Should probably check for correlations between only columns that are related. Ie Camera and Recording with other Camera and Recording stuff. Though proctoring beliefs and cameras might have some relationship, it is less relevant and computationally infeasible because of how many codes we have. We can try to correlate between just the survey likert responses.

In [44]:
```
def cramers_corrected_stat(confusion_matrix):
    """ calculate Cramers V statistic for categorial-categorial association.
        uses correction from Bergsma and Wicher,
        Journal of the Korean Statistical Society 42 (2013): 323-328
        Found: https://stackoverflow.com/a/39266194/1072532
    """
    chi2 = ss.chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum()
    phi2 = chi2/n
    r,k = confusion_matrix.shape
    phi2corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))
    rcorr = r - ((r-1)**2)/(n-1)
    kcorr = k - ((k-1)**2)/(n-1)
    return np.sqrt(phi2corr / min( (kcorr-1), (rcorr-1)))
```

In [45]:
```
cols = ['Age']+[c for c in data.columns if 'if' in c and 'Id' not in c and 'Used' not in c][:8]
print(data['Age'].value_counts())
likerts = data[cols].apply(lambda x : pd.factorize(x, na_sentinel=-1)[0])#+1
#Standard Chi Squared for categorical values doesnt work. Sample size too small for some categories?
#https://github.com/scipy/scipy/issues/14298
#pd.DataFrame([ss.chisquare(likerts[x].values,f_exp=likerts.values.T,axis=1)[0] for x in likerts])
```

```
18-24    82
25-34    40
35-44     9
55-64     1
Name: Age, dtype: int64
```

In [46]:
```
contingencymatrices = []
#for each column get confusion matrix with all other columns
#if matrix does not already exist ie calculate conf[i,j] if conf[j,i] DNE
# probably just check whether j > i because if i < j then its transpose has been done

for i in range (len(likerts.columns)):
    contingencymatrices.append([])
    for j in range(len(likerts.columns)):
        if i == j:
            contingencymatrices[i].append(1) #because the same column vs same column is just going to be 1
        if j > i:
            contingencymatrices[i].append(None)
        if i > j: #if i > j then redundant
            #drop rows where either cols value is -1
            temp = likerts[ (likerts[likerts.columns[i]] > -1) & (likerts[likerts.columns[j]] > -1) ]
            contingencymatrices[i].append(pd.crosstab(temp[likerts.columns[i]], temp[likerts.columns[j]]))

#c = temp.groupby(['ifComfortableProctor'], dropna=False, as_index=False).size()
#print(c)
#print(contingencymatrices[8][0])
```

