

Nombre: Monjaraz Briseño Luis Fernando

Materia: Compiladores

Actividad: Hands-on 1 Implementación de Autómatas

Tema: autómatas

Fecha: 09 de marzo de 2025.

Índice

<i>Índice</i>	2
<i>Tabla de imágenes</i>	2
<i>Implementación de autómatas</i>	3
Autómata 1: Validación de cadenas alfabéticas	3
Autómata 2: Validación de números reales.	5
Autómata 3: Validación de sentencias selectivas (if-else).	7
imagen general de evidencia en IDLE Visual Studio Code	10
<i>Referencias bibliográficas</i>	11

Tabla de imágenes

Imagen 1 Código Autómata 1 Python – Visual Studio Code	3
Imagen 2 Resultado Autómata 1 – Terminal Visual Studio Code.....	4
Imagen 3 Código Autómata 2 Python – Visual Studio Code	5
Imagen 4 Resultado Autómata 2 – Terminal Visual Studio Code.....	6
Imagen 5 Código Autómata 3 Python – Visual Studio Code	7
Imagen 6 Resultado Autómata 3 – Terminal Visual Studio Code.....	8
Imagen 7 Evidencia en Visual Studio Code	10

Implementación de autómatas

Autómata 1: Validación de cadenas alfabéticas

```
print("Monjaraz Briseño Luis Fernando")
print("-----")
print("Autómata 1: Validación de cadenas alfabéticas.")

def validate_alpha(s):
    return s.isalpha()

# Original
input_str = "HelloWorld"
if validate_alpha(input_str):
    print("Cadena válida.")
else:
    print("Cadena inválida.")

# Varias cadenas alfabéticas 5 válidas y 5 inválidas.
input_str = ["HelloWorld", "Hello", "World", "HelloWorld123", "Hello123", "123HelloWorld", "Hello World", "Hello123World", "Hello123World!", "Hello123World@"]
for s in input_str:
    if validate_alpha(s):
        print(s, "- Cadena válida.")
    else:
        print(s, "- Cadena inválida.")
```

Imagen 1 Código Autómata 1 Python – Visual Studio Code

Descripción: Este es el autómata 1: validación de cadenas alfabéticas, como se puede apreciar, realice una versión modificada con un ciclo for, esto para facilitar la ejecución del programa y no tener que correrlo varias veces para cada caso, siendo los casos de uso HelloWorld, Hello, World, HelloWorld123, Hello123, 123HelloWorld, Hello World, Hello123World, Hello123World!, Hello123World@, donde los primeros cinco casos son válidos HelloWorld, Hello, World, HelloWorld123, Hello123 y los últimos cinco son inválidos 123HelloWorld, Hello World, Hello123World, Hello123World!, Hello123World@, se podría decir que es el único cambio y agregue. El código se realizó en el IDLE Visual Studio Code y fue programado en Python.

```
Monjaraz Briseño Luis Fernando
-----
Autómata 1: Validación de cadenas alfabéticas.
Cadena válida.
HelloWorld - Cadena válida.
Hello - Cadena válida.
World - Cadena válida.
HelloWorld123 - Cadena inválida.
Hello123 - Cadena inválida.
123HelloWorld - Cadena inválida.
Hello World - Cadena inválida.
Hello123World - Cadena inválida.
Hello123World! - Cadena inválida.
Hello123World@ - Cadena inválida.
-----
```

Imagen 2 Resultado Autómata 1 – Terminal Visual Studio Code

Descripción: Como se puede apreciar los resultados son los esperados. Esto demuestra que se cumple con lo esperado y que el autómata funciona correctamente, siendo el primer print que solo dice “Cadena válida” correspondiente al código original y el resto al código modificado con un ciclo for. A continuación los resultados esperados y que corresponden a los datos.

HelloWorld - Cadena válida.

Hello - Cadena válida.

World - Cadena válida.

HelloWorld123 - Cadena inválida.

Hello123 - Cadena inválida.

123HelloWorld - Cadena inválida.

