Ejercicios sobre máquinas de Turing

Julian Esteban Ballesteros Ortiz

Programa de ingeniería de sistemas y computación, Universidad de Cundinamarca

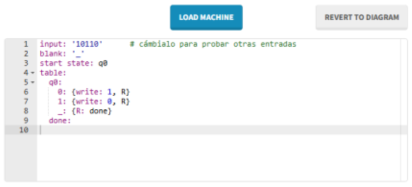
Lenguajes y Autómatas 801 ISC TRAN

Docente

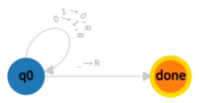
Ing. Fabio Alejandro Sastoque Rincón

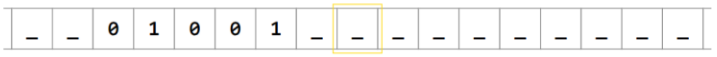
Miércoles 1 de octubre de 2025

Problema 1

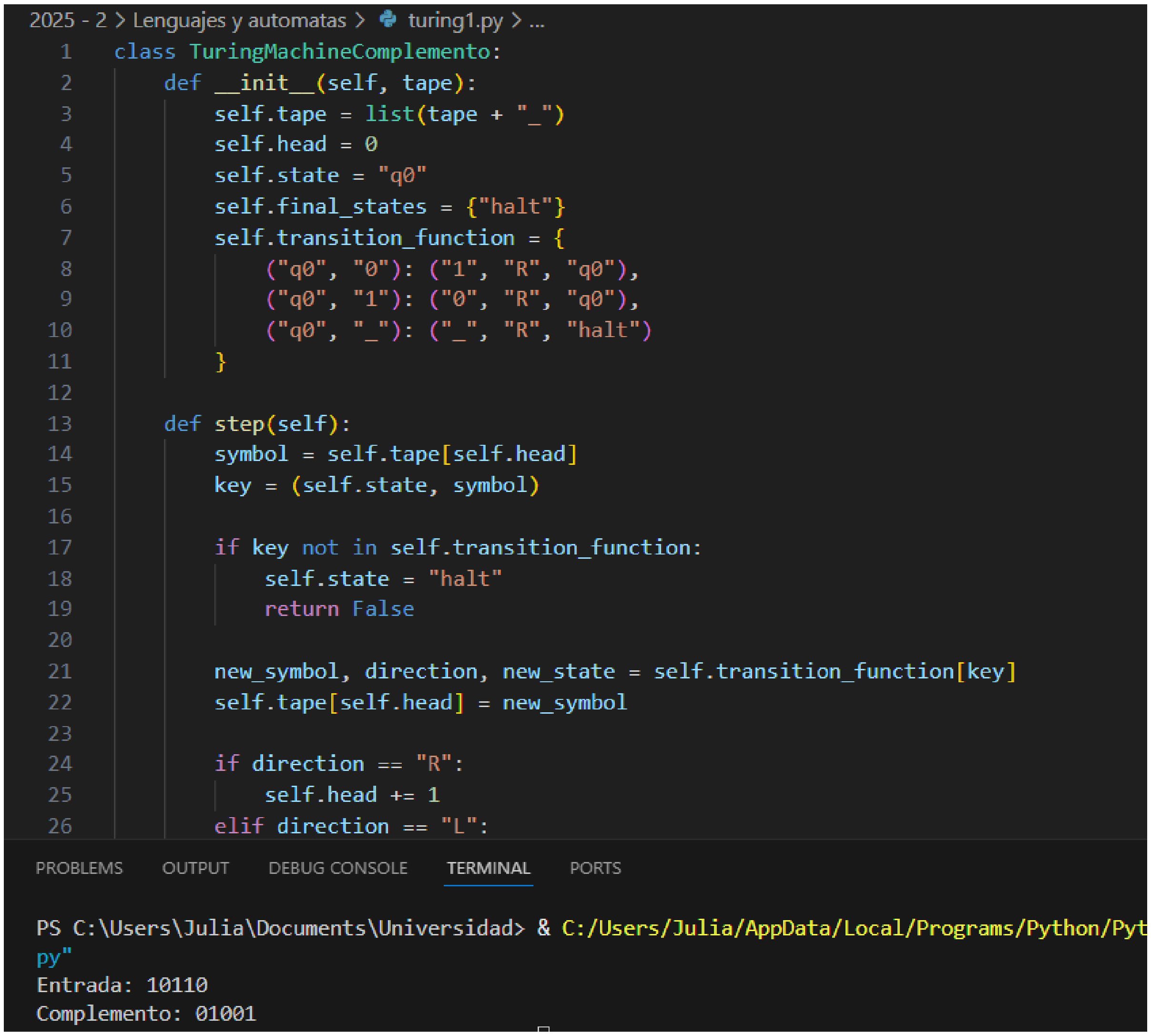


```
input: '10110'      # cámbialo para probar otras entradas
blank: '_'
start state: q0
table:
  q0:
    0: {write: 1, R}
    1: {write: 0, R}
    _: {R: done}
  done:
```





