Lordick-plab-03_Exs_1+2

May 3, 2019

In [1]:
```
#Exercise 1#

#right way:
n=3
m=2
mat=[[0]*n for _ in range(m)]
mat
```

Out[1]:
```
[[0, 0, 0], [0, 0, 0]]
```

In [2]:
```
#wrong way
mat_wrong=[[0]*n]*m
mat_wrong
```

Out[2]:
```
[[0, 0, 0], [0, 0, 0]]
```

Using the * operator is while declaring 2d arrays. Using the * would create shallow lists i.e only one list object would be created and all the indices would refer to this object. This might create unwanted complications. Hence using list comprehensions is a safer way to create 2d lists. Example: Creating matrices with randomemized rows:

In [3]:
```
# what we want to have:
import random
mat=[]
for i in range(m):
    row = [random.randint(1,5)]*n
    mat.append(row) # each row is n times a random integer
mat
```

Out[3]:
```
[[4, 4, 4], [2, 2, 2]]
```

In [4]:
```
# applying to wrong alternative returns a different result:
row = [random.randint(1,5)]*n # WRONG
mat_wrong=[]
for i in range(m):
    mat_wrong.append(row)
mat_wrong # in every iteration its refers to the same row object. Resulting in similar r
```

Out[4]:
```
[[1, 1, 1], [1, 1, 1]]
```

Exercise 2

```
In [5]: import numpy as np
```

```
In [14]: seq1="abcdfefedsfdsrewe"
         seq2="ebcfgdfdfsdgdcc"
         w=3
         s=2
         dp=np.zeros((len(seq2),len(seq1)),dtype=int)
         for i in range(len(seq1)):
             for j in range(len(seq2)):
                 if seq1.lower()[i]==seq2.lower()[j]:
                     dp[j][i]=1
```

```
In [15]: dp
```

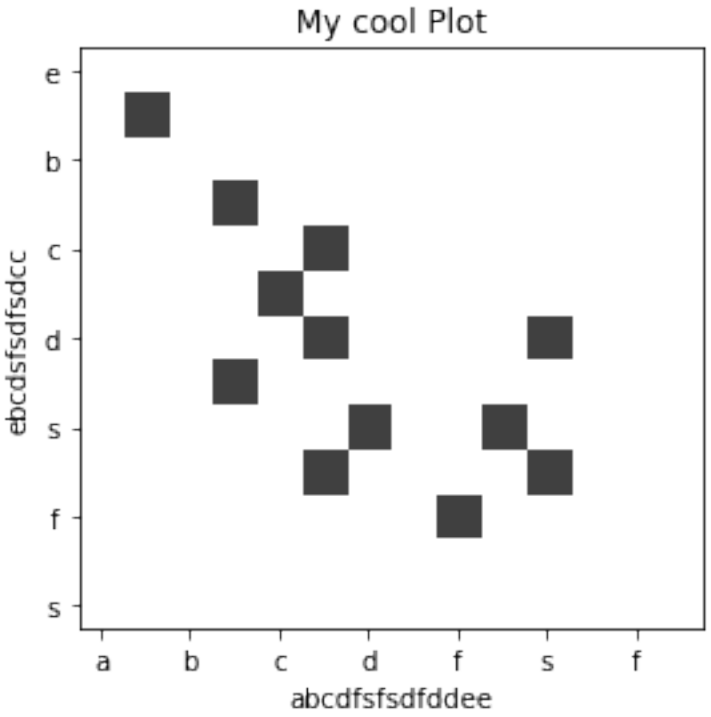
```
Out[15]: array([[0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1],
                [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0],
                [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 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, 1, 0, 0, 1, 0, 0, 0, 0, 0],
                [0, 0, 1, 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, 0, 0, 0, 0, 0, 0]])
```

a)

```
In [29]: seq1="abcdfsfsdfddee"
         seq2="ebcdsfsdfsdcc"

         def dot_plot(seq1,seq2,w=1,s=1):
             seq1=seq1.lower()
             seq2=seq2.lower()
             assert w % 2 != 0
             assert s>0
             dp=np.zeros((len(seq2),len(seq1)),dtype=int)
             for i in range(w//2,len(seq1)):
                 for j in range(w//2,len(seq2)):
                     if len((seq1[i-(w//2):i+(w//2)+1]))==len((seq2[j-(w//2):j+(w//2)+1]))==w:
                         if sum(a==b for a, b in zip(seq1[i-(w//2):i+(w//2)+1], seq2[j-(w//2):j+
                             dp[j][i]=1

             return dp
```

```
In [30]: dp=dot_plot(seq1,seq2,3,2)
         dp

Out[30]: array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 1, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0],
                [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0],
                [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])

In [26]: def dotplot2ASCII(dp,seq1,seq2,title,outputfile):
             seq1=seq1.lower()
             seq2=seq2.lower()
             with open(outputfile,"w") as out:
                 seq1_formatted=" "+seq1[:]
                 seq1_formatted_fin=" ".join(seq1_formatted)
                 out.writelines(title+"\n"+"\n")
                 out.writelines(seq1_formatted_fin+"\n")
                 dp_list=dp.tolist()
                 for i in dp_list:
                     i.insert(0,seq2[dp_list.index(i)])
                     i=[i[0]]+["*" if x==1 else " " for x in i[1:]]
                     out.writelines(" ".join(map(str, i))+"\n")

In [27]: dotplot2ASCII(dp,seq1,seq2,"My minimalistic DotPlot","Desktop/out.txt")

In [34]: import matplotlib.pyplot as plt

In [35]: def dotplot2Graphics(dp,seq1,seq2,heading,filename):
             data = np.random.random((4,4))
             fig = plt.figure()
             ax = fig.add_subplot(111)
             cax = ax.imshow(dp,cmap=plt.cm.binary,alpha=0.75)

             ax.set_xticklabels(['']+list(seq1))
             ax.set_yticklabels(['']+list(seq2))
             plt.title(heading)
             plt.xlabel(seq1)
             plt.ylabel(seq2)
             plt.savefig(filename)
             return plt.show()
```

```
In [37]: dotplot2Graphics(dp,seq1,seq2,"My cool Plot","Desktop/lol.jpeg")
```



c)

```
In [329]: #helper function to read the input fasta files.
          def fasta2dic(inputfile):
              liste=["",""]
              with open(inputfile,"r") as inputt:
                  lines=inputt.readlines()
                  for i in lines:
                      if i.startswith(">"):
                          liste[0]=liste[0]+ i.strip()
                      else:
                          liste[1]=liste[1] + i.strip()
              return liste

In [332]: #testing
          fasta2dic("Downloads/eyeless_NP_001014693.1.txt")[1]

Out[332]: 'MFTLQPTPTAIGTVVPPWSAGTLIERLPSLEDMAHKDNVIAMRNLPCLGTAGGSGLGGIAGKPSPTMEAVEASTASHPHSTSSYF

In [ ]: #!/usr/bin/python
        # the .py script. Feel free to export it as a .py script.
```

```
import sys
import numpy as np
import matplotlib.pyplot as plt

def fasta2list(inputfile):
    liste=["",""]
    with open(inputfile,"r") as inputt:
        lines=inputt.readlines()
        for i in lines:
            if i.startswith(">"):
                liste[0]=liste[0]+ i.strip()
            else:
                liste[1]=liste[1] + i.strip()
    return liste

def dotplot2Graphics(dp,seq1,seq2,filename):

    fig = plt.figure()
    ax = fig.add_subplot(111)
    cax = ax.imshow(dp,cmap=plt.cm.binary,alpha=0.75)

    ax.set_xticklabels([])
    ax.set_yticklabels([])
    plt.xlabel(seq1[0])
    plt.ylabel(seq2[0])
    plt.savefig(filename)
    plt.title("Alignment")
    return plt.show()

def dotplot2ASCII(dp,seq1,seq2,outputfile):

    with open(outputfile,"w") as out:
        seq1_formatted=" "+seq1[1][:]
        seq1_formatted_fin=" ".join(seq1_formatted)
        #out.writelines(title+"\n"+"\n")
        out.writelines(seq1_formatted_fin+"\n")
        print (seq1_formatted_fin)
        dp_list=dp.tolist()
        for i in dp_list:
            i.insert(0,seq2[1][dp_list.index(i)])
            i=[i[0]]+["*" if x==1 else " " for x in i[1:]]
            out.writelines(" ".join(map(str, i))+"\n")

def dot_plot(seq1,seq2,w=1,s=1):
    assert w % 2 != 0
    assert s>0
    dp=np.zeros((len(seq2),len(seq1)),dtype=int)
    for i in range(w//2,len(seq1)):
```

```
                for j in range(w//2,len(seq2)):
                    if len((seq1[i-(w//2):i+(w//2)+1]))==len((seq2[j-(w//2):j+(w//2)+1]))==w:
                        if sum(a==b for a, b in zip(seq1[i-(w//2):i+(w//2)+1], seq2[j-(w//2):j+(
                            dp[j][i]=1

            return dp

        print (sys.argv)
        seq1=fasta2list(sys.argv[3])
        seq2=fasta2list(sys.argv[4])
        dp=dot_plot(seq1[1],seq2[1],int(sys.argv[1]),int(sys.argv[2]))

        if sys.argv[5].endswith("fasta"):
            dotplot2Graphics(dp,seq1,seq2,sys.argv[5])

        else:
            dotplot2ASCII(dp,seq1,seq2,sys.argv[5],outputfile)
```

```
In [39]: ! python Desktop/dot.py 7 4 "Downloads/human_pax6_NM_001604.fasta" "Downloads/human_pax
```

```
['Desktop/dot.py', '7', '4', 'Downloads/human_pax6_NM_001604.fasta', 'Downloads/human_pax6_NM_00
Figure(640x480)
```

d)

```
In [42]: ! python Desktop/dot.py 23 12 "Downloads/human_pax6_NM_001604.fasta" "Downloads/mouse_p
```

```
['Desktop/dot.py', '23', '12', 'Downloads/human_pax6_NM_001604.fasta', 'Downloads/mouse_pax6_NM_
Figure(640x480)
```

```
In [ ]: ! python Desktop/dot.py 7 4 "Downloads/PAX6_HUMAN_P26367.fasta" "Downloads/eyeless_NP_00
```