Hello World - Cadena inválida.

Hello123World - Cadena inválida.

Hello123World! - Cadena inválida.

Hello123World@ - Cadena inválida.

Autómata 2: Validación de números reales.

```
print("-----")
print("Autómata 2: Validación de números reales.")
def validate_real(s):
    try:
        float(s)
        return True
    except ValueError:
        return False

# Original
input_str = "-123.456"
if validate_real(input_str):
    print("Número válido.")
else:
    print("Número inválido.")

# Varias cadenas numéricas 6 válidas y 4 inválidas.
input_str = ["-123.456", "123.456", "123", "-123", "123.", ".123", "123.456.789", "123.456.789", "123,456", "123,456,789"]
for s in input_str:
    if validate_real(s):
        print(s, "- Número válido.")
    else:
        print(s, "- Número inválido.")

print("-----")
```

Imagen 3 Código Autómata 2 Python – Visual Studio Code

Descripción: Este es el autómata 2: validación de números reales. Al igual que en el autómata 1, realicé una versión modificada con un ciclo for para facilitar la ejecución del programa y evitar tener que correrlo varias veces para cada caso. Los casos de uso son -123.456, 123.456, 123, -123, 123., .123, 123.456.789, 123.456.789, 123,456, 123,456,789, donde los primeros seis casos son válidos (-123.456, 123.456, 123, -123, 123., .123) y los últimos cuatro son inválidos (123.456.789, 123.456.789, 123,456, 123,456,789). El código se realizó en el IDLE Visual Studio Code y fue programado en Python.

```
-----  
Autómata 2: Validación de números reales.  
Número válido.  
-123.456 - Número válido.  
123.456 - Número válido.  
123 - Número válido.  
-123 - Número válido.  
123. - Número válido.  
.123 - Número válido.  
123.456.789 - Número inválido.  
123.456.789 - Número inválido.  
123,456 - Número inválido.  
123,456,789 - Número inválido.  
-----
```

Imagen 4 Resultado Autómata 2 – Terminal Visual Studio Code

Descripción: Como se puede apreciar, los resultados son los esperados. Esto demuestra que se cumple con lo esperado y que el autómata funciona correctamente, siendo el primer print que solo dice “Número válido” correspondiente al código original y el resto al código modificado con un ciclo for. A continuación, los resultados esperados y que corresponden a los datos:

-123.456 - Número válido.
123.456 - Número válido.
123 - Número válido.
-123 - Número válido.
123. - Número válido.
.123 - Número válido.
123.456.789 - Número inválido.
123.456.789 - Número inválido.
123,456 - Número inválido.

123,456,789 - Número inválido.

Autómata 3: Validación de sentencias selectivas (if-else).

```
print("-----")
print("Autómata 3: Validación de sentencias selectivas (if-else).")

def validate_if_else(s):
    return "if" in s and "else" in s

# Original
input_str = "if x > 0: pass else: pass"
if validate_if_else(input_str):
    print("Sentencia válida.")
else:
    print("Sentencia inválida.")

# Varias sentencias selectivas 5 válidas y 5 inválidas.
input_str = [
    "if x > 0: pass else: pass",
    "if (x > 0) { } else { }",
    "if x > 0: print(x) else: print('No')",
    "if x > 0: pass elif x == 0: pass else: pass",
    "if x > 0: pass else: pass",
    "if x > 0: pass",
    "else: pass",
    "if x > 0: pass elif x == 0: pass",
    "if x > 0: pass else if x == 0: pass",
    "print('Hello')"
]

for s in input_str:
    if validate_if_else(s):
        print(s, "- Sentencia válida.")
    else:
        print(s, "- Sentencia inválida.")

print("-----")
```

