - 1][j - 1] + (s1[i] != s2[j] ? 1 : 0));
14
15  return d[len1][len2];
16 }
17
18
19 template<class T>
20 unsigned int levenshtein_distance(const T &s1, const T &s2) {
21   const size_t len1 = s1.size(), len2 = s2.size();
22   vector<unsigned int> col(len2+1), prevCol(len2+1);
23
24   for (unsigned int i = 0; i < prevCol.size(); i++)
25     prevCol[i] = i;
26   for (unsigned int i = 0; i < len1; i++) {
27     col[0] = i+1;
28     for (unsigned int j = 0; j < len2; j++)
29       col[j+1] = std::min( std::min(prevCol[i + j] + 1,
30                                     prevCol[j] + (s1[i]==s2[j] ? 0 : 1)),
31                           col[j] + (s1[i] != s2[j] ? 1 : 0));
32   }
33   col.swap(prevCol);
34 }
```

# OBJETIVO

---

En este proyecto se usaron **expresiones regulares** para traducir la sintaxis básica del lenguaje de código **Python** a **pseudocódigo**, además de su inverso, es decir, **pseudocódigo a código Python**.

De manera que permita ayudar a asimilar con más facilidad y velocidad las ideas básicas.

# CÓDIGO A PSEUDOCÓDIGO

01

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

print(factorial(5))
```

```
j = True
while j:
    j = False
    print("while")
```

```
for i in range(3):
    print("for")
    if i == 2:
        print("if")
    elif i==1:
        print("1")
    else:
        print("else")
```

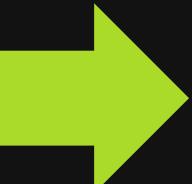
02

Procedimiento factorial(n)  
Si n == 0 entonces  
Devolver 1  
SiNo  
Devolver n \* factorial(n-1)

Escribir (factorial(5))

leer j = True  
Mientras j hacer  
leer j = else  
Escribir ("while")

Para i en range(3) hacer  
Escribir ("for")  
Si i == 2 entonces  
Escribir ("if")  
SiNo i==1 entonces  
Escribir ("1")  
SiNo  
Escribir ("else")



# PROCEDIMIENTO CO - PSE

## Expresiones regulares

Expresiones regulares para patrones específicos en el código Python

```
expresiones = [
    (r'\bdef\b +([?![0-9]]\w+) *\\((.*?)) *:', r'Procedimiento \1(\2)'),
    (r'\bif\b +(.*):', r'Si \1 entonces'),
    (r'\belif\b +(.*?):', r'SiNo \1 entonces'),
    (r'\(\belse\b):', r'SiNo'),
    (r'\bfor\b\s+(\w+)\s+\bin\s+(.*?):', r'Para \1 en \2 hacer'),
    (r'\bwhile\b\s+(.*?):}', r'Mientras \1 hacer'),
    (r'\breturn\b(.*)', r'Devolver \1'),
    (r'\bprint\b(.*)', r'Escribir \1'),
    (r'(\b([?![0-9]]\w+)\b) *= *[^=](.+)', r'leer \1 = \2')
]
```

```
pseudocodigo = codigo_python
```

```
for expresion, reemplazar in expresiones:
    codigo_python = re.sub(expresion, reemplazar, codigo_python,
flags=re.MULTILINE)

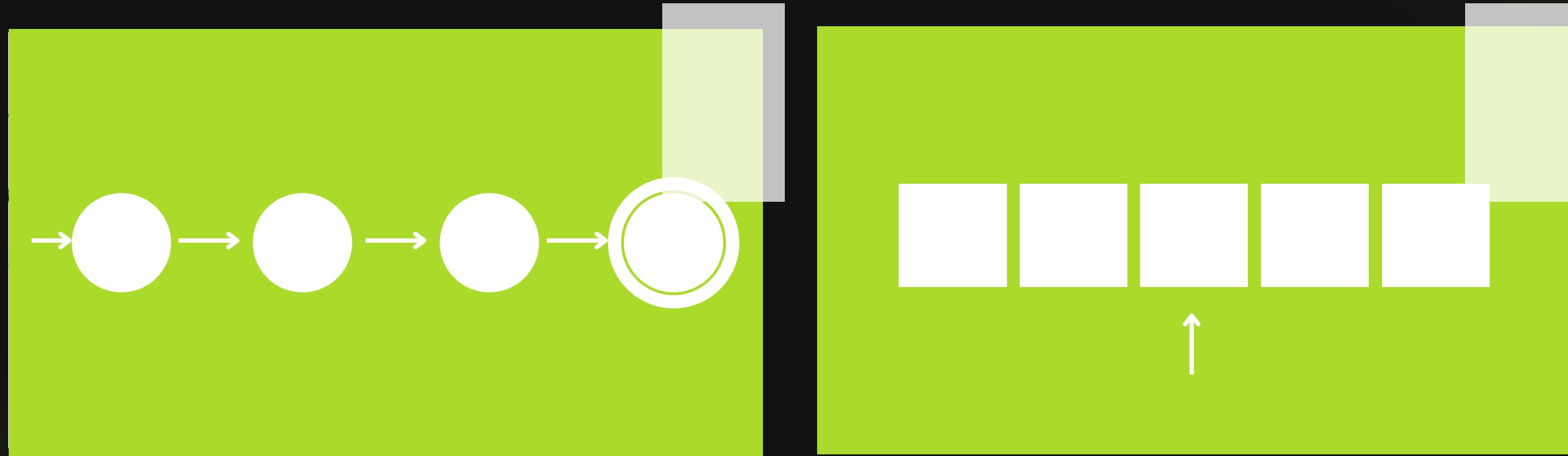
return codigo_python
```

