Image for pattern recognition

```
1   3  5  7  9  3  4   4  5  6
1  20 25 24  3  5  6   4  2  4
1  22 35 24  3  5  6   4  5  7
1  20 28 34  2  5  6   4  8  9
1   3  5  7  9  3  4   4  5  6
1   3  5  7  9  3 67   4  5  6
1   3  5  7  9 78 54  94  5  6
1   3  5  7  9 99 98  54  5  6
1   3  5  7  9  3 64   4  5  6
1   3  5  7  9  3  4   4  5  6
```

Code:

```
image = [
                    [1,3,5,7,9,3,4,4,5,6],
                    [1,20,25,24,3,5,6,4,2,4],
                    [1,22,35,24,3,5,6,4,5,7],
                    [1,20,28,34,2,5,6,4,8,9],
                    [1,3,5,7,9,3,4,4,5,6],
                    [1,3,5,7,9,3,67,4,5,6],
                    [1,3,5,7,9,78,54,94,5,6],
                    [1,3,5,7,9,99,98,54,5,6],
                    [1,3,5,7,9,3,64,4,5,6],
                    [1,3,5,7,9,3,4,4,5,6],
             ]
print("Image pattern")
def print_image():
   for x in range(10):
      print(image[x][0], " ", image[x][1], " ", image[x][2], " ", image[x][3], " ",
image[x][4], " ", image[x][5], " ", image[x][6], " ", image[x][7], " ", image[x][8],
" ", image[x][9])
print_image()
```

1. Histogram
   - Code

```
i1 = 0
i2 = 0
i3 = 0
i4 = 0
i5 = 0
i6 = 0
i7 = 0
i8 = 0
i9 = 0
i20 = 0
i22 = 0
i24 = 0
```

```
i25 = 0
i28 = 0
i34 = 0
i35 = 0
i54 = 0
i64 = 0
i67 = 0
i78 = 0
i94 = 0
i98 = 0
i99 = 0
for i in range(10):
   for j in range(10):
          if image[i][j] == 1:
                 i1 += 1
          if image[i][j] == 2:
                 i2 += 1
          if image[i][j] == 3:
                 i3 += 1
          if image[i][j] == 4:
                 i4 += 1
          if image[i][j] == 5:
                 i5 += 1
          if image[i][j] == 6:
                 i6 += 1
          if image[i][j] == 7:
                 i7 += 1
          if image[i][j] == 8:
                 i8 += 1
          if image[i][j] == 9:
                 i9 += 1
          if image[i][j] == 20:
                 i20 += 1
          if image[i][j] == 22:
                 i22 += 1
          if image[i][j] == 24:
                 i24 += 1
          if image[i][j] == 25:
                 i25 += 1
          if image[i][j] == 28:
                 i28 += 1
          if image[i][j] == 34:
                 i34 += 1
          if image[i][j] == 35:
                 i35 += 1
          if image[i][j] == 54:
                 i54 += 1
          if image[i][j] == 64:
```

```
                i64 += 1
            if image[i][j] == 78:
                i78 += 1
            if image[i][j] == 94:
                i94 += 1
            if image[i][j] == 98:
                i98 += 1
            if image[i][j] == 99:
                i99 += 1
print("\nHistogram \nGray level: Number of pixels")
print("1: ", i1)
print("2: ", i2)
print("3: ", i3)
print("4: ", i4)
print("5: ", i5)
print("6: ", i6)
print("7: ", i7)
print("8: ", i8)
print("9: ", i9)
print("20: ", i20)
print("22: ", i22)
print("24: ", i24)
print("25: ", i25)
print("28: ", i28)
print("34: ", i34)
print("35: ", i35)
print("54: ", i54)
print("64: ", i64)
print("78: ", i78)
print("94: ", i94)
print("98: ", i98)
print("99: ", i99)
```

- Explanation
  The numbers represent the pixel and the output shows a list to show how many numbers of pixels there are.
- Output

```
Histogram
Gray level: Number of pixels
1:  10
2:  2
3:  14
4:  12
5:  18
6:  10
7:  8
8:  1
9:  8
20:  2
22:  1
24:  2
25:  1
28:  1
34:  1
35:  1
54:  2
64:  1
78:  1
94:  1
98:  1
99:  1
```

2. Thresholding
   - Code

```
for i in range(10):
    for j in range(10):
            if image[i][j] < 20:
                    image[i][j] = 0
            else:
                    image[i][j] = 1
print("\nThreshold")
print_image()
```

   - Explanation

     Starting from 20, there are only 2 or 1 number of pixels. Using the threshold, the image can be changed from a gray level image into a binary image.

   - Output

```
Threshold
0   0   0   0   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   1   0   0   0
0   0   0   0   0   1   1   1   0   0
0   0   0   0   0   1   1   1   0   0
0   0   0   0   0   0   1   0   0   0
0   0   0   0   0   0   0   0   0   0
```

3. Connectivity analysis

- Code

```
# First pass
first = []
second = []
label = 1
for i in range(10):
	for j in range(10):
		# If current is labeled
		if image[i][j] != 0:
			# If above or left are labeled
			if image[i][j-1] or image[i-1][j] != 0:
				# Assign the smaller value to current
				if (image[i][j-1] != 0) and (image[i][j-1] <
image[i-1][j]):
					image[i][j] = image[i][j-1]
					first.append(image[i][j])
				elif (image[i-1][j] != 0) and (image[i][j-1] >
image[i-1][j]):
					image[i][j] = image[i-1][j]
					first.append(image[i][j])
			# If above and left is 0
			elif (image[i][j-1] and image[i-1][j]) == 0:
				image[i][j] = label
				label += 1
				second.append(image[i][j])
print("\nConnectivity analysis \nFirst pass")
print_image()
# Grouping
for i in first:
	if i in second:
		first.remove(i)
print("\nGrouping \nFirst object: ", first)
print("Second object: ", second)
for i in second:
	if i in first:
		second.remove(i)
# Second pass
for i in range(10):
	for j in range(10):
		# If current is labeled
		if image[i][j] != 0:
			# Label according to group
			if image[i][j] in first:
				image[i][j] = 1
				if image[i][j-1] in second or image[i-1][j] in
second:
					image[i][j] = 2
			elif image[i][j] in second:
				image[i][j] = 2
print("\nSecond pass")
print_image()
```

- Explanation
  I used a 4-connectivity analysis where it will check the pixels above and to the left of the current pixel. First pass will put labels on the pixel and its neighbors. We group the labels by object then do a second pass to make a final label for each object.
- Output

```
Connectivity analysis
First pass
0   0   0   0   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   2   0   0   0
0   0   0   0   0   3   2   1   0   0
0   0   0   0   0   1   1   1   0   0
0   0   0   0   0   0   1   0   0   0
0   0   0   0   0   0   0   0   0   0

Grouping
First object:  [1]
Second object:  [1, 2, 3]

Second pass
0   0   0   0   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   1   1   1   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   2   0   0   0
0   0   0   0   0   2   2   2   0   0
0   0   0   0   0   2   2   2   0   0
0   0   0   0   0   0   2   0   0   0
0   0   0   0   0   0   0   0   0   0
```

4. Pattern recognition
   - Code

```
import math
...
area1 = 0
peri1 = 0
area2 = 0
peri2 = 0
for i in range(10):
   for j in range(10):
          if image[i][j] == 1:
                 area1 += 1
                 if image[i-1][j] or image[i][j-1] == 0:
                        if image[i+1][j] or image[i][j+1] != 0:
                               peri1 += 1
          if image[i][j] == 2:
                 area2 += 1
                 if image[i-1][j] or image[i][j-1] == 0:
                        if image[i+1][j] or image[i][j+1] != 0:
                               peri2 += 1
```

```
# Calculate
print("\nPattern recognition")
r1 = 16 * area1 / pow(peri1, 2)
r2 = 16 * area2 / pow(peri2, 2)
if r1 < r2:
   print("First object: Square")
   print("Second object: Circle")
else:
   print("First object: Circle")
   print("Second object: Square")
```

- Explanation

  Square will have a smaller ratio than a circle. I created two areas and perimeter for each object. Respective areas will increase if the label matches and respective perimeters will increase if it is located next to the background.

- Output