Download (right-click, save target as ...) this page as a jupyterlab notebook from: Lab5-HW

Exercise Set 5: Sequence, Selection, and Repetition - Oh My!

Medrano, Giovanni

R11521018

ENGR 1330 ES-5 - Homework

In [1]:

```
# Preamble script block to identify host, user, and kernel
import sys
! hostname
! whoami
print(sys.executable)
print(sys.version)
print(sys.version_info)
```

```
DESKTOP-6HAS1BN
desktop-6has1bn\medra
C:\Users\medra\anaconda3\python.exe
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
sys.version_info(major=3, minor=8, micro=5, releaselevel='final', serial=0)
```

Exercise 1 : Find the Treasure Part 1

Consider the structure below (a treasure map)

```
+-------------------------+
¦ 34 ¦ 21 ¦ 32 ¦ 41 ¦ 25  ¦
+----+----+----+----+-----¦
¦ 14 ¦ 42 ¦ 43 ¦ 14 ¦ 31  ¦
+----+----+----+----+-----¦
¦ 54 ¦ 45 ¦ 52 ¦ 42 ¦ 23  ¦
+----+----+----+----+-----¦
¦ 33 ¦ 15 ¦ 51 ¦ 31 ¦ 35  ¦
+----+----+----+----+-----¦
¦ 21 ¦ 52 ¦ 33 ¦ 13 ¦ 23  ¦
+-------------------------+
```

In this problem you are to write a program to explore the above array for a treasure. The values in the array are clues. Each cell contains an integer between 11 and 55; for each value the ten's digit represents the row number and the unit's digit represents the column number of the cell containing the next clue. Starting in the upper left corner (at 1,1), use the clues to guide your search of the array. (The first three clues are 11, 34, 42). The treasure is a cell whose value is the same as its coordinates. Your program must first read in the treasure map data into a 5 by 5 array. Your program should output the cells it visits during its search, and a message indicating where you found the treasure.

The "Treasure Hunt Problem" is from the HackerRank.com avaiable at https://www.hackerrank.com/contests/startatastartup/challenges/treasure-hunt

Now for your problem we have not yet learned how to read from files, its a small problem so lets just construct the map manually by a sequence of expressions; Your colleague got it started, but three of the rows are duplicates of the second row; repair the script and demonstrate that the repair correctly prints the treasure map.

In [2]:

```
treasuremap = [] # empty list to store row, column information
treasuremap.append([34,21,32,41,25]) # first row of the map
treasuremap.append([14,42,43,14,31]) # second row of the map
treasuremap.append([54,45,52,42,23])# third row of the map
treasuremap.append([33,15,51,31,35])# fourth row of the map
treasuremap.append([21,52,33,13,23])# fifth row of the map
print(treasuremap) # print the map (which is a list)
```

```
[[34, 21, 32, 41, 25], [14, 42, 43, 14, 31], [54, 45, 52, 42, 23], [33, 15, 51, 31, 35],
[21, 52, 33, 13, 23]]
```

Exercise 1 Part 2 -- Adding Some Selection

Using your corrected treasure map; script a selection statement that tests if a cell contains treasure (does the cell value agree with the row and column index), but it needs useful messages - **add the messages**.

Test your script using:

- Case 1 `row=1 column=3`
- Case 2 `row=5 column=2`

In [4]:

```
row = 1
column = 3
print(treasuremap[row-1][column-1]) # print a single element

maprowval = str(treasuremap[row-1][column-1])[0]
mapcolval = str(treasuremap[row-1][column-1])[1]
if int(maprowval) == row and int(mapcolval) == column :
    print('congratulations you found the treasure!')
    pass #comment this line out when have message
else:
    # message here for no treasure
    print('No treasure here, keep looking!')
    pass #comment this line out when have message
```

```
32
No treasure here, keep looking!
```