## Traducción

Traduce el código línea por línea

# EQUIVALENCIA

---



## AUTÓMATA FINITO

Se implementa autómatas finitos para la validación del lenguaje.

## MÁQUINA DE TURING

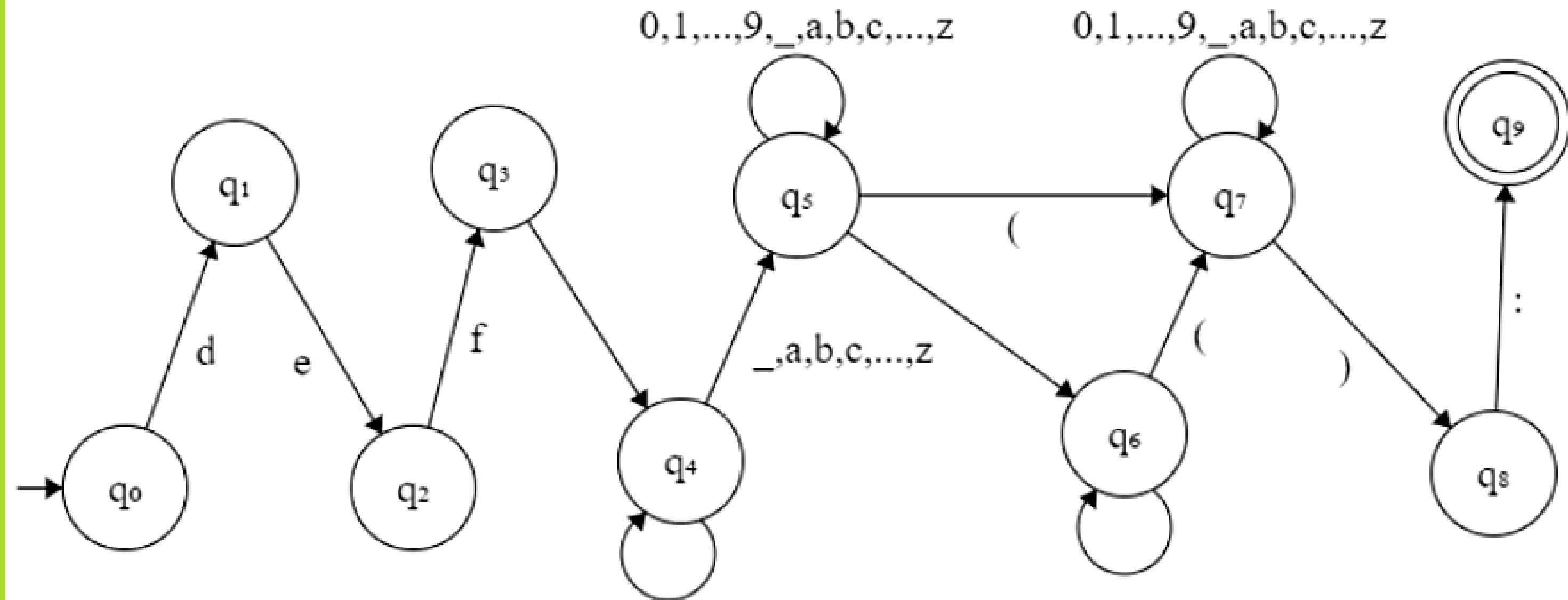
Se implementa máquinas de Turing para realizar la traducción.