Cramer's V

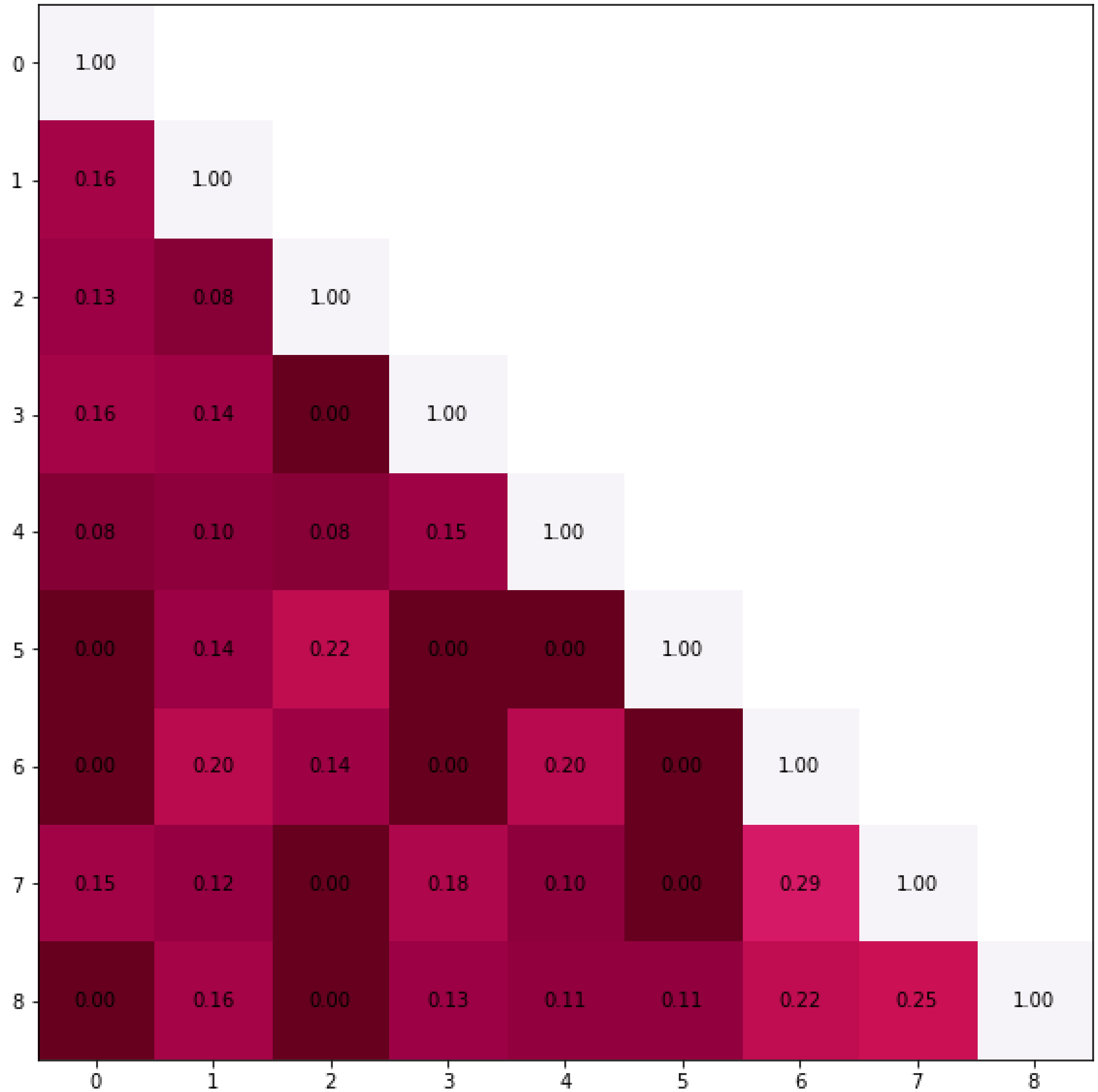
In [47]:
```python
#print(pd.show_versions())
cramersV = []
for i in range (len(contingencymatrices)):
    cramersV.append([])
    for j in contingencymatrices[i]:
        if isinstance(j, pd.DataFrame):
            cramersV[i].append(cramers_corrected_stat(j.to_numpy()))
        else:
            cramersV[i].append(j)

cramersVDf = pd.DataFrame(cramersV)
print(cramersVDf)
```

```
          0         1         2         3         4         5         6  \
0  1.000000       NaN       NaN       NaN       NaN       NaN       NaN
1  0.157447  1.000000       NaN       NaN       NaN       NaN       NaN
2  0.133681  0.083102  1.000000       NaN       NaN       NaN       NaN
3  0.158726  0.143441  0.000000  1.000000       NaN       NaN       NaN
4  0.081006  0.099969  0.083697  0.147410  1.000000       NaN       NaN
5  0.000000  0.136129  0.215745  0.000000  0.000000  1.000000       NaN
6  0.000000  0.204273  0.143827  0.000000  0.201096  0.000000  1.000000
7  0.152175  0.122432  0.000000  0.180479  0.104000  0.000000  0.291882
8  0.000000  0.159539  0.000000  0.133478  0.109498  0.112745  0.218880

          7    8
0       NaN  NaN
1       NaN  NaN
2       NaN  NaN
3       NaN  NaN
4       NaN  NaN
5       NaN  NaN
6       NaN  NaN
7  1.000000  NaN
8  0.245325  1.0
```

In [48]:
```
#fig = plt.subplots()

ifg = plt.figure()
fig, ax = plt.subplots(1,1, figsize=(5,5))
tablecolumns=[[i, likerts.columns[i]] for i in range(len(likerts.columns))]
#ax.axis('tight')
ax.axis('off')
table = ax.table(cellText=tablecolumns,loc="center", edges='open')
table.auto_set_font_size(False)
table.set_fontsize(12)
table.scale(1.5, 1.5)

plt.show()

fig, x = plt.subplots(figsize=(10,10))
x.matshow(cramersVDf, cmap='PuRd_r', vmin=0.0000000)
#fig.colorbar(x)

x_ticks = np.arange(0, len(cramersVDf), 1)
plt.xticks(x_ticks)
plt.yticks(x_ticks)
x.xaxis.set_ticks_position('bottom')

for (i, j), z in np.ndenumerate(cramersVDf):
    if not np.isnan(z):
        x.text(j, i, '{:0.2f}'.format(z), ha='center', va='center')

plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

0	Age
1	ifMandatoryCamera
2	ifLecturesRecorded
3	ifWantRemote
4	ifWantCameraOn
5	ifWantRecording
6	ifComfortableProctor
7	ifPreferProctor
8	ifBelieveFairProctor



Chi^2 Goodness of fit

In [55]:
```
likerts.corr(method='pearson', min_periods=1)
chi2 = []
p = []
df = []
for i in range (len(contingencymatrices)):
    chi2.append([])
    p.append([])
    df.append([])
    for j in contingencymatrices[i]:
        if isinstance(j, pd.DataFrame):
            chi2[i].append(ss.chi2_contingency(j)[0])
            p[i].append(ss.chi2_contingency(j)[0:2])
            df[i].append(ss.chi2_contingency(j)[2])
        else:
            chi2[i].append(None)
            p[i].append(None)
            df[i].append(None)

chi2 = pd.DataFrame(chi2)
p = pd.DataFrame(p)
df = pd.DataFrame(df)
print("chi2\n", chi2)
print("p\n",p)
print("df\n",df)
```

chi2

	0	1	2	3	4	5	6	7	8
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	None
1	18.586711	NaN	NaN	NaN	