week1_assessment

February 1, 2022

You will use the values of what you find in this assignment to answer questions in the quiz that follows. You may want to open this notebook to be displayed side-by-side on screen with this next quiz.

1. Write a function that inputs an integers and returns the negative

In [1]:
```
# Write your function here
def get_negative(x):
    if(x > 0):
        return -x
    else:
        return x
```

In [2]:
```
print(get_negative(5))
# Test your function with input x
x = 4
print(get_negative(4))
```

```
-5
-4
```

2. Write a function that inputs a list of integers and returns the minimum value

In [3]:
```
# Write your function here
def get_minimum(x):
    x.sort()
    return x[0]

def get_minimum_with_min(x):
    return min(x)
```

In [4]:
```
# Test your function with input lst
lst = [-3, 0, 2, 100, -1, 2]
print(get_minimum([-3, 0, 2, 100, -1, 2]))

print(get_minimum_with_min([-3, 0, 2, 100, -1, 2]))

# Create you own input list to test with
print(get_minimum([3,2,5, 0, 1]))
```

```
-3
-3
0
```

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

        # with Numpy Array
        def get_minimum_with_numpy(x):
            return np.amin(x)

In [6]: print(get_minimum_with_numpy([-3, 0, 2, 100, -1, 2]))

-3
```

Challenge problem: Write a function that take in four arguments: lst1, lst2, str1, str2, and returns a pandas DataFrame that has the first column labeled str1 and the second column labaled str2, that have values lst1 and lst2 scaled to be between 0 and 1.

For example

```
lst1 = [1, 2, 3]
lst2 = [2, 4, 5]
str1 = 'one'
str2 = 'two'

my_function(lst1, lst2, str1, str2)
```

should return a DataFrame that looks like:

	one	two
0	0	0
1	.5	.666
2	1	1

```
In [15]: lst1 = [1, 2, 3]
         lst2 = [2, 4, 5]
         str1 = 'one'
         str2 = 'two'

In [22]: def normalize_min_max(val, min_val, max_val):
             return (val - min_val) / (max_val - min_val) * (1-0) + 0

In [25]: # apply min-max normalization
         f = lambda x,y,z: normalize_min_max(x, y, z)

         lst1_normalized = [f(i, min(lst1), max(lst1)) for i in lst1]
         lst1_normalized
```

```
Out[25]: [0.0, 0.5, 1.0]

In [26]: lst2_normalized = [f(i, min(lst2), max(lst2)) for i in lst2]
         lst2_normalized

Out[26]: [0.0, 0.6666666666666666, 1.0]

In [33]: import pandas as pd
         def chalenge_function(l1, l2, str1, str2):
             df = pd.DataFrame({str1: l1, str2: l2})
             return df

In [34]: chalenge_function(lst1_normalized, lst2_normalized, str1, str2)

Out[34]:    one       two
         0  0.0  0.000000
         1  0.5  0.666667
         2  1.0  1.000000

In [35]: # test your challenge problem function
         import numpy as np

         lst1 = np.random.randint(-234, 938, 100)
         lst2 = np.random.randint(-522, 123, 100)
         str1 = 'one'
         str2 = 'alpha'

         lst1_normalized = [f(i, min(lst1), max(lst1)) for i in lst1]
         lst1_normalized

Out[35]: [0.36684303350970016,
          0.020282186948853614,
          0.9700176366843033,
          0.8377425044091711,
          0.8677248677248677,
          0.7795414462081128,
          0.4409171075837742,
          0.15961199294532627,
          0.15255731922398588,
          0.19576719576719576,
          0.18253968253968253,
          0.8562610229276896,
          0.4038800705467372,
          0.8218694885361552,
          0.9947089947089947,
          0.6067019400352733,
          0.13227513227513227,
          0.0,
          0.013227513227513227,
```

```
1.0,
0.8315696649029982,
0.845679012345679,
0.35537918871252205,
0.4506172839506173,
0.9541446208112875,
0.7045855379188712,
0.8289241622574955,
0.36067019400352734,
0.6410934744268078,
0.6834215167548501,
0.4991181657848324,
0.036155202821869487,
0.5299823633156967,
0.21075837742504408,
0.0564373897707231,
0.03880070546737213,
0.10934744268077601,
0.9708994708994709,
0.12169312169312169,
0.5520282186948854,
0.31216931216931215,
0.8827160493827161,
0.5873015873015873,
0.6340388007054674,
0.5502645502645502,
0.7619047619047619,
0.01763668430335097,
0.06525573192239859,
0.3447971781305115,
0.47883597883597884,
0.2601410934744268,
0.24426807760141092,
0.2707231040564374,
0.5149911816578483,
0.37213403880070545,
0.35714285714285715,
0.42504409171075835,
0.03527336860670194,
0.6463844797178131,
0.9444444444444444,
0.2839506172839506,
0.3492063492063492,
0.9135802469135802,
0.7310405643738977,
0.335978835978836,
0.7954144620811288,
0.8544973544973545,
```