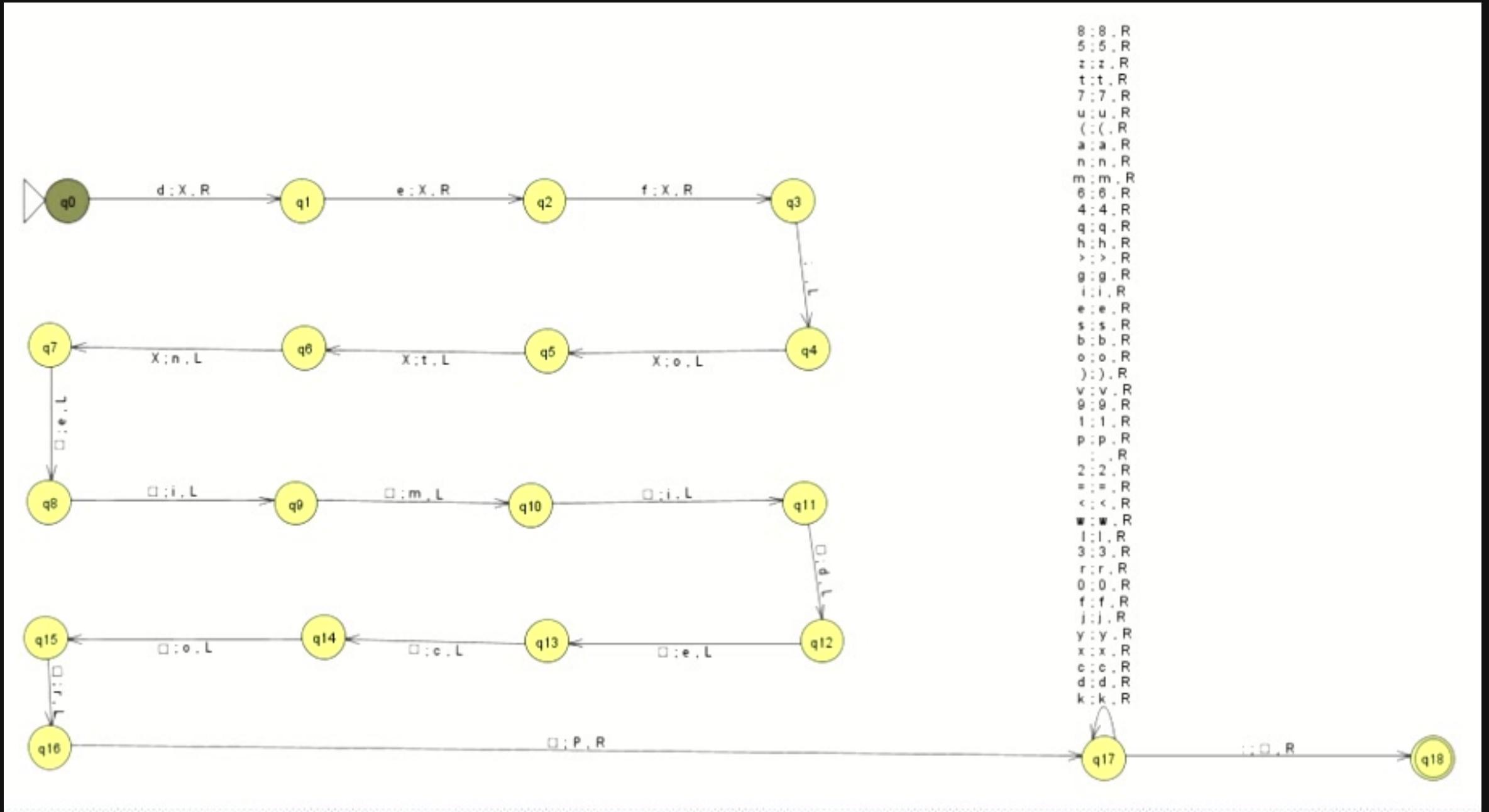
# 01. AUTÓMATA FINITO DETERMINISTA

Verifica el código suministrado en phyton

def factorial(n):

def factorial(n)





## 02. MÁQUINA DE TURING

def factorial(n):

Procedimiento factorial(n)

Traduce



# PSEUDOCÓDIGO A CÓDIGO

01

Procedimiento factorial(n)  
Si n == 0 entonces  
Devolver 1  
SiNo  
Devolver n \* factorial(n-1)

Escribir (factorial(5))

leer j = True  
Mientras j hacer  
leer j = alse  
Escribir ("while")