0.1 Exercise 3

```
In [1]: class Graph:
            def __init__(self):
                self.nodes = ['R', 'S', 'T', 'V', 'W', 'Q', 'P', 'N', 'M', 'L', 'K', 'J', 'H', '
                self.edges = {
                    'R': ['S', 'W', 'Q'],
                    'S': ['R', 'T', 'N'],
                    'T': ['S', 'L', 'V'],
                    'V': ['T', 'J', 'W'],
                    'W': ['V', 'X', 'R'],
                    'Q': ['R', 'Z', 'P'],
                    'P': ['Q', 'N', 'C'],
                    'N': ['P', 'M', 'S'],
```

```
                'M': ['N', 'L', 'D'],
                'L': ['M', 'K', 'T'],
                'K': ['L', 'J', 'F'],
                'J': ['K', 'H', 'V'],
                'H': ['J', 'X', 'G'],
                'X': ['H', 'W', 'Z'],
                'Z': ['X', 'B', 'Q'],
                'B': ['Z', 'C', 'G'],
                'C': ['B', 'P', 'D'],
                'D': ['C', 'M', 'F'],
                'F': ['D', 'K', 'G'],
                'G': ['F', 'B', 'H']
            }
            print(f'Init graph with {len(self.nodes)} nodes')

    g = Graph()
    routes = []
    route = []

    def construct_circles(graph, routes, route, node):
        if len(route) >= 20 and node == 'R':
            route.append('R')
            routes.append(route)
            print('Route added: ', route)
            return
        elif len(route) >= 20:
            return

        route.append(node)
        for next_node in graph.edges[node]:
            cur_route = route[:]
            if next_node in cur_route and (next_node != 'R' or len(cur_route) != 20):
                continue
            construct_circles(graph, routes, cur_route, next_node)

    construct_circles(g, routes, route, 'R')
    print(f'Found {len(routes)} number of circles')
```

```
Init graph with 20 nodes
Route added:  ['R', 'S', 'T', 'L', 'M', 'N', 'P', 'Q', 'Z', 'X', 'H', 'G', 'B', 'C', 'D', 'F', '
Route added:  ['R', 'S', 'T', 'L', 'M', 'N', 'P', 'C', 'D', 'F', 'K', 'J', 'V', 'W', 'X', 'H', '
Route added:  ['R', 'S', 'T', 'L', 'K', 'J', 'V', 'W', 'X', 'H', 'G', 'F', 'D', 'M', 'N', 'P', '
Route added:  ['R', 'S', 'T', 'L', 'K', 'F', 'D', 'M', 'N', 'P', 'C', 'B', 'G', 'H', 'J', 'V', '
Route added:  ['R', 'S', 'T', 'L', 'K', 'F', 'G', 'B', 'C', 'D', 'M', 'N', 'P', 'Q', 'Z', 'X', '
Route added:  ['R', 'S', 'T', 'L', 'K', 'F', 'G', 'H', 'J', 'V', 'W', 'X', 'Z', 'B', 'C', 'D', '
Route added:  ['R', 'S', 'T', 'V', 'J', 'K', 'L', 'M', 'N', 'P', 'Q', 'Z', 'B', 'C', 'D', 'F', '
Route added:  ['R', 'S', 'T', 'V', 'J', 'H', 'G', 'B', 'C', 'D', 'F', 'K', 'L', 'M', 'N', 'P', '
Route added:  ['R', 'S', 'T', 'V', 'W', 'X', 'H', 'J', 'K', 'L', 'M', 'N', 'P', 'C', 'D', 'F', '
```

```
Route added:  ['R', 'S', 'T', 'V', 'W', 'X', 'Z', 'B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', '
Route added:  ['R', 'S', 'N', 'P', 'Q', 'Z', 'X', 'H', 'J', 'K', 'F', 'G', 'B', 'C', 'D', 'M', '
Route added:  ['R', 'S', 'N', 'P', 'Q', 'Z', 'B', 'C', 'D', 'M', 'L', 'T', 'V', 'J', 'K', 'F', '
Route added:  ['R', 'S', 'N', 'P', 'C', 'B', 'G', 'H', 'J', 'K', 'F', 'D', 'M', 'L', 'T', 'V', '
Route added:  ['R', 'S', 'N', 'P', 'C', 'D', 'M', 'L', 'T', 'V', 'W', 'X', 'H', 'J', 'K', 'F', '
Route added:  ['R', 'S', 'N', 'M', 'L', 'T', 'V', 'J', 'K', 'F', 'D', 'C', 'P', 'Q', 'Z', 'B', '
Route added:  ['R', 'S', 'N', 'M', 'L', 'T', 'V', 'W', 'X', 'Z', 'B', 'G', 'H', 'J', 'K', 'F', '
Route added:  ['R', 'S', 'N', 'M', 'D', 'C', 'P', 'Q', 'Z', 'B', 'G', 'F', 'K', 'L', 'T', 'V', '
Route added:  ['R', 'S', 'N', 'M', 'D', 'F', 'K', 'L', 'T', 'V', 'J', 'H', 'G', 'B', 'C', 'P', '
Route added:  ['R', 'S', 'N', 'M', 'D', 'F', 'G', 'B', 'C', 'P', 'Q', 'Z', 'X', 'H', 'J', 'K', '
Route added:  ['R', 'S', 'N', 'M', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'T', 'V', 'W', 'X', 'Z', '
Route added:  ['R', 'W', 'V', 'T', 'S', 'N', 'P', 'C', 'B', 'G', 'F', 'D', 'M', 'L', 'K', 'J', '
Route added:  ['R', 'W', 'V', 'T', 'S', 'N', 'M', 'L', 'K', 'J', 'H', 'X', 'Z', 'B', 'G', 'F', '
Route added:  ['R', 'W', 'V', 'T', 'L', 'M', 'D', 'C', 'B', 'G', 'F', 'K', 'J', 'H', 'X', 'Z', '
Route added:  ['R', 'W', 'V', 'T', 'L', 'K', 'J', 'H', 'X', 'Z', 'Q', 'P', 'C', 'B', 'G', 'F', '
Route added:  ['R', 'W', 'V', 'J', 'K', 'L', 'T', 'S', 'N', 'M', 'D', 'F', 'G', 'H', 'X', 'Z', '
Route added:  ['R', 'W', 'V', 'J', 'K', 'F', 'D', 'C', 'B', 'G', 'H', 'X', 'Z', 'Q', 'P', 'N', '
Route added:  ['R', 'W', 'V', 'J', 'K', 'F', 'D', 'M', 'L', 'T', 'S', 'N', 'P', 'C', 'B', 'G', '
Route added:  ['R', 'W', 'V', 'J', 'K', 'F', 'G', 'H', 'X', 'Z', 'B', 'C', 'D', 'M', 'L', 'T', '
Route added:  ['R', 'W', 'V', 'J', 'H', 'X', 'Z', 'B', 'G', 'F', 'K', 'L', 'T', 'S', 'N', 'M', '
Route added:  ['R', 'W', 'V', 'J', 'H', 'X', 'Z', 'Q', 'P', 'N', 'M', 'D', 'C', 'B', 'G', 'F', '
Route added:  ['R', 'W', 'X', 'H', 'J', 'V', 'T', 'S', 'N', 'P', 'C', 'D', 'M', 'L', 'K', 'F', '
Route added:  ['R', 'W', 'X', 'H', 'J', 'V', 'T', 'L', 'K', 'F', 'G', 'B', 'Z', 'Q', 'P', 'C', '
Route added:  ['R', 'W', 'X', 'H', 'G', 'F', 'D', 'C', 'B', 'Z', 'Q', 'P', 'N', 'M', 'L', 'K', '
Route added:  ['R', 'W', 'X', 'H', 'G', 'F', 'D', 'M', 'L', 'K', 'J', 'V', 'T', 'S', 'N', 'P', '
Route added:  ['R', 'W', 'X', 'H', 'G', 'F', 'K', 'J', 'V', 'T', 'L', 'M', 'D', 'C', 'B', 'Z', '
Route added:  ['R', 'W', 'X', 'H', 'G', 'B', 'Z', 'Q', 'P', 'C', 'D', 'F', 'K', 'J', 'V', 'T', '
Route added:  ['R', 'W', 'X', 'Z', 'B', 'C', 'D', 'M', 'L', 'K', 'F', 'G', 'H', 'J', 'V', 'T', '
Route added:  ['R', 'W', 'X', 'Z', 'B', 'G', 'H', 'J', 'V', 'T', 'S', 'N', 'M', 'L', 'K', 'F', '
Route added:  ['R', 'W', 'X', 'Z', 'Q', 'P', 'N', 'M', 'L', 'K', 'F', 'D', 'C', 'B', 'G', 'H', '
Route added:  ['R', 'W', 'X', 'Z', 'Q', 'P', 'C', 'B', 'G', 'H', 'J', 'V', 'T', 'L', 'K', 'F', '
Route added:  ['R', 'Q', 'Z', 'X', 'H', 'J', 'K', 'L', 'M', 'D', 'F', 'G', 'B', 'C', 'P', 'N', '
Route added:  ['R', 'Q', 'Z', 'X', 'H', 'G', 'B', 'C', 'P', 'N', 'S', 'T', 'L', 'M', 'D', 'F', '
Route added:  ['R', 'Q', 'Z', 'X', 'W', 'V', 'T', 'L', 'M', 'D', 'F', 'K', 'J', 'H', 'G', 'B', '
Route added:  ['R', 'Q', 'Z', 'X', 'W', 'V', 'J', 'H', 'G', 'B', 'C', 'P', 'N', 'M', 'D', 'F', '
Route added:  ['R', 'Q', 'Z', 'B', 'C', 'P', 'N', 'M', 'D', 'F', 'G', 'H', 'X', 'W', 'V', 'J', '
Route added:  ['R', 'Q', 'Z', 'B', 'C', 'P', 'N', 'S', 'T', 'V', 'J', 'K', 'L', 'M', 'D', 'F', '
Route added:  ['R', 'Q', 'Z', 'B', 'G', 'F', 'D', 'C', 'P', 'N', 'M', 'L', 'K', 'J', 'H', 'X', '
Route added:  ['R', 'Q', 'Z', 'B', 'G', 'F', 'K', 'L', 'M', 'D', 'C', 'P', 'N', 'S', 'T', 'V', '
Route added:  ['R', 'Q', 'Z', 'B', 'G', 'F', 'K', 'J', 'H', 'X', 'W', 'V', 'T', 'L', 'M', 'D', '
Route added:  ['R', 'Q', 'Z', 'B', 'G', 'H', 'X', 'W', 'V', 'J', 'K', 'F', 'D', 'C', 'P', 'N', '
Route added:  ['R', 'Q', 'P', 'N', 'M', 'L', 'K', 'J', 'H', 'G', 'F', 'D', 'C', 'B', 'Z', 'X', '
Route added:  ['R', 'Q', 'P', 'N', 'M', 'D', 'C', 'B', 'Z', 'X', 'W', 'V', 'J', 'H', 'G', 'F', '
Route added:  ['R', 'Q', 'P', 'N', 'S', 'T', 'L', 'M', 'D', 'C', 'B', 'Z', 'X', 'H', 'G', 'F', '
Route added:  ['R', 'Q', 'P', 'N', 'S', 'T', 'V', 'J', 'H', 'G', 'F', 'K', 'L', 'M', 'D', 'C', '
Route added:  ['R', 'Q', 'P', 'C', 'B', 'Z', 'X', 'H', 'G', 'F', 'D', 'M', 'N', 'S', 'T', 'L', '
Route added:  ['R', 'Q', 'P', 'C', 'B', 'Z', 'X', 'W', 'V', 'T', 'L', 'K', 'J', 'H', 'G', 'F', '
Route added:  ['R', 'Q', 'P', 'C', 'D', 'M', 'N', 'S', 'T', 'L', 'K', 'F', 'G', 'B', 'Z', 'X', '
```

```
Route added:  ['R', 'Q', 'P', 'C', 'D', 'F', 'K', 'L', 'M', 'N', 'S', 'T', 'V', 'J', 'H', 'G', '
Route added:  ['R', 'Q', 'P', 'C', 'D', 'F', 'K', 'J', 'H', 'G', 'B', 'Z', 'X', 'W', 'V', 'T', '
Route added:  ['R', 'Q', 'P', 'C', 'D', 'F', 'G', 'B', 'Z', 'X', 'H', 'J', 'K', 'L', 'M', 'N', '
Found 60 number of circles
```

0.2 Exercise 4

```
In [2]: from itertools import combinations_with_replacement
        import itertools

        import numpy as np
        from itertools import product
        count = 0
        y1 = np.arange(101)
        y2,y3,y4,y5,y6 = y1[0:50],y1[0:20],y1[0:10],y1[0:5],y1[0:2]
        a,b,c,d,e,f =y1*1,y2*2,y3*5,y4*10,y5*20,y6*50
        x = list(itertools.product(a,b,c,d,e,f))
        for each in x:
            if sum(each) == 100: count +=1
        print(f'There are {count} was to combine the coins to get 100')
```

```
There are 4557 was to combine the coins to get 100
```

```
In [ ]:
```