Imagen 5 Código Autómata 3 Python – Visual Studio Code

Descripción: Este es el autómata 3: validación de sentencias selectivas (if-else). Al igual que en los autómatas anteriores, utilicé un ciclo for para facilitar la ejecución del programa y evitar tener que correrlo varias veces para cada caso. Los casos de uso son if x > 0: pass else: pass, if (x > 0) { } else { }, if x > 0: print(x) else: print('No'), if x > 0: pass elif x == 0: pass else: pass, if x > 0: pass else: pass, if x > 0: pass, else: pass, if x > 0: pass elif x == 0: pass, if x > 0: pass else if x == 0: pass, print('Hello'), donde los primeros cinco casos son válidos (if x > 0: pass else: pass, if (x > 0) { } else { }, if x > 0: print(x) else: print('No'), if x > 0: pass elif x == 0: pass else: pass, if x > 0: pass else: pass) y los últimos cinco son inválidos (if x > 0: pass, else: pass, if x > 0: pass elif x == 0: pass, if x > 0: pass else if x == 0: pass, print('Hello')). El código se realizó en el IDLE Visual Studio Code y fue programado en Python.

```
-----
Autómata 3: Validación de sentencias selectivas (if-else).
Sentencia válida.
if x > 0: pass else: pass - Sentencia válida.
if (x > 0) { } else { } - Sentencia válida.
if x > 0: print(x) else: print('No') - Sentencia válida.
if x > 0: pass elif x == 0: pass else: pass - Sentencia válida.
if x > 0: pass else: pass - Sentencia válida.
if x > 0: pass - Sentencia inválida.
else: pass - Sentencia inválida.
if x > 0: pass elif x == 0: pass - Sentencia inválida.
if x > 0: pass else if x == 0: pass - Sentencia válida.
print('Hello') - Sentencia inválida.
-----
```

Imagen 6 Resultado Autómata 3 – Terminal Visual Studio Code

Descripción: Como se puede apreciar, los resultados son los esperados. Esto demuestra que se cumple con lo esperado y que el autómata funciona correctamente, siendo el primer print que solo dice “Sentencia válida” correspondiente al código original y el resto al código

modificado con un ciclo for. A continuación, los resultados esperados y que corresponden a los dados:

if x > 0: pass else: pass - Sentencia válida.

if (x > 0) { } else { } - Sentencia válida.

if x > 0: print(x) else: print('No') - Sentencia válida.

if x > 0: pass elif x == 0: pass else: pass - Sentencia válida.

if x > 0: pass else: pass - Sentencia válida.

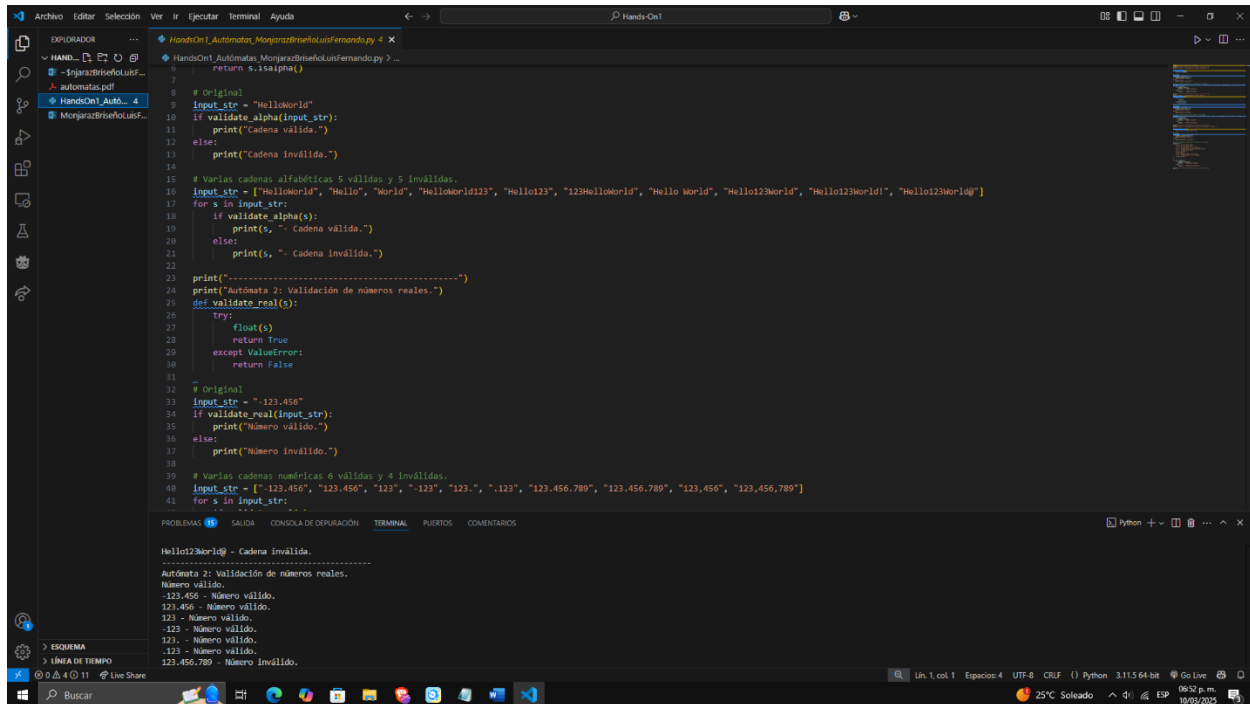
if x > 0: pass - Sentencia inválida.

