Date
Page

Lab-2 Vacuum Cleaning Agent (Reflex-based)

→ **Initial Conditions:**

```
# Setting up a 2-room space with a 1x2
# array
    arr = ["None, None]
  # Setting up a randomized environment
    ~~for i in range~~ (2)
        ~~arr [0] = arr [i] =~~
    for i ← 0 to 2
        arr [i] = random.choice ("C", "D")
                                    and p-sequence
 # Setting up a position for the vacuum cleaner
    pos = 0 ; pSeq = []
```

→ **Algorithm for cleaning the sequence of rooms**

```
  def ~~clean~~ (pos, arr):
     ~~if pos pSeq. append~~ (
  while True:          # Infinite loop
     pSeq. append ( ~~arr [pos]~~, (pos, arr[pos]) )
     if arr [pos] == "D"
          print (" Cleaning room {}".format
                        (pos)); arr[pos] = "C"
     else arr [pos] == "C":
          print (" Room is clean")
  ~~# Randomize environment again~~
     ~~for i in range (2)~~
        ~~arr [i] = random choice ("C", "D")~~
  # ~~Increment~~ position
        pos = ~~(pos +1) % 2~~
  #   arr [pos] = random.choice ("C", "D")
      pos = (pos +1) % 2
```

Date
Page

→ Utility Functions:
→ display():
```
def display(~~E~~ arr, pseq):
    print(" Percept Sequence: ", pseq);
~~def~~ print(" Current env: ", arr);
```

→ Sample Percept Sequence:
Initial env: pSeq = []
arr = [~~4~~"C", "D"]
pos = 0

Showing 5 Steps
S1: pSeq = [(0, "C")]
~~arr = [(1, "D") (0, "C"), (1, "D")]~~

S2: pSeq = [(0, "C"), (1, "D")]
Cleans in this case

S3: ~~PSeq~~ pSeq = [(0, "C"), (1, "D"),
(0, "D")]

S4: pSeq = [(0, "C"), (1, "D"),
(0, "D"), (1, "C")]

S5: pSeq = [(0, "C"), (1, "D"),
(0, "D"), (1, "C"),
(0, "C")]

[illegible]

→ Program:

```
import random
class Cleaner:
    def __init__(self):
        Self.env = [None, None]
        for i in range(2):
            Self.env[i] = random.choice(["C", "D"])
        Self.pos = 0
        Self.pseq = []
        Self.clean()

    def display(self):
        Print(" Current percept seq: ", Self.pseq)
        Print(" Current env: ", self.env)

    def clean(self):
        while True:
            Self.pseq.append((self.pos,
                              Self.env[self.pos]))
            if Self.env[self.pos] == "D":
                print(f" Room {self.pos} is dirty, cleaning... ")
                Self.env[self.pos] = "C"
            else:
                print(" Room is clean... ")
                If Self.pos == 1:
                    Print(" Moving left... ")
                else:
                    print (" Moving right... ")
            Self.display()
            Self.env[self.pos] =
                        random.choice(["C", "D"])
            self.pos = (Self.pos +1) % 2

c = Cleaner()
```

Output:

```
Room 0 is dirty, cleaning...
Current percept seq: [(0, 'D')]
Current env: ['C', 'D']
Room 1 is dirty, cleaning...
Current percept seq: ~~['D'~~, [(0, 'D'), (1, 'D'
Current env: ['D', 'C']
Room 0 is dirty, cleaning...
Current percept seq: [(0, 'D'), (1, 'D'), (0,
Current env: ['C', 'C']
Room ~~1 is cl~~ is clean...
Moving left...
Current percept seq: [(0, 'D'), (1, 'D'),
                      (0, 'D'), (1, 'C')]
Current env: ['D', 'C']
```

[illegible] 1/10/24