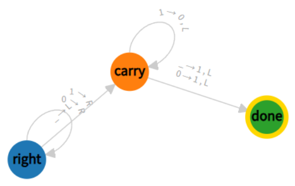
```
_ _ 0 1 0 0 1 _ _ _ _ _ _ _ _ _ _
```

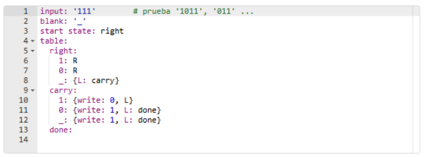


```python
class TuringMachineComplemento:
    def __init__(self, tape):
        self.tape = list(tape + "_")
        self.head = 0
        self.state = "q0"
        self.final_states = {"halt"}
        self.transition_function = {
            ("q0", "0"): ("1", "R", "q0"),
            ("q0", "1"): ("0", "R", "q0"),
            ("q0", "_"): ("_", "R", "halt")
        }

    def step(self):
        symbol = self.tape[self.head]
        key = (self.state, symbol)

        if key not in self.transition_function:
            self.state = "halt"
            return False

        new_symbol, direction, new_state = self.transition_function[key]
        self.tape[self.head] = new_symbol

        if direction == "R":
            self.head += 1
        elif direction == "L":
```

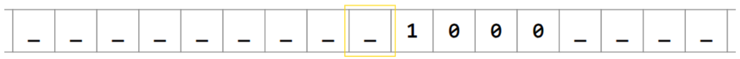
```
PS C:\Users\Julia\Documents\Universidad> & C:/Users/Julia/AppData/Local/Programs/Python/Pyt
py"
Entrada: 10110
Complemento: 01001
```

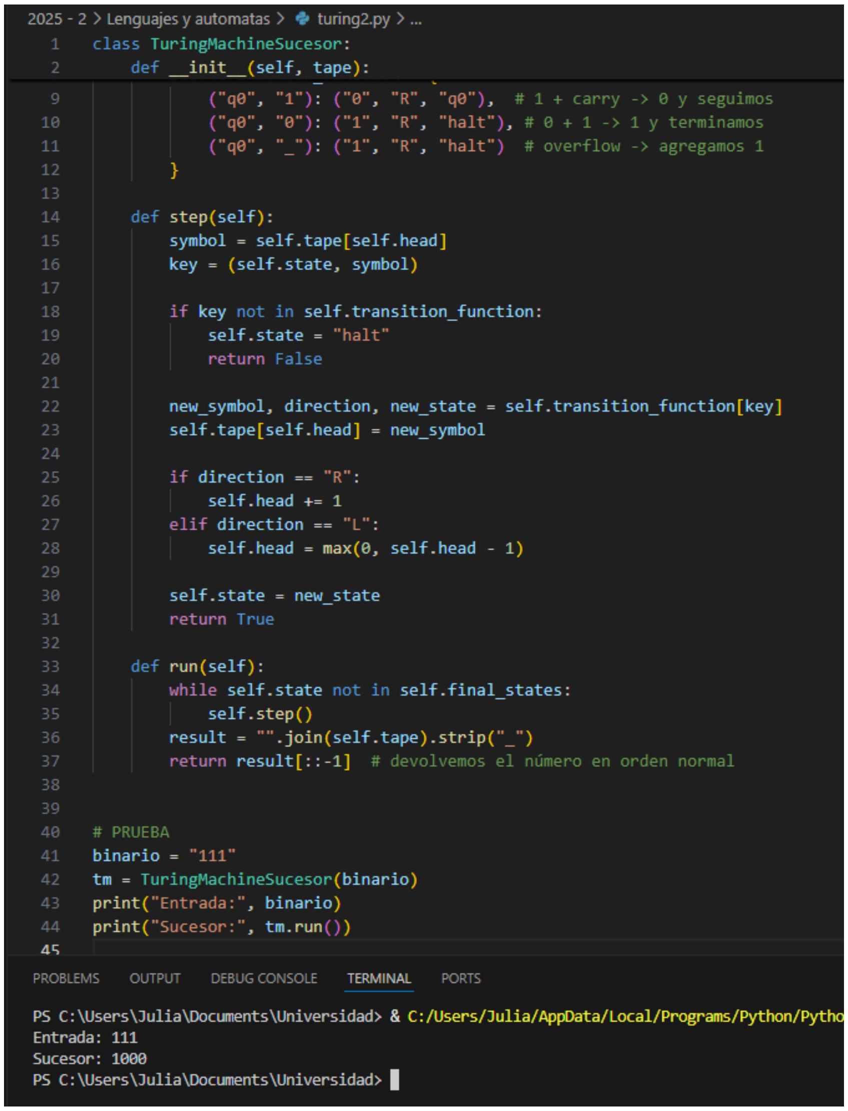
Problema 2





```
input: '111'        # prueba '1011', '011' ...
blank: '_'
start state: right
table:
  right:
    1: R
    0: R
    _: {L: carry}
  carry:
    1: {write: 0, L}
    0: {write: 1, L: done}
    _: {write: 1, L: done}
  done:
```

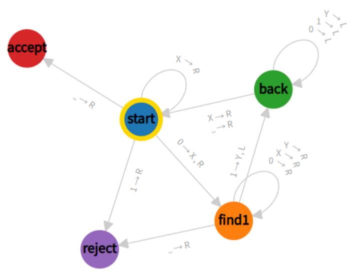


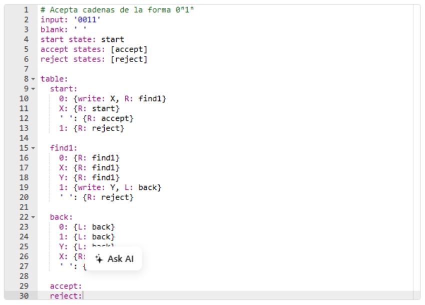


```python
class TuringMachineSucesor:
    def __init__(self, tape):
                ("q0", "1"): ("0", "R", "q0"),  # 1 + carry -> 0 y seguimos
                ("q0", "0"): ("1", "R", "halt"), # 0 + 1 -> 1 y terminamos
                ("q0", "_"): ("1", "R", "halt")  # overflow -> agregamos 1
            }

    def step(self):
        symbol = self.tape[self.head]
        key = (self.state, symbol)

        if key not in self.transition_function:
            self.state = "halt"
            return False

        new_symbol, direction, new_state = self.transition_function[key]
        self.tape[self.head] = new_symbol

        if direction == "R":
            self.head += 1
        elif direction == "L":
            self.head = max(0, self.head - 1)

        self.state = new_state
        return True

    def run(self):
        while self.state not in self.final_states:
            self.step()
        result = "".join(self.tape).strip("_")
        return result[::-1]  # devolvemos el número en orden normal


# PRUEBA
binario = "111"
tm = TuringMachineSucesor(binario)
print("Entrada:", binario)
print("Sucesor:", tm.run())
```

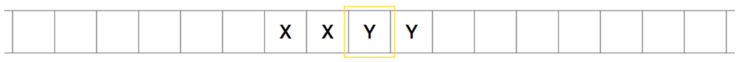
```
PS C:\Users\Julia\Documents\Universidad> & C:/Users/Julia/AppData/Local/Programs/Python/Pytho
Entrada: 111
Sucesor: 1000
PS C:\Users\Julia\Documents\Universidad>
```

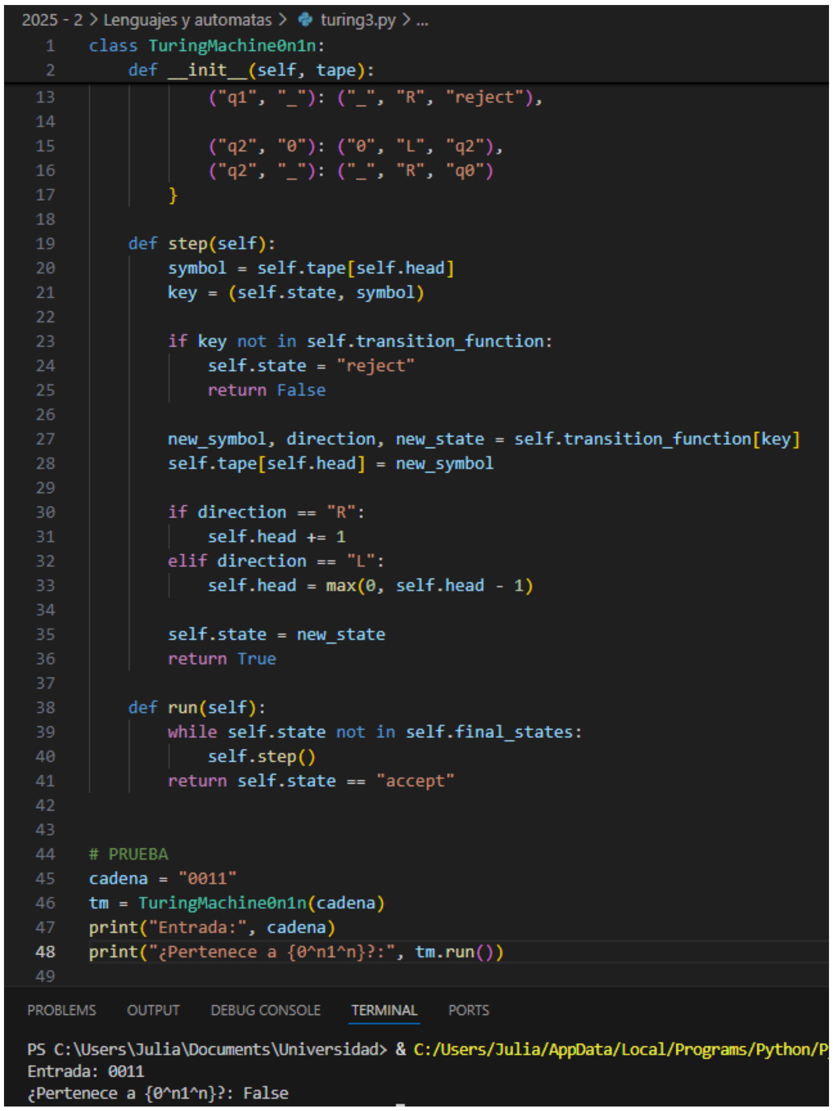
Problema 3





```
# Acepta cadenas de la forma 0ⁿ1ⁿ
input: '0011'
blank: ' '
start state: start
accept states: [accept]
reject states: [reject]

table:
  start:
    0: {write: X, R: find1}
    X: {R: start}
    ' ': {R: accept}
    1: {R: reject}

  find1:
    0: {R: find1}
    X: {R: find1}
    Y: {R: find1}
    1: {write: Y, L: back}
    ' ': {R: reject}

  back:
    0: {L: back}
    1: {L: back}
    Y: {L: back}
    X: {R:
    ' ': {

  accept:
  reject:
```



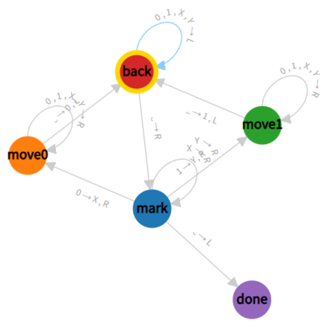


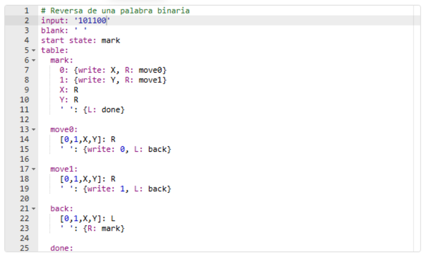
2025 - 2 > Lenguajes y automatas > turing3.py > ...

```python
class TuringMachine0n1n:
    def __init__(self, tape):
                ("q1", "_"): ("_", "R", "reject"),

                ("q2", "0"): ("0", "L", "q2"),
                ("q2", "_"): ("_", "R", "q0")
            }

    def step(self):
        symbol = self.tape[self.head]
        key = (self.state, symbol)

        if key not in self.transition_function:
            self.state = "reject"
            return False

        new_symbol, direction, new_state = self.transition_function[key]
        self.tape[self.head] = new_symbol

        if direction == "R":
            self.head += 1
        elif direction == "L":
            self.head = max(0, self.head - 1)

        self.state = new_state
        return True

    def run(self):
        while self.state not in self.final_states:
            self.step()
        return self.state == "accept"


# PRUEBA
cadena = "0011"
tm = TuringMachine0n1n(cadena)
print("Entrada:", cadena)
print("¿Pertenece a {0^n1^n}?:", tm.run())
```

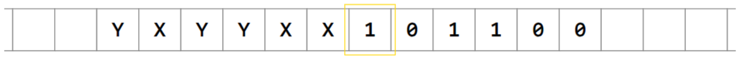
```
PS C:\Users\Julia\Documents\Universidad> & C:/Users/Julia/AppData/Local/Programs/Python/P
Entrada: 0011
¿Pertenece a {0^n1^n}?: False
```

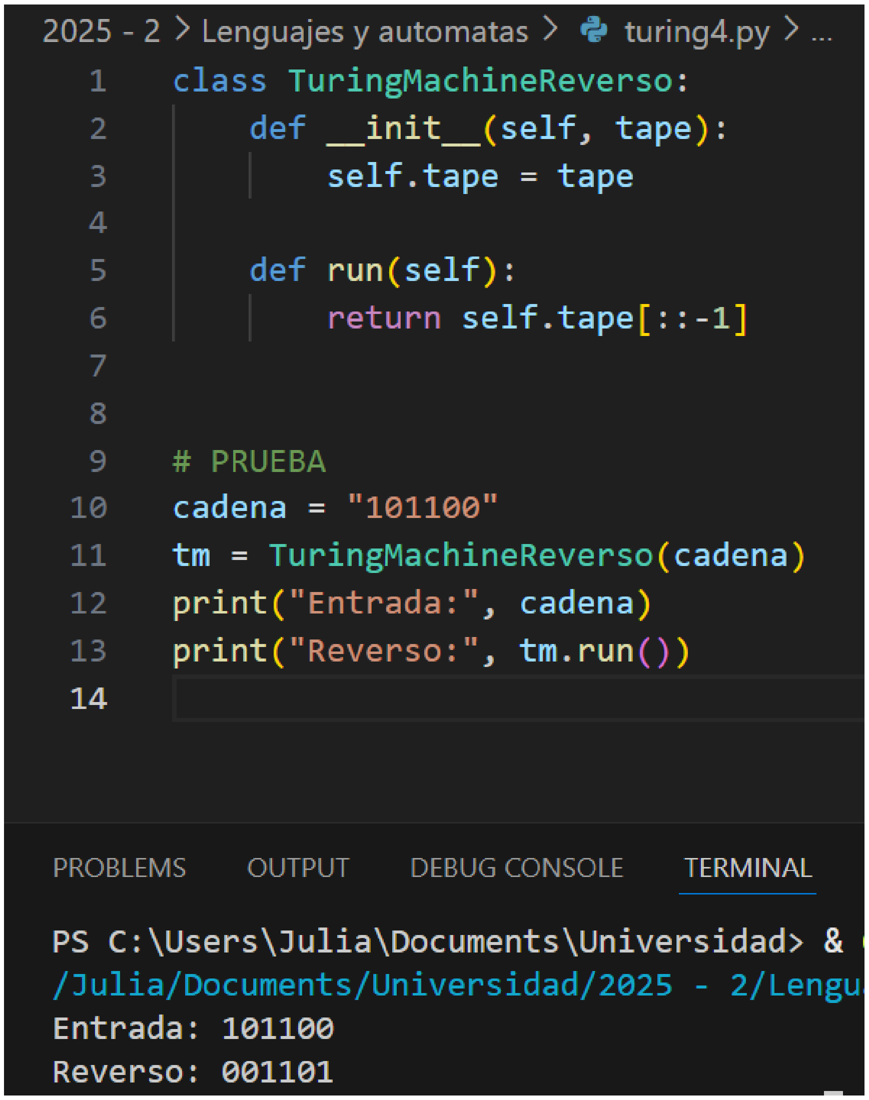
Problema 4





```
# Reversa de una palabra binaria
input: '101100'
blank: ' '
start state: mark
table:
  mark:
    0: {write: X, R: move0}
    1: {write: Y, R: move1}
    X: R
    Y: R
    ' ': {L: done}

  move0:
    [0,1,X,Y]: R
    ' ': {write: 0, L: back}

  move1:
    [0,1,X,Y]: R
    ' ': {write: 1, L: back}

  back:
    [0,1,X,Y]: L
    ' ': {R: mark}

  done:
```





```python
class TuringMachineReverso:
    def __init__(self, tape):
        self.tape = tape

    def run(self):
        return self.tape[::-1]


# PRUEBA
cadena = "101100"
tm = TuringMachineReverso(cadena)
print("Entrada:", cadena)
print("Reverso:", tm.run())

```

```
PS C:\Users\Julia\Documents\Universidad> & 
/Julia/Documents/Universidad/2025 - 2/Lengu
Entrada: 101100
Reverso: 001101
```