```
 0.023809523809523808,
 0.6358024691358025,
 0.6375661375661376,
 0.09876543209876543,
 0.08465608465608465,
 0.6684303350970018,
 0.06790123456790123,
 0.9047619047619048,
 0.2751322751322751,
 0.40299823633156967,
 0.4532627865961199,
 0.7574955908289241,
 0.6155202821869489,
 0.4382716049382716,
 0.029982363315696647,
 0.716931216931217,
 0.810405643738977,
 0.16137566137566137,
 0.2222222222222222,
 0.1781305114638448,
 0.5758377425044092,
 0.7072310405643739,
 0.8827160493827161,
 0.01675485008818342,
 0.5185185185185185,
 0.5185185185185185,
 0.6657848324514991,
 0.8298059964726632,
 0.3844797178130511,
 0.5529100529100529,
 0.17724867724867724,
 0.12169312169312169,
 0.28835978835978837]
```

```
In [36]: lst2_normalized = [f(i, min(lst2), max(lst2)) for i in lst2]
         lst2_normalized
```

```
Out[36]: [0.014240506329113924,
          0.995253164556962,
          0.6993670886075949,
          0.9414556962025317,
          0.38449367088607594,
          0.6566455696202531,
          0.7025316455696202,
          0.15981012658227847,
          0.7025316455696202,
          0.19462025316455697,
          0.9177215189873418,
```

```
0.34177215189873417,
0.0189873417721519,
0.14082278481012658,
0.47943037974683544,
0.17088607594936708,
0.5506329113924051,
0.564873417721519,
0.5253164556962026,
0.3670886075949367,
0.6962025316455697,
0.5996835443037974,
0.11392405063291139,
0.8908227848101266,
0.814873417721519,
0.5474683544303798,
0.6091772151898734,
0.15348101265822786,
0.8291139240506329,
0.6012658227848101,
0.21835443037974683,
0.5427215189873418,
0.5917721518987342,
0.09018987341772151,
0.2848101265822785,
0.7389240506329114,
0.31962025316455694,
0.4272151898734177,
0.555379746835443,
0.8591772151898734,
0.5822784810126582,
0.0680379746835443,
0.19778481012658228,
0.2848101265822785,
0.7072784810126582,
0.4572784810126582,
0.43037974683544306,
0.44936708860759494,
0.35443037974683544,
0.34335443037974683,
0.7674050632911392,
0.48417721518987344,
0.564873417721519,
0.7484177215189873,
0.1360759493670886,
0.5506329113924051,
0.7183544303797469,
0.8212025316455697,
0.9367088607594937,
```

```
 0.555379746835443,
 0.40031645569620256,
 0.27531645569620256,
 0.0189873417721519,
 0.7689873417721519,
 0.02689873417721519,
 0.31170886075949367,
 0.2310126582278481,
 0.805379746835443,
 0.9920886075949367,
 0.4699367088607595,
 0.9414556962025317,
 0.0,
 0.056962025316455694,
 0.5443037974683544,
 0.7879746835443038,
 0.061708860759493674,
 0.14082278481012658,
 0.15348101265822786,
 0.47151898734177217,
 0.8322784810126582,
 0.7341772151898734,
 0.5174050632911392,
 0.2610759493670886,
 0.14082278481012658,
 0.3069620253164557,
 0.4224683544303797,
 0.8686708860759493,
 0.7357594936708861,
 0.629746835443038,
 0.4098101265822785,
 0.5886075949367089,
 0.06962025316455696,
 0.9034810126582279,
 1.0,
 0.7674050632911392,
 0.4825949367088608,
 0.03481012658227848,
 0.05221518987341772,
 0.5838607594936709,
 0.5253164556962026]
```

In [37]:
```
chalenge_function(lst1_normalized, lst2_normalized, str1, str2)
```

Out[37]:

	one	alpha
0	0.366843	0.014241
1	0.020282	0.995253
2	0.970018	0.699367

```
3   0.837743  0.941456
4   0.867725  0.384494
5   0.779541  0.656646
6   0.440917  0.702532
7   0.159612  0.159810
8   0.152557  0.702532
9   0.195767  0.194620
10  0.182540  0.917722
11  0.856261  0.341772
12  0.403880  0.018987
13  0.821869  0.140823
14  0.994709  0.479430
15  0.606702  0.170886
16  0.132275  0.550633
17  0.000000  0.564873
18  0.013228  0.525316
19  1.000000  0.367089
20  0.831570  0.696203
21  0.845679  0.599684
22  0.355379  0.113924
23  0.450617  0.890823
24  0.954145  0.814873
25  0.704586  0.547468
26  0.828924  0.609177
27  0.360670  0.153481
28  0.641093  0.829114
29  0.683422  0.601266
..       ...       ...
70  0.098765  0.941456
71  0.084656  0.000000
72  0.668430  0.056962
73  0.067901  0.544304
74  0.904762  0.787975
75  0.275132  0.061709
76  0.402998  0.140823
77  0.453263  0.153481
78  0.757496  0.471519
79  0.615520  0.832278
80  0.438272  0.734177
81  0.029982  0.517405
82  0.716931  0.261076
83  0.810406  0.140823
84  0.161376  0.306962
85  0.222222  0.422468
86  0.178131  0.868671
87  0.575838  0.735759
88  0.707231  0.629747
89  0.882716  0.409810
```

```
90  0.016755  0.588608
91  0.518519  0.069620
92  0.518519  0.903481
93  0.665785  1.000000
94  0.829806  0.767405
95  0.384480  0.482595
96  0.552910  0.034810
97  0.177249  0.052215
98  0.121693  0.583861
99  0.288360  0.525316

[100 rows x 2 columns]
```