else: pass - Sentencia inválida.

if x > 0: pass elif x == 0: pass - Sentencia inválida.

if x > 0: pass else if x == 0: pass - Sentencia inválida.

print('Hello') - Sentencia inválida.

imagen general de evidencia en IDLE Visual Studio Code

The screenshot displays the Visual Studio Code interface with a Python file named `HandsOn1_Automas_MonjazzBrisefolusFernando.py`. The code implements two automata: one for validating strings of letters and another for validating real numbers. The terminal output shows the results of running the code.

```
7
8
9 # Original
10 input_str = "HelloWorld"
11 if validate_alpha(input_str):
12     print("Cadena válida.")
13 else:
14     print("Cadena inválida.")
15
16 # Varias cadenas alfabéticas 5 válidas y 5 inválidas.
17 input_str = ["HelloWorld", "Hello", "World", "HelloWorld123", "Hello123", "123HelloWorld", "Hello World", "Hello123World", "Hello123World@"]
18 for s in input_str:
19     if validate_alpha(s):
20         print(s, "- Cadena válida.")
21     else:
22         print(s, "- Cadena inválida.")
23
24 print("-----")
25 print("Automata 2: Validación de números reales.")
26 def validate_real(s):
27     try:
28         float(s)
29         return True
30     except ValueError:
31         return False
32
33 # Original
34 input_str = "-123.456"
35 if validate_real(input_str):
36     print("Número válido.")
37 else:
38     print("Número inválido.")
39
40 # Varias cadenas numéricas 6 válidas y 4 inválidas.
41 input_str = ["-123.456", "123.456", "123", "-123", "123.", "123.", "123.", "123.456.789", "123.456.789", "123.456", "123.456.789"]
42 for s in input_str:
43     ...
```

Terminal Output:

```
Hello123World@ - Cadena inválida.
-----
Automata 2: Validación de números reales.
Número válido.
-123.456 - Número válido.
123.456 - Número válido.
123 - Número válido.
-123 - Número válido.
123. - Número válido.
123. - Número válido.
123.456.789 - Número inválido.
```

Imagen 7 Evidencia en Visual Studio Code

Referencias bibliográficas

automata-lib. (2025, January 23). Retrieved from <https://pypi.org/project/automata-lib/>

Fernandez, R. (2019, May 28). Autómata Finito en Python - ▷ Cursos de Programación de 0 a Experto © Garantizados. Retrieved from <https://unipython.com/automata-finito-en-python/>

Mateo Ortega. (2022, July 5). *Autómata Finito Determinista Reconocedor implementado en python* [Video file]. Retrieved from <https://www.youtube.com/watch?v=ReR1hOhQKHY>