

In [1]:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, Aer, transpile
2 import numpy as np
3 from qiskit.visualization import plot_histogram
4 from qiskit import *
5 import random
6 import matplotlib.pyplot as plt
7 from operator import attrgetter
8 import matplotlib.pyplot as plt
9 import heapq
10 from operator import itemgetter
11 from pydub import AudioSegment # for audio
12 from pydub.playback import play # for audio

```

In [2]:

```

1 # Target & reward -----

```

In [3]:

```

1 class Target:
2     def __init__(self, name, x, y): # no indetermination in the target's position
3         self.name = name
4         self.x = x
5         self.y = y

```

In [4]:

```

1 T = Target("T", 0.9, 0.5) # deep in the ocean
2
3 # for getting back to the beginning
4 T2 = Target("T2", 0.2, 0.5) # back to the ship

```

In [5]:

```

1 def reward(T, betax, betay):
2     return 1 - ((T.x - betax)**2 + (T.y - betay)**2)**0.5
3     # the closer the target, the less the distance, the higher the reward
4

```

In [6]:

```

1 # Obstacles -----

```

In [7]:

```

1 class Obstacle: # Just a point for now
2     def __init__(self, name, x, y):
3         self.name = name
4         self.x = x
5         self.y = y

```

In [8]:

```

1 O = Obstacle("Oo", 0.8, 0.2) # deep in the ocean

```

In [9]:

```
1 # Robots -----
```

In [10]:

```
1 class Robotx(object):
2     _registry = []
3
4     def __init__(self, name, alphax, betax, alphay, betay, gamma, delta, position):
5         self._registry.append(self)
6         self.name = name
7         self.alphax = alphax
8         self.betax = betax
9         self.alphay = alphay
10        self.betay = betay
11        delta = reward(T, betax, betay)
12        gamma = 1 - delta
13        self.gamma = gamma
14        self.delta = delta
15        self.position = position # new -- I need it for sound
```

In [11]:

```
1 # arbitrary number of robots that, at the start, are uniformly distributed in the
2 # centered in starting_cluster_coord
3 #
4 num_of_robots = 10
5 radius = 0.1
6 starting_cluster_coord = (0.6, 0.6)
7 # starting_cluster_coord = (0.2, 0.5)
8
9 a_x, a_y = 1-starting_cluster_coord[0]-radius, 1-starting_cluster_coord[0]+radius
10 b_x, b_y = 1-starting_cluster_coord[1]-radius, 1-starting_cluster_coord[1]+radius
11
12 for i in range(num_of_robots):
13     x = random.uniform(a_x,a_y)
14     y = random.uniform(b_x,b_y)
15     Robotx('R'+str(i), x, 1-x, y, 1-y, 1 - reward(T, 1-x, 1-y), reward(T, 1-x, 1-y))
```

In [12]:

```
1 # note: values are stored with full precision, rounding is done only on visualization
2
3 for k in Robotx._registry:
4     print(f"{k.name} {k.betax:.2f} {k.betay:.2f} {k.gamma:.2f} {k.delta:.2f} {k.position}")
```

```
R0 0.66 0.68 0.30 0.70 1
R1 0.68 0.52 0.22 0.78 2
R2 0.57 0.65 0.36 0.64 3
R3 0.58 0.69 0.38 0.62 4
R4 0.67 0.66 0.28 0.72 5
R5 0.53 0.68 0.42 0.58 6
R6 0.60 0.51 0.30 0.70 7
R7 0.67 0.51 0.23 0.77 8
R8 0.64 0.64 0.29 0.71 9
R9 0.52 0.55 0.39 0.61 10
```

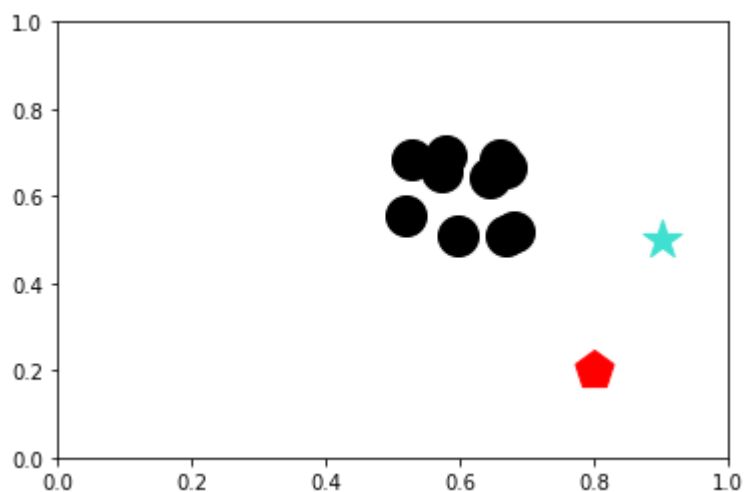
In [13]:

```
1 for k in Robotx._registry:  
2     print(f"{k.name} {k.delta:.2f}")
```

R0 0.70
R1 0.78
R2 0.64
R3 0.62
R4 0.72
R5 0.58
R6 0.70
R7 0.77
R8 0.71
R9 0.61

In [14]:

```
1 def plot_scatterplot():  
2     for i in Robotx._registry:  
3         plt.scatter(i.betax, i.betay, s = 400, marker = 'o', color = 'black')  
4  
5     plt.scatter(T.x, T.y, s = 400, marker = '*', color = 'turquoise')  
6     plt.scatter(O.x, O.y, s = 400, marker = 'p', color = 'red')  
7  
8     plt.axis([0, 1, 0, 1])  
9  
10    plt.show()  
11  
12 plot_scatterplot()
```



In [15]:

```

1  # initialization of sound parameters
2
3
4  # we need 'append' to create such a list!
5
6  l = []
7  for x in range(11):
8      value = AudioSegment.from_file("notes_/tC.mp3")
9      l.append(value)
10  for i in range(11):
11      print(l[i])
12
13  for k in Robotx._registry:
14      print(k.position)
15
16  for k in Robotx._registry:
17      print(l[k.position])

```

```

<pydub.audio_segment.AudioSegment object at 0x7fe4d0f635e0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63be0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63b20>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63b50>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63580>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f636d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f639a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1521730>
<pydub.audio_segment.AudioSegment object at 0x7fe4d15212e0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d15216d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274910>

```