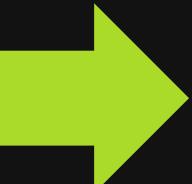
Para i en range(3) hacer  
Escribir ("for")  
Si i == 2 entonces  
Escribir ("if")  
SiNo i==1 entonces  
Escribir ("1")  
SiNo  
Escribir ("else")

02

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n-1)  
  
print(factorial(5))
```

```
j = True  
while j:  
    j = False  
    print("while")
```

```
for i in range(3):  
    print("for")  
    if i == 2:  
        print("if")  
    elif i==1:  
        print("1")  
    else:  
        print("else")
```



# PROCEDIMIENTO CO - PSE

## Expresiones regulares

Expresiones regulares para patrones específicos en el código Python

```
expresiones = [
    (r'\bProcedimiento\b +([?![0-9]]\w+) *\\(.*)\\)', r'def \1(\2):',
    (r'\bSi\b +(.* ) +\bentonces\b', r'if \1:',
    (r'\bSiNo\b +(.*?) +\bentonces\b', r'elif \1:',
    (r'\bSiNo\b', r'else:'),
    (r'\bPara\b +(\w+) +en +( .+?) +hacer', r'for \1 in \2:'),
    (r'\bMientras\b +( .+?) +hacer', r'while \1:'),
    (r'\bDevolver\b +(.* )', r'return \1'),
    (r'\bEscribir\b *(.* )', r'print\1'),
    (r'\bleer\b +(\b([?![0-9]]\w+)\b) *= *( .+ )', r'\1 = \2')
]
```

```
codigo_python = pseudocodigo
```

```
for expresion, reemplazar in expresiones:
    codigo_python = re.sub(expresion, reemplazar, codigo_python,
flags=re.MULTILINE)

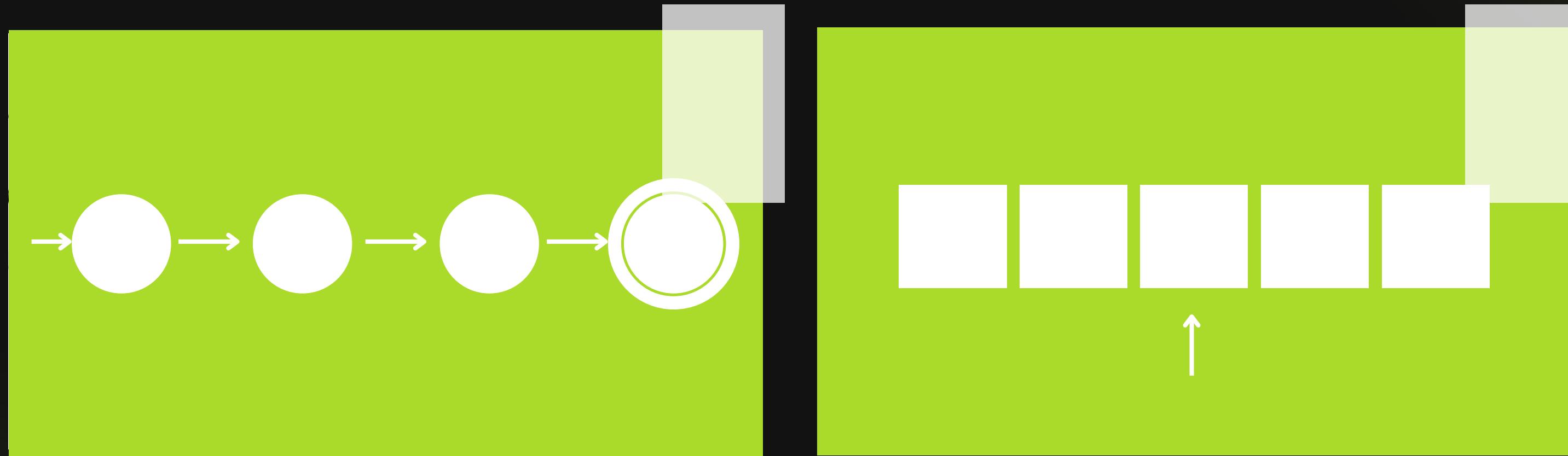
return codigo_python
```

## Traducción

Traduce el código línea por línea

# EQUIVALENCIA

---



## AUTÓMATA FINITO

Se implementa autómatas finitos para la validación del lenguaje.

## MÁQUINA DE TURING

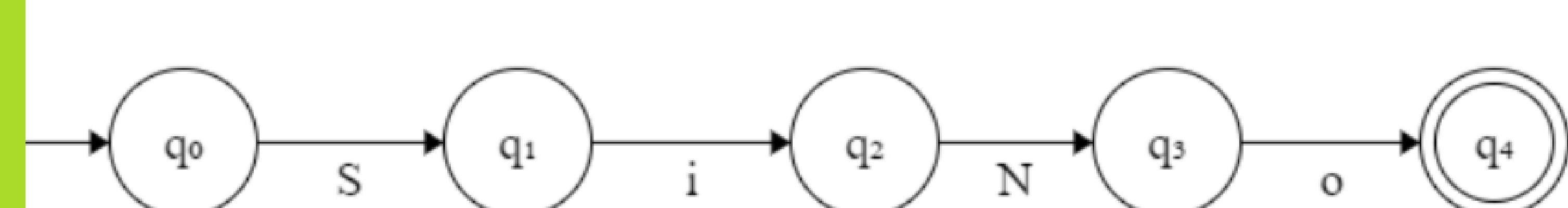
Se implementa máquinas de Turing para realizar la traducción.

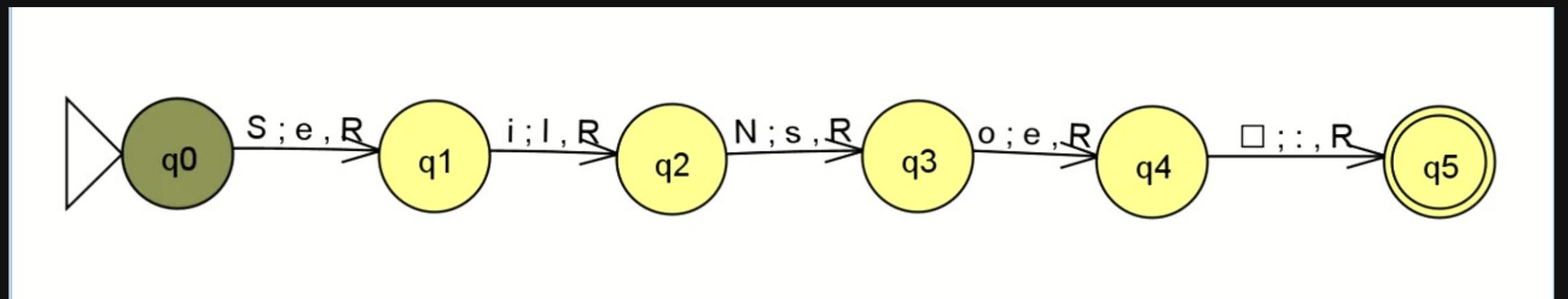
# 01. AUTÓMATA FINITO DETERMINISTA

■ SiNo

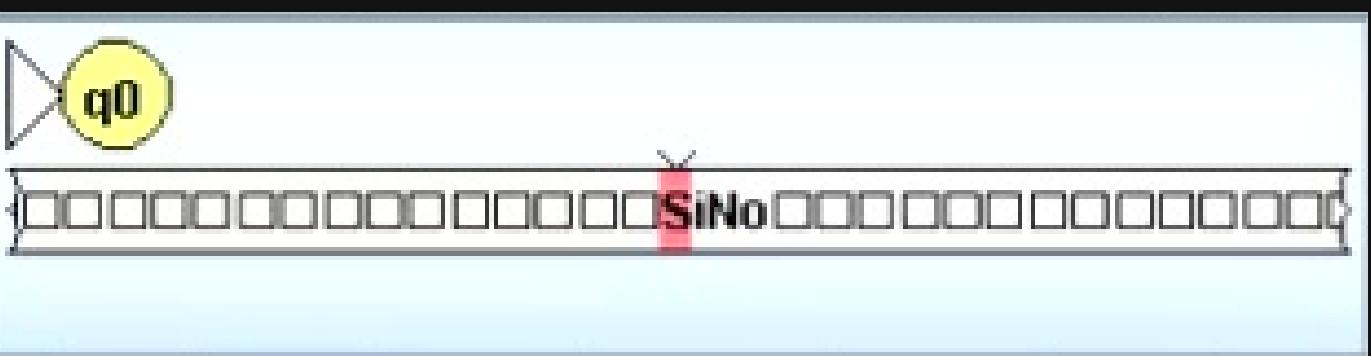
■ Sino

Verifica el  
código  
suministrado  
en phyton





# Traduce



# 02. MÁQUINA DE TURING

SiNo  
else:

# GRACIAS