Exercise 1 : Part 3 - Completing the Hunt

Now you have all the parts needed to automatically find the treasure using your script from Part 2, and the scaffold below (incomplete - thats your job) insert your selection script code into the appropriate place and find the treasure. Stop the search when the treasure is found.

```
+-------------------------+
¦ 34 ¦ 21 ¦ 32 ¦ 41 ¦ 25  ¦
+----+----+----+----+-----¦
¦ 14 ¦ 42 ¦ 43 ¦ 14 ¦ 31  ¦
+----+----+----+----+-----¦
¦ 54 ¦ 45 ¦ 52 ¦ 42 ¦ 23  ¦
+----+----+----+----+-----¦
¦ 33 ¦ 15 ¦ 51 ¦ 31 ¦ 35  ¦
+----+----+----+----+-----¦
¦ 21 ¦ 52 ¦ 33 ¦ 13 ¦ 23  ¦
+-------------------------+
```

In [6]:

```
# put the correct treasure map here
########################
for i in range(0,5,1):
    print(treasuremap[i][:]) # print the map by row
########################
for i in range(0,5,1): # visit the rows
    for j in range(0,5,1): # visit the columns
        print(treasuremap[i][j]) # print the element by element, suppress when working
        row = i + 1
        column = j + 1
        maprowval = str(treasuremap[row-1][column-1])[0]
        mapcolval = str(treasuremap[row-1][column-1])[1]
        # get row and column from i and j values
        # get maprowval and mapcolval as in part 2
        if int(maprowval) == row and int(mapcolval) == column:
            print('The Treasure has been found at ({},{})'.format(row, column))
            break
        else:
            print('No Treasure has been found at ({},{})'.format(row, column))

        # test if cell is a treasure cell or not as in part 2 and issue message
```

```
[34, 21, 32, 41, 25]
[14, 42, 43, 14, 31]
[54, 45, 52, 42, 23]
[33, 15, 51, 31, 35]
[21, 52, 33, 13, 23]
34
No Treasure has been found at (1,1)
21
No Treasure has been found at (1,2)
32
No Treasure has been found at (1,3)
41
No Treasure has been found at (1,4)
25
No Treasure has been found at (1,5)
14
No Treasure has been found at (2,1)
42
No Treasure has been found at (2,2)
43
No Treasure has been found at (2,3)
```

```
14
No Treasure has been found at (2,4)
31
No Treasure has been found at (2,5)
54
No Treasure has been found at (3,1)
45
No Treasure has been found at (3,2)
52
No Treasure has been found at (3,3)
42
No Treasure has been found at (3,4)
23
No Treasure has been found at (3,5)
33
No Treasure has been found at (4,1)
15
No Treasure has been found at (4,2)
51
No Treasure has been found at (4,3)
31
No Treasure has been found at (4,4)
35
No Treasure has been found at (4,5)
21
No Treasure has been found at (5,1)
52
The Treasure has been found at (5,2)
```

Exercise 2: A Loop for Leaps!

1904 was a leap year. Create a script that prints out all the leap years from in the 20th century (1904-1999).

In [12]:

```
# A loop for leaps - some hints
# Make a script that prints all years from 1904-1999; then modify to just print the Lea
for i in range(1904,2000): # 2000 because its not inclusive
    if(i % 4 == 0 and (i % 400 == 0 or (not i % 100 == 0))):
        print(i)
```

```
1904
1908
1912
1916
1920
1924
1928
1932
1936
1940
1944
1948
1952
1956
1960
1964
1968
1972
1976
1980
```

```
1984
1988
1992
1996
```

Exercise 3: Whats your sine?

Print a table of the sines of angles (in radians) between 0 and 1.57 with steps of 0.01. The script below might help (it will need modification!)