```

1
2
3
4
5
6
7
8
9
10

```

```

<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63be0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63b20>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63b50>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f63580>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f636d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f639a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1521730>
<pydub.audio_segment.AudioSegment object at 0x7fe4d15212e0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d15216d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274910>

```

In [16]:

```

1  # audio block #1
2
3  # audio 1
4
5  # we can define "audio" as an attribute... no, better not.
6
7  audio = []
8
9  for x in range(11): # it should be between 1 and 11
10     valuex = AudioSegment.from_file("notes_/tC.mp3")
11     audio.append(valuex)
12  for i in range(11):
13     print(audio[i]) # at this stage, they are supposed to all give tC.mp3
14
15  for i in Robotx._registry:
16     if (i.betax == 0):
17         if (i.betay == 0.5):
18             valuex = AudioSegment.from_file("notes_/tC.mp3") # i.audio
19             audio.append(valuex)
20             print("tC")
21     if (i.betax > 0 and i.betax <= 0.17):
22         if (i.betay < 0.5):
23             valuex = AudioSegment.from_file("notes_/tB.mp3")
24             audio.append(valuex)
25             print("tB")
26         if (i.betay >= 0.5):
27             valuex = AudioSegment.from_file("notes_/tC#.mp3")
28             audio.append(valuex)
29             print("tC#")
30     if (i.betax > 0.17 and i.betax <= 0.3):
31         if (i.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
32             valuex = AudioSegment.from_file("notes_/tA#.mp3")
33             audio.append(valuex)
34             print("tA#")
35         if (i.betay >= 0.5):
36             valuex = AudioSegment.from_file("notes_/tD.mp3")
37             audio.append(valuex)
38             print("tD")
39     if (i.betax > 0.3 and i.betax <= 0.5):
40         if (i.betay < 0.5): # (R1.betay == 1):
41             valuex = AudioSegment.from_file("notes_/tD#.mp3")
42             audio.append(valuex)
43             print("tD#")
44         if (i.betay >= 0.5):
45             valuex = AudioSegment.from_file("notes_/tA.mp3")
46             audio.append(valuex)
47             print("tA")
48     if (i.betax > 0.5 and i.betax <= 0.64):
49         if (i.betay < 0.5):
50             valuex = AudioSegment.from_file("notes_/tE.mp3")
51             audio.append(valuex)
52             print("tE")
53         if (i.betay >= 0.5):
54             valuex = AudioSegment.from_file("notes_/tG#.mp3")
55             audio.append(valuex)
56             print("tG#")
57     if (i.betax > 0.64 and i.betax <= 0.84):
58         if (i.betay < 0.5):
59             valuex = AudioSegment.from_file("notes_/tF.mp3")

```

```

60         audio.append(valuex)
61         print("tF")
62     if (i.betay >= 0.5):
63         valuex = AudioSegment.from_file("notes_/tG.mp3")
64         audio.append(valuex)
65         print("tG")
66     if (i.betax > 0.84 and i.betax <= 1):
67         #if (R1.betay == 0.5):
68         valuex = AudioSegment.from_file("notes_/tF#.mp3")
69         audio.append(valuex)
70         print("tF#")
71
72
73
74 for i in Robotx._registry:
75     print(audio[i.position]) # at this stage, they are supposed to all give tC.
76
77
78
79
80 mix = []
81
82 for s in range(11): # it should be between 1 and 11
83     #values = (audio[s].overlay(audio[s+1])).overlay(audio[s+3])
84
85     # is there a more synthetic way to write this??
86     values = audio[s].overlay(audio[s+1])
87     values2 = values.overlay(audio[s+2])
88     values3 = values2.overlay(audio[s+3])
89     values4 = values3.overlay(audio[s+4])
90     values5 = values4.overlay(audio[s+5])
91     values6 = values5.overlay(audio[s+6])
92     values7 = values6.overlay(audio[s+7])
93     values8 = values7.overlay(audio[s+8])
94     values9 = values8.overlay(audio[s+9])
95     mix.append(values9)
96     print(mix[s])
97
98 mix[10].export("notes_/10_robot_sound/mixed_time_1.mp3", format='mp3') # export
99 play(mix[10])
100

```

```

<pydub.audio_segment.AudioSegment object at 0x7fe4d14e4370>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163eb20>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163ebe0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163eb50>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163e2e0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163eb80>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eab80>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eabb0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea490>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ead00>
tG
tG
tG#
tG#
tG
tG#
tG#

```

```

tG
tG
tG#
<pydub.audio_segment.AudioSegment object at 0x7fe4d163eb20>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163ebe0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163eb50>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163e2e0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163eb80>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eab80>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eabb0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea490>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ead00>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163efa0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14f0400>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eaa30>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274700>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14f0640>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274550>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f4e970>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274d30>
<pydub.audio_segment.AudioSegment object at 0x7fe4d163ef40>
<pydub.audio_segment.AudioSegment object at 0x7fe4b22749a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b22741c0>
Could not import the PyAudio C module '_portaudio'.

```

```

avplay version 12.3, Copyright (c) 2003-2018 the Libav developers
  built on Nov  2 2021 03:53:01 with Apple clang version 13.0.0 (clang
-1300.0.29.3)
Failed to set value '-hide_banner' for option 'autoexit'

```

In [17]:

```

1  for r in Robotx._registry:
2      if (r.delta < 0.5):
3          print(f"{r.name} {r.delta:.2f} achtung!") # and start from this point to

```

In [18]:

```

1  # Reshuffling -----

```

In [19]:

```

1  # I'm adding this one as the only non-quantum thing:
2
3  result = all(i.delta < 0.8 for i in Robotx._registry)
4
5  # Printing result
6  print("Do all the robots have a reward lower than 0.8? : " + str(result))
7
8  # if True: reshuffle positions
9  # if False: do nothing
10
11 if result == True:
12     flag = True
13     while flag:
14         flag = False
15         for i in Robotx._registry:
16             i.alphax = np.random.uniform(0,0.9)
17             i.betax = 1 - i.alphax
18             i.alphay = np.random.uniform(0,0.9)
19             i.betay = 1 - i.alphay
20             if (i.betax - 0.x <= 0.2 and i.betay - 0.y <= 0.2 <= 0.2):
21                 flag = True

```

Do all the robots have a reward lower than 0.8? : True

In [20]:

```

1  for k in Robotx._registry:
2      print(f"{k.name} {k.betax:.2f} {k.betay:.2f} {k.gamma:.2f} {k.position}")

```

```

R0 0.44 0.53 0.30 1
R1 0.83 0.71 0.22 2
R2 0.11 0.65 0.36 3
R3 0.89 0.66 0.38 4
R4 0.87 0.68 0.28 5
R5 0.11 0.45 0.42 6
R6 0.37 0.50 0.30 7
R7 0.47 0.58 0.23 8
R8 0.35 0.73 0.29 9
R9 0.85 0.96 0.39 10

```

In [21]:

```

1  for i in Robotx._registry: # recalculate the rewards
2      i.delta = reward(T, i.betax, i.betay)
3      i.gamma = 1 - i.delta
4      print(f"{i.name} {i.delta:.2f}")

```

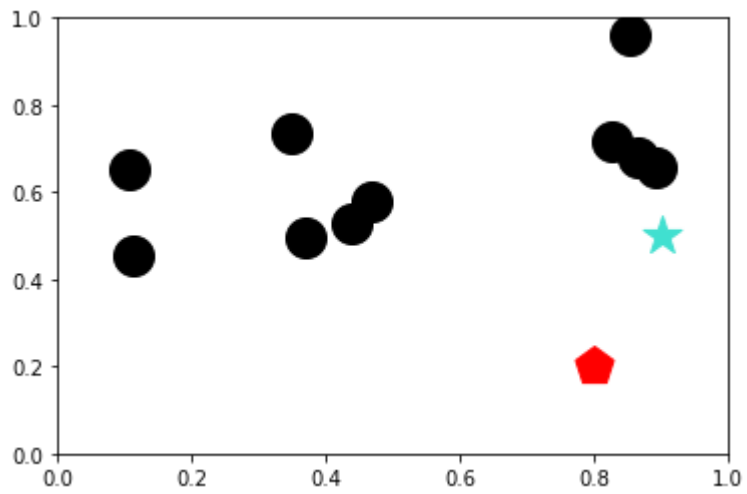
```

R0 0.54
R1 0.77
R2 0.19
R3 0.84
R4 0.82
R5 0.21
R6 0.47
R7 0.56
R8 0.40
R9 0.54

```


In [22]:

```
1 plot_scatterplot()
```



In [23]:

```

1  # audio block #2
2
3
4  # audio 2
5
6  audio = []
7
8  for x in range(11): # it should be between 1 and 11
9      valuex = AudioSegment.from_file("notes_/tC.mp3")
10     audio.append(valuex)
11  for i in range(11):
12     print(audio[i]) # at this stage, they are supposed to all give tC.mp3
13
14  for i in Robotx._registry:
15     if (i.betax == 0):
16         if (i.betay == 0.5):
17             valuex = AudioSegment.from_file("notes_/tC.mp3") # i.audio
18             audio.append(valuex)
19             print("tC")
20     if (i.betax > 0 and i.betax <= 0.17):
21         if (i.betay < 0.5):
22             valuex = AudioSegment.from_file("notes_/tB.mp3")
23             audio.append(valuex)
24             print("tB")
25         if (i.betay >= 0.5):
26             valuex = AudioSegment.from_file("notes_/tC#.mp3")
27             audio.append(valuex)
28             print("tC#")
29     if (i.betax > 0.17 and i.betax <= 0.3):
30         if (i.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
31             valuex = AudioSegment.from_file("notes_/tA#.mp3")
32             audio.append(valuex)
33             print("tA#")
34         if (i.betay >= 0.5):
35             valuex = AudioSegment.from_file("notes_/tD.mp3")
36             audio.append(valuex)
37             print("tD")
38     if (i.betax > 0.3 and i.betax <= 0.5):
39         if (i.betay < 0.5): # (R1.betay == 1):
40             valuex = AudioSegment.from_file("notes_/tD#.mp3")
41             audio.append(valuex)
42             print("tD#")
43         if (i.betay >= 0.5):
44             valuex = AudioSegment.from_file("notes_/tA.mp3")
45             audio.append(valuex)
46             print("tA")
47     if (i.betax > 0.5 and i.betax <= 0.64):
48         if (i.betay < 0.5):
49             valuex = AudioSegment.from_file("notes_/tE.mp3")
50             audio.append(valuex)
51             print("tE")
52         if (i.betay >= 0.5):
53             valuex = AudioSegment.from_file("notes_/tG#.mp3")
54             audio.append(valuex)
55             print("tG#")
56     if (i.betax > 0.64 and i.betax <= 0.84):
57         if (i.betay < 0.5):
58             valuex = AudioSegment.from_file("notes_/tF.mp3")
59             audio.append(valuex)

```

```

60         print("tF")
61     if (i.betay >= 0.5):
62         valuex = AudioSegment.from_file("notes_/tG.mp3")
63         audio.append(valuex)
64         print("tG")
65     if (i.betax > 0.84 and i.betax <= 1):
66         #if (R1.betay == 0.5):
67         valuex = AudioSegment.from_file("notes_/tF#.mp3")
68         audio.append(valuex)
69         print("tF#")
70
71
72
73 for i in Robotx._registry:
74     print(audio[i.position]) # at this stage, they are supposed to all give tC.
75
76
77
78
79 mix = []
80
81 for s in range(11): # it should be between 1 and 11
82     #values = (audio[s].overlay(audio[s+1])).overlay(audio[s+3])
83
84     # is there a more synthetic way to write this??
85     values = audio[s].overlay(audio[s+1])
86     values2 = values.overlay(audio[s+2])
87     values3 = values2.overlay(audio[s+3])
88     values4 = values3.overlay(audio[s+4])
89     values5 = values4.overlay(audio[s+5])
90     values6 = values5.overlay(audio[s+6])
91     values7 = values6.overlay(audio[s+7])
92     values8 = values7.overlay(audio[s+8])
93     values9 = values8.overlay(audio[s+9])
94     mix.append(values9)
95     print(mix[s])
96
97 mix[10].export("notes_/10_robot_sound/mixed_time_2.mp3", format='mp3') # export
98 play(mix[10])
99
100 # I'm trying to use the same code, but saving the file as another one.

```

```

<pydub.audio_segment.AudioSegment object at 0x7fe4b22633a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2263970>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ead00>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274e20>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eabe0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea6d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea910>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eadc0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea220>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eaa00>

```

```

tA
tG
tC#
tF#
tF#
tB
tD#
tA

```

```

tA
tF#
<pydub.audio_segment.AudioSegment object at 0x7fe4b2263970>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ead00>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274e20>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eabe0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea6d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea910>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eadc0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea220>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eaa00>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eab80>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eabb0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea490>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2263820>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2263460>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2263be0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1521370>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea130>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1521460>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14e4370>
Could not import the PyAudio C module '_portaudio'.

```

```

avplay version 12.3, Copyright (c) 2003-2018 the Libav developers
  built on Nov  2 2021 03:53:01 with Apple clang version 13.0.0 (clang
-1300.0.29.3)
Failed to set value '-hide_banner' for option 'autoexit'

```

In [24]:

```
1 # Quantum circuit construction -----
```

In [25]:

```

1 q = QuantumRegister(5, 'q') # qubits # changed to 9, formerly 15
2 m2 = ClassicalRegister(1, 'c1') # classical bits (separated is better)
3 m3 = ClassicalRegister(1, 'c2')
4 m4 = ClassicalRegister(1, 'c3')
5
6 qc3 = QuantumCircuit(q, m2, m3, m4) # to reach the target
7 qc4 = QuantumCircuit(q, m2, m3, m4) # to get back to the nest

```

In [26]:

```
1 # Which robot should enter the gate? -----
```

In [27]:

```

1 def print_formatted_vector(*args):
2     for vector in args:
3         print("[" + "".join(f"{val:.2f} " for val in vector).strip() + "]")

```

In [28]:

```

1  # in case of ties on delta score, the max() function outputs the first maximum
2  closest_robot = max(Robotx._registry, key=attrgetter('delta'))
3  print(f"Closest robot to the target: {closest_robot.name} {closest_robot.betax:."
4
5  # and then it enters the gate
6  vector0 = [closest_robot.alphax, closest_robot.betax]
7  vector1 = [closest_robot.alphay, closest_robot.betay]
8  vector3 = [closest_robot.gamma, closest_robot.delta]
9
10 normalized_v0 = vector0/np.linalg.norm(vector0)
11 normalized_v1 = vector1/np.linalg.norm(vector1)
12 normalized_v3 = vector3/np.linalg.norm(vector3)
13
14 print_formatted_vector(vector0, vector1, vector3)
15 print_formatted_vector(normalized_v0, normalized_v1, normalized_v3)

```

```

Closest robot to the target: R3 0.89 0.66 0.84
[0.11 0.89]
[0.34 0.66]
[0.16 0.84]
[0.12 0.99]
[0.47 0.88]
[0.18 0.98]

```

In [29]:

```

1  # Setting up |q_0> -----

```

In [30]:

```

1  # direct initialization with amplitudes vector
2  qc3.initialize(normalized_v0, q[0])
3  qc3.initialize(normalized_v1, q[1])
4  qc3.initialize(normalized_v3, q[2])

```

Out[30]:

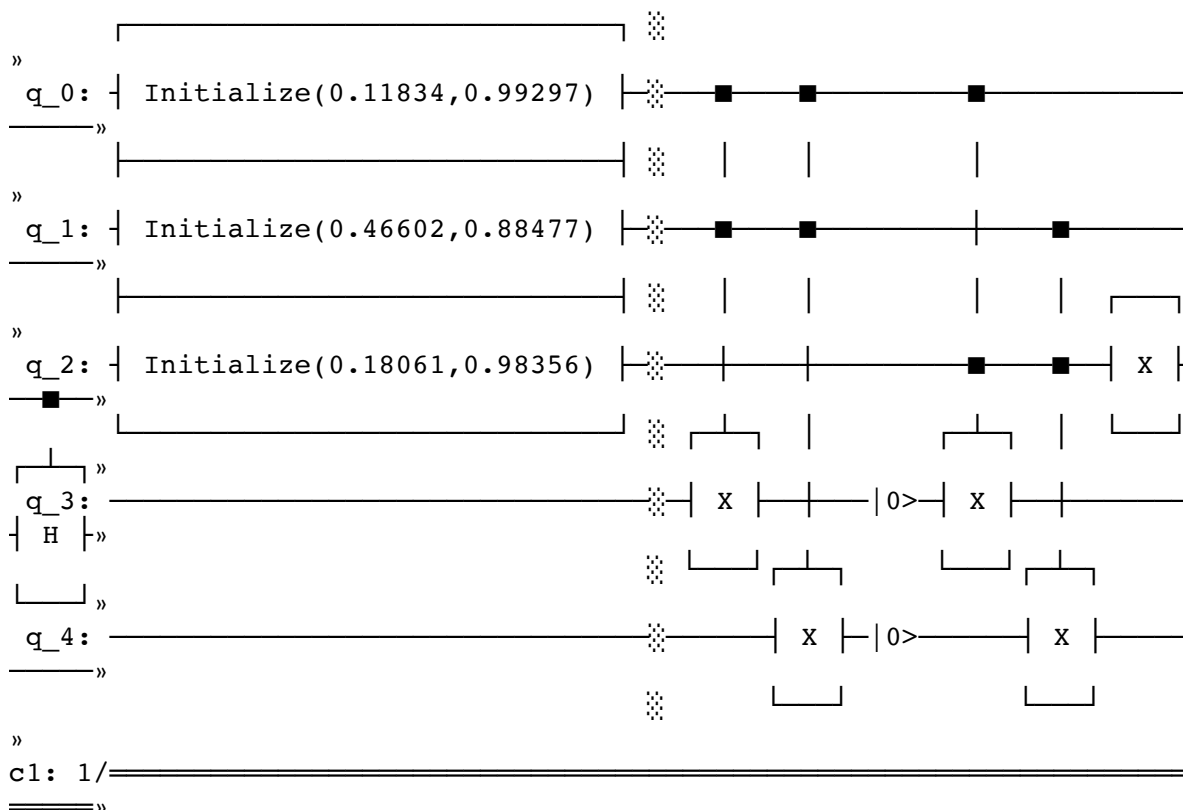
```
<qiskit.circuit.instructionset.InstructionSet at 0x7fe480023ec0>
```

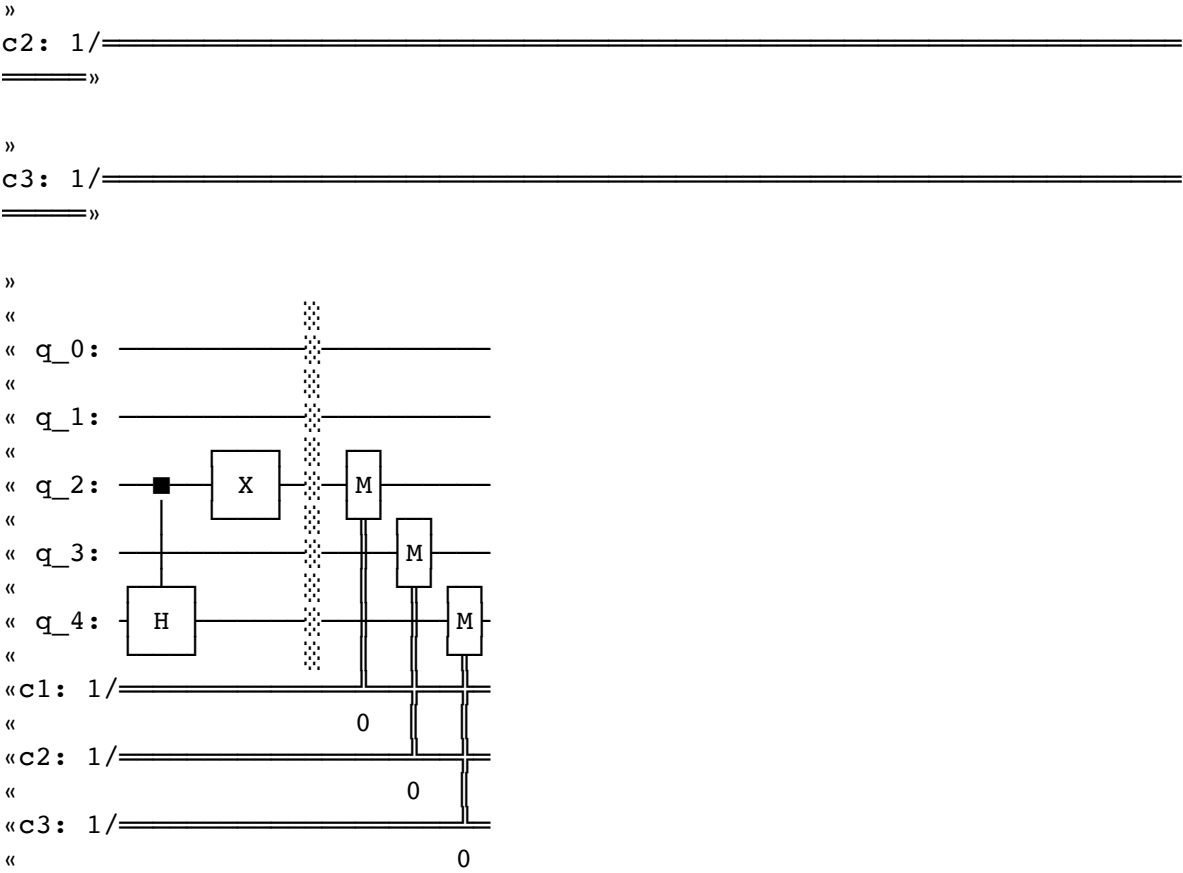
In [31]:

```

1  # this is the core code, and it is unchanged across time
2
3  qc3.barrier(q)
4  qc3.ccx(q[0],q[1],q[3])
5  qc3.ccx(q[0],q[1],q[4])
6
7  qc3.reset(q[3])
8  qc3.reset(q[4])
9
10 qc3.ccx(q[0],q[2],q[3])
11 qc3.ccx(q[1],q[2],q[4])
12
13 qc3.x(q[2])
14
15 qc3.ch(q[2],q[3])
16 qc3.ch(q[2],q[4])
17
18 qc3.x(q[2])
19
20 qc3.barrier(q)
21
22 # perform measurements and store them in classical bits
23
24 qc3.measure(q[2],m2[0])
25 qc3.measure(q[3],m3[0])
26 qc3.measure(q[4],m4[0])
27
28 # visualization of the circuit
29
30 # qc3.draw(fold=-1, output="mpl")
31 # plt.show();
32
33 print(qc3)

```





In [32]:

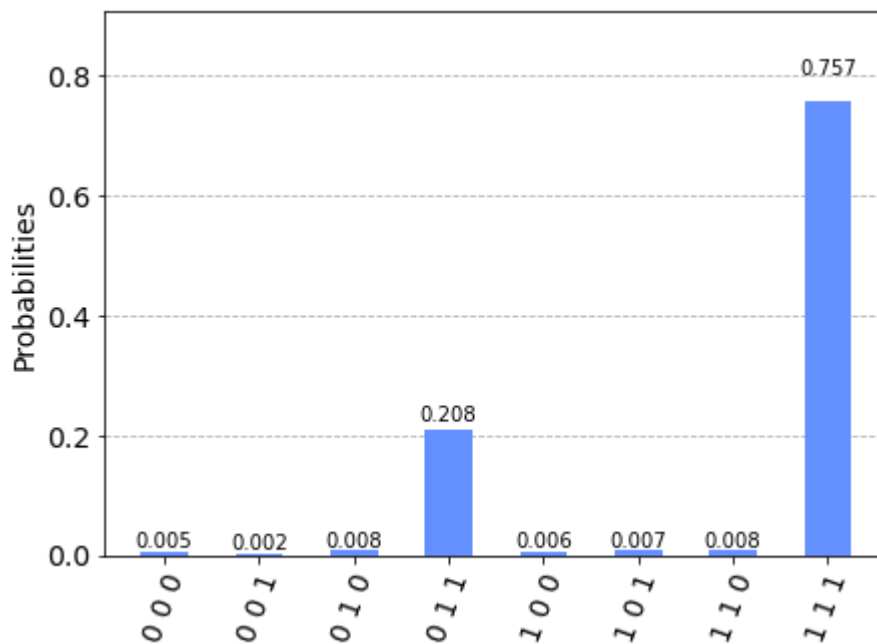
```

1 aer_sim = Aer.get_backend("aer_simulator")
2 transpiled_qc = transpile(qc3, aer_sim)
3 result = aer_sim.run(transpiled_qc).result()
4
5 counts = result.get_counts()
6 print("counts: ", counts)
7
8 plot_histogram(result.get_counts())

```

counts: {'1 0 1': 7, '1 1 0': 8, '1 0 0': 6, '0 0 1': 2, '0 1 0': 8, '0 0 0': 5, '0 1 1': 213, '1 1 1': 775}

Out[32]:



In [33]:

```

1 def eval_outcome(most_prob_dict, n_outcome):
2     mapped_weights0 = list(map(lambda res: int(res[n_outcome*2])*most_prob_dict[
3     return sum(mapped_weights0)/sum(most_prob_dict.values())

```


In [34]:

```

1 num_most_prob_states = 4
2
3 # https://docs.python.org/3/library/heapq.html:
4 #
5 # heapq.nlargest(n, iterable, key=None) returns a list with the n largest elements
6 #
7 most_prob_dict = dict(heapq.nlargest(num_most_prob_states, counts.items(), key=lambda x: x[1]))
8 print(f"{num_most_prob_states} most probable states: {most_prob_dict}")
9
10 outcome0, outcome1 = eval_outcome(most_prob_dict, 0), eval_outcome(most_prob_dict, 1)
11
12 print(f"outcome0: {outcome0:.2f}\noutcome1: {outcome1:.2f}")
13

```

4 most probable states: {'1 1 1': 775, '0 1 1': 213, '1 1 0': 8, '0 1 0': 8}
 outcome0: 0.78
 outcome1: 1.00

In [35]:

```

1 # Setting new positions after the gate -----

```

In [36]:

```

1 for i in Robotx._registry:
2     print(f"{i.name} {i.betax:.2f} {i.betay:.2f}")
3     if (i.delta != closest_robot.delta or all(i.delta == j.delta for j in Robotx._registry)):
4         # CHANGE: but taking into account the case where all robots have the same delta
5         # for z
6         #i.betaz = outcome0
7         # the lower this value, the closer the robot to the 0, the higher alpha
8         #i.alphaz = round(1 - i.betaz, 3)
9         # for y
10        i.betay = outcome0 # changed this
11        i.alphay = 1 - i.betay
12        # for x
13        i.betax = outcome1 # changed this
14        i.alphax = 1 - i.betax

```

R0 0.44 0.53
 R1 0.83 0.71
 R2 0.11 0.65
 R3 0.89 0.66
 R4 0.87 0.68
 R5 0.11 0.45
 R6 0.37 0.50
 R7 0.47 0.58
 R8 0.35 0.73
 R9 0.85 0.96

In [37]:

```
1 for k in Robotx._registry:
2     print(f"{k.name} {k.betax:.2f} {k.betay:.2f} {k.gamma:.2f} {k.position}")
```

```
R0 1.00 0.78 0.46 1
R1 1.00 0.78 0.23 2
R2 1.00 0.78 0.81 3
R3 0.89 0.66 0.16 4
R4 1.00 0.78 0.18 5
R5 1.00 0.78 0.79 6
R6 1.00 0.78 0.53 7
R7 1.00 0.78 0.44 8
R8 1.00 0.78 0.60 9
R9 1.00 0.78 0.46 10
```

In [38]:

```
1 # former rewards
2 for i in Robotx._registry:
3     print(f"before the gate: {i.name} {i.delta:.2f}")
```

```
before the gate: R0 0.54
before the gate: R1 0.77
before the gate: R2 0.19
before the gate: R3 0.84
before the gate: R4 0.82
before the gate: R5 0.21
before the gate: R6 0.47
before the gate: R7 0.56
before the gate: R8 0.40
before the gate: R9 0.54
```

In [39]:

```
1 # new rewards
2 for i in Robotx._registry: # recalculate the rewards
3     i.delta = reward(T, i.betax, i.betay)
4     i.gamma = 1 - i.delta
5     print(f"after the gate: {i.name} {i.delta:.2f}")
```

```
after the gate: R0 0.70
after the gate: R1 0.70
after the gate: R2 0.70
after the gate: R3 0.84
after the gate: R4 0.70
after the gate: R5 0.70
after the gate: R6 0.70
after the gate: R7 0.70
after the gate: R8 0.70
after the gate: R9 0.70
```

In [40]:

```

1  # audio block #3
2
3  # audio 3
4
5  # we can define "audio" as an attribute... no, better not.
6
7  audio = []
8
9  for x in range(11): # it should be between 1 and 11
10     valuex = AudioSegment.from_file("notes_/tC.mp3")
11     audio.append(valuex)
12  for i in range(11):
13     print(audio[i]) # at this stage, they are supposed to all give tC.mp3
14
15  for i in Robotx._registry:
16     if (i.betax == 0):
17         if (i.betay == 0.5):
18             valuex = AudioSegment.from_file("notes_/tC.mp3") # i.audio
19             audio.append(valuex)
20             print("tC")
21     if (i.betax > 0 and i.betax <= 0.17):
22         if (i.betay < 0.5):
23             valuex = AudioSegment.from_file("notes_/tB.mp3")
24             audio.append(valuex)
25             print("tB")
26         if (i.betay >= 0.5):
27             valuex = AudioSegment.from_file("notes_/tC#.mp3")
28             audio.append(valuex)
29             print("tC#")
30     if (i.betax > 0.17 and i.betax <= 0.3):
31         if (i.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
32             valuex = AudioSegment.from_file("notes_/tA#.mp3")
33             audio.append(valuex)
34             print("tA#")
35         if (i.betay >= 0.5):
36             valuex = AudioSegment.from_file("notes_/tD.mp3")
37             audio.append(valuex)
38             print("tD")
39     if (i.betax > 0.3 and i.betax <= 0.5):
40         if (i.betay < 0.5): # (R1.betay == 1):
41             valuex = AudioSegment.from_file("notes_/tD#.mp3")
42             audio.append(valuex)
43             print("tD#")
44         if (i.betay >= 0.5):
45             valuex = AudioSegment.from_file("notes_/tA.mp3")
46             audio.append(valuex)
47             print("tA")
48     if (i.betax > 0.5 and i.betax <= 0.64):
49         if (i.betay < 0.5):
50             valuex = AudioSegment.from_file("notes_/tE.mp3")
51             audio.append(valuex)
52             print("tE")
53         if (i.betay >= 0.5):
54             valuex = AudioSegment.from_file("notes_/tG#.mp3")
55             audio.append(valuex)
56             print("tG#")
57     if (i.betax > 0.64 and i.betax <= 0.84):
58         if (i.betay < 0.5):
59             valuex = AudioSegment.from_file("notes_/tF.mp3")

```

```

60         audio.append(valuex)
61         print("tF")
62     if (i.betay >= 0.5):
63         valuex = AudioSegment.from_file("notes_/tG.mp3")
64         audio.append(valuex)
65         print("tG")
66     if (i.betax > 0.84 and i.betax <= 1):
67         #if (R1.betay == 0.5):
68         valuex = AudioSegment.from_file("notes_/tF#.mp3")
69         audio.append(valuex)
70         print("tF#")
71
72
73
74 for i in Robotx._registry:
75     print(audio[i.position]) # at this stage, they are supposed to all give tC.
76
77
78
79
80 mix = []
81
82 for s in range(11): # it should be between 1 and 11
83     #values = (audio[s].overlay(audio[s+1])).overlay(audio[s+3])
84
85     # is there a more synthetic way to write this??
86     values = audio[s].overlay(audio[s+1])
87     values2 = values.overlay(audio[s+2])
88     values3 = values2.overlay(audio[s+3])
89     values4 = values3.overlay(audio[s+4])
90     values5 = values4.overlay(audio[s+5])
91     values6 = values5.overlay(audio[s+6])
92     values7 = values6.overlay(audio[s+7])
93     values8 = values7.overlay(audio[s+8])
94     values9 = values8.overlay(audio[s+9])
95     mix.append(values9)
96     print(mix[s])
97
98 mix[10].export("notes_/10_robot_sound/mixed_time_3.mp3", format='mp3') # export
99 play(mix[10])
100

```

```

<pydub.audio_segment.AudioSegment object at 0x7fe4d316edf0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f1ef10>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179880>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179fa0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179280>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179610>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179df0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179dc0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179730>
<pydub.audio_segment.AudioSegment object at 0x7fe4d31799a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d31792b0>
tF#
tF#
tF#
tF#
tF#
tF#
tF#
tF#

```

```
tF#
tF#
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f1ef10>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179880>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179fa0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179280>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179610>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179df0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179dc0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179730>
<pydub.audio_segment.AudioSegment object at 0x7fe4d31799a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d31792b0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b22634f0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d0f1e910>
<pydub.audio_segment.AudioSegment object at 0x7fe4d316ec10>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1521520>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14e4370>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2263be0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b22a4fd0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3115af0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1521220>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1521340>
Could not import the PyAudio C module '_portaudio'.
```

```
avplay version 12.3, Copyright (c) 2003-2018 the Libav developers
  built on Nov  2 2021 03:53:01 with Apple clang version 13.0.0 (clang
-1300.0.29.3)
Failed to set value '-hide_banner' for option 'autoexit'
```

In [41]:

```
1 # Reach the most successful robot -----
```

In [42]:

ot for now

```

i in Robotx._registry: # recalculate the rewards
i.delta = reward(T, i.betax, i.betay)
i.gamma = (1 - i.delta, 3)

_attr_ = max(Robotx._registry, key=attrgetter('delta'))
nt(f"max_attr_.delta: {max_attr_.delta:.2f}")

i in Robotx._registry:
if (i.delta == max_attr_.delta):
    print(f"Most successful robot: {i.name} {i.betax:.2f} {i.betay:.2f} {i.delta:.2f}")

j in Robotx._registry:
# to get other robots following it:
if (j != max_attr_): # changed here
    flag = True
    while flag:
        flag = False
        j.alphax = max_attr_.alphax + np.random.uniform(0,0.01)
        j.betax = 1 - j.alphax
        j.alphay = max_attr_.alphay + np.random.uniform(0,0.01)
        j.betay = 1 - j.alphay
        if (j.betax - 0.x <= 0.2 and j.betay - 0.y <= 0.2):
            flag = True

recalculate the rewards here:

k in Robotx._registry: # recalculate the rewards
k.delta = reward(T, k.betax, k.betay)
k.gamma = 1 - k.delta
print(f"{k.name} {k.delta:.2f}")

```

```

max_attr_.delta: 0.84
Most successful robot: R3 0.89 0.66 0.84
R0 0.85
R1 0.84
R2 0.85
R3 0.84
R4 0.85
R5 0.85
R6 0.85
R7 0.85
R8 0.85
R9 0.85

```

In [43]:

```
1 for i in Robotx._registry: # recalculate the rewards
2     i.delta = reward(T, i.betax, i.betay)
3     i.gamma = 1 - i.delta
4     print(f"{i.name} {i.delta:.2f}")
```

R0 0.85

R1 0.84

R2 0.85

R3 0.84

R4 0.85

R5 0.85

R6 0.85

R7 0.85

R8 0.85

R9 0.85

In [44]:

```
1 for k in Robotx._registry:
2     print(f"{k.name} {k.betax:.2f} {k.betay:.2f} {k.gamma:.2f} {k.delta:.2f} {k.}
```

R0 0.89 0.65 0.15 0.85 1

R1 0.89 0.65 0.16 0.84 2

R2 0.89 0.65 0.15 0.85 3

R3 0.89 0.66 0.16 0.84 4

R4 0.89 0.65 0.15 0.85 5

R5 0.89 0.65 0.15 0.85 6

R6 0.89 0.65 0.15 0.85 7

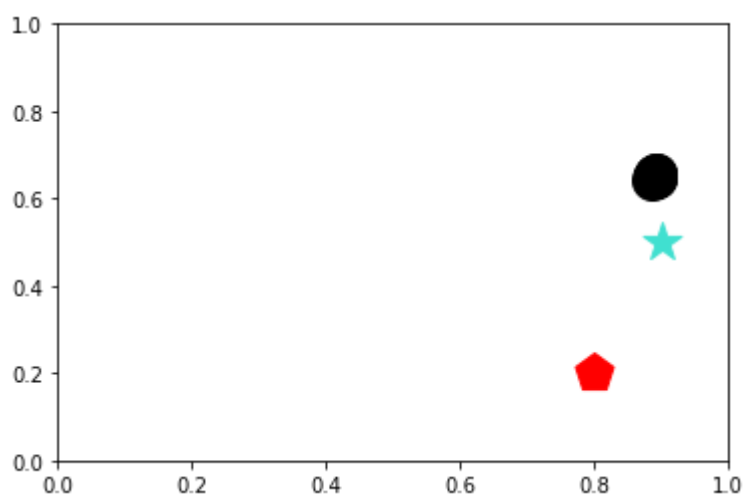
R7 0.89 0.65 0.15 0.85 8

R8 0.89 0.65 0.15 0.85 9

R9 0.89 0.65 0.15 0.85 10

In [45]:

```
1 plot_scatterplot()
```



In []:

1

In [46]:

```

1  # audio block #4
2
3  # audio 4
4
5  # we can define "audio" as an attribute... no, better not.
6
7  audio = []
8
9  for x in range(11): # it should be between 1 and 11
10     valuex = AudioSegment.from_file("notes_/tC.mp3")
11     audio.append(valuex)
12  for i in range(11):
13     print(audio[i]) # at this stage, they are supposed to all give tC.mp3
14
15  for i in Robotx._registry:
16     if (i.betax == 0):
17         if (i.betay == 0.5):
18             valuex = AudioSegment.from_file("notes_/tC.mp3") # i.audio
19             audio.append(valuex)
20             print("tC")
21     if (i.betax > 0 and i.betax <= 0.17):
22         if (i.betay < 0.5):
23             valuex = AudioSegment.from_file("notes_/tB.mp3")
24             audio.append(valuex)
25             print("tB")
26         if (i.betay >= 0.5):
27             valuex = AudioSegment.from_file("notes_/tC#.mp3")
28             audio.append(valuex)
29             print("tC#")
30     if (i.betax > 0.17 and i.betax <= 0.3):
31         if (i.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
32             valuex = AudioSegment.from_file("notes_/tA#.mp3")
33             audio.append(valuex)
34             print("tA#")
35         if (i.betay >= 0.5):
36             valuex = AudioSegment.from_file("notes_/tD.mp3")
37             audio.append(valuex)
38             print("tD")
39     if (i.betax > 0.3 and i.betax <= 0.5):
40         if (i.betay < 0.5): # (R1.betay == 1):
41             valuex = AudioSegment.from_file("notes_/tD#.mp3")
42             audio.append(valuex)
43             print("tD#")
44         if (i.betay >= 0.5):
45             valuex = AudioSegment.from_file("notes_/tA.mp3")
46             audio.append(valuex)
47             print("tA")
48     if (i.betax > 0.5 and i.betax <= 0.64):
49         if (i.betay < 0.5):
50             valuex = AudioSegment.from_file("notes_/tE.mp3")
51             audio.append(valuex)
52             print("tE")
53         if (i.betay >= 0.5):
54             valuex = AudioSegment.from_file("notes_/tG#.mp3")
55             audio.append(valuex)
56             print("tG#")
57     if (i.betax > 0.64 and i.betax <= 0.84):
58         if (i.betay < 0.5):
59             valuex = AudioSegment.from_file("notes_/tF.mp3")

```



```

60         audio.append(valuex)
61         print("tF")
62     if (i.betay >= 0.5):
63         valuex = AudioSegment.from_file("notes_/tG.mp3")
64         audio.append(valuex)
65         print("tG")
66     if (i.betax > 0.84 and i.betax <= 1):
67         #if (R1.betay == 0.5):
68         valuex = AudioSegment.from_file("notes_/tF#.mp3")
69         audio.append(valuex)
70         print("tF#")
71
72
73
74 for i in Robotx._registry:
75     print(audio[i.position]) # at this stage, they are supposed to all give tC.
76
77
78
79
80 mix = []
81
82 for s in range(11): # it should be between 1 and 11
83     #values = (audio[s].overlay(audio[s+1])).overlay(audio[s+3])
84
85     # is there a more synthetic way to write this??
86     values = audio[s].overlay(audio[s+1])
87     values2 = values.overlay(audio[s+2])
88     values3 = values2.overlay(audio[s+3])
89     values4 = values3.overlay(audio[s+4])
90     values5 = values4.overlay(audio[s+5])
91     values6 = values5.overlay(audio[s+6])
92     values7 = values6.overlay(audio[s+7])
93     values8 = values7.overlay(audio[s+8])
94     values9 = values8.overlay(audio[s+9])
95     mix.append(values9)
96     print(mix[s])
97
98 mix[10].export("notes_/10_robot_sound/mixed_time_4.mp3", format='mp3') # export
99 play(mix[10])
100

```

```

<pydub.audio_segment.AudioSegment object at 0x7fe4d316ef70>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1672fa0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1672f10>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274e20>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ead00>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea490>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eab80>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eadc0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea6d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179df0>

```

```

tF#
tF#
tF#
tF#
tF#
tF#
tF#
tF#

```

```
tF#
tF#
<pydub.audio_segment.AudioSegment object at 0x7fe4d1672fa0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d1672f10>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2274e20>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ead00>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea670>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea490>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eab80>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eadc0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14ea6d0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179df0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b22633a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179dc0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d31799a0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d3179610>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14c7520>
<pydub.audio_segment.AudioSegment object at 0x7fe4d316edf0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d14eabe0>
<pydub.audio_segment.AudioSegment object at 0x7fe4b554d850>
<pydub.audio_segment.AudioSegment object at 0x7fe4b2263be0>
<pydub.audio_segment.AudioSegment object at 0x7fe4d310b130>
<pydub.audio_segment.AudioSegment object at 0x7fe4b22a4f10>
Could not import the PyAudio C module '_portaudio'.
```

```
avplay version 12.3, Copyright (c) 2003-2018 the Libav developers
  built on Nov  2 2021 03:53:01 with Apple clang version 13.0.0 (clang
-1300.0.29.3)
Failed to set value '-hide_banner' for option 'autoexit'
```

In []:

1

In []:

1