In [15]:
```
import math # package that contains trig functions
print("     Sines     ")
print("   x   ","|"," sin(x) ")
print("--------|--------")
for i in range(0,158,1):
    x = float(i)*0.01
    print("%.3f" % x, "  |", " %.4f "  % math.sin(x)) # note the format code and the pl
```

```
     Sines     
   x    |  sin(x) 
--------|--------
0.000   |  0.0000 
0.010   |  0.0100 
0.020   |  0.0200 
0.030   |  0.0300 
0.040   |  0.0400 
0.050   |  0.0500 
0.060   |  0.0600 
0.070   |  0.0699 
0.080   |  0.0799 
0.090   |  0.0899 
0.100   |  0.0998 
0.110   |  0.1098 
0.120   |  0.1197 
0.130   |  0.1296 
0.140   |  0.1395 
0.150   |  0.1494 
0.160   |  0.1593 
0.170   |  0.1692 
0.180   |  0.1790 
0.190   |  0.1889 
0.200   |  0.1987 
0.210   |  0.2085 
0.220   |  0.2182 
0.230   |  0.2280 
0.240   |  0.2377 
0.250   |  0.2474 
0.260   |  0.2571 
0.270   |  0.2667 
0.280   |  0.2764 
0.290   |  0.2860 
0.300   |  0.2955 
0.310   |  0.3051 
0.320   |  0.3146 
0.330   |  0.3240 
0.340   |  0.3335 
0.350   |  0.3429 
0.360   |  0.3523 
0.370   |  0.3616 
0.380   |  0.3709 
```

```
0.390   |  0.3802
0.400   |  0.3894
0.410   |  0.3986
0.420   |  0.4078
0.430   |  0.4169
0.440   |  0.4259
0.450   |  0.4350
0.460   |  0.4439
0.470   |  0.4529
0.480   |  0.4618
0.490   |  0.4706
0.500   |  0.4794
0.510   |  0.4882
0.520   |  0.4969
0.530   |  0.5055
0.540   |  0.5141
0.550   |  0.5227
0.560   |  0.5312
0.570   |  0.5396
0.580   |  0.5480
0.590   |  0.5564
0.600   |  0.5646
0.610   |  0.5729
0.620   |  0.5810
0.630   |  0.5891
0.640   |  0.5972
0.650   |  0.6052
0.660   |  0.6131
0.670   |  0.6210
0.680   |  0.6288
0.690   |  0.6365
0.700   |  0.6442
0.710   |  0.6518
0.720   |  0.6594
0.730   |  0.6669
0.740   |  0.6743
0.750   |  0.6816
0.760   |  0.6889
0.770   |  0.6961
0.780   |  0.7033
0.790   |  0.7104
0.800   |  0.7174
0.810   |  0.7243
0.820   |  0.7311
0.830   |  0.7379
0.840   |  0.7446
0.850   |  0.7513
0.860   |  0.7578
0.870   |  0.7643
0.880   |  0.7707
0.890   |  0.7771
0.900   |  0.7833
0.910   |  0.7895
0.920   |  0.7956
0.930   |  0.8016
0.940   |  0.8076
0.950   |  0.8134
0.960   |  0.8192
0.970   |  0.8249
0.980   |  0.8305
0.990   |  0.8360
1.000   |  0.8415
1.010   |  0.8468
1.020   |  0.8521
1.030   |  0.8573
```

```
 1.040   |  0.8624
 1.050   |  0.8674
 1.060   |  0.8724
 1.070   |  0.8772
 1.080   |  0.8820
 1.090   |  0.8866
 1.100   |  0.8912
 1.110   |  0.8957
 1.120   |  0.9001
 1.130   |  0.9044
 1.140   |  0.9086
 1.150   |  0.9128
 1.160   |  0.9168
 1.170   |  0.9208
 1.180   |  0.9246
 1.190   |  0.9284
 1.200   |  0.9320
 1.210   |  0.9356
 1.220   |  0.9391
 1.230   |  0.9425
 1.240   |  0.9458
 1.250   |  0.9490
 1.260   |  0.9521
 1.270   |  0.9551
 1.280   |  0.9580
 1.290   |  0.9608
 1.300   |  0.9636
 1.310   |  0.9662
 1.320   |  0.9687
 1.330   |  0.9711
 1.340   |  0.9735
 1.350   |  0.9757
 1.360   |  0.9779
 1.370   |  0.9799
 1.380   |  0.9819
 1.390   |  0.9837
 1.400   |  0.9854
 1.410   |  0.9871
 1.420   |  0.9887
 1.430   |  0.9901
 1.440   |  0.9915
 1.450   |  0.9927
 1.460   |  0.9939
 1.470   |  0.9949
 1.480   |  0.9959
 1.490   |  0.9967
 1.500   |  0.9975
 1.510   |  0.9982
 1.520   |  0.9987
 1.530   |  0.9992
 1.540   |  0.9995
 1.550   |  0.9998
 1.560   |  0.9999
 1.570   |  1.0000
```

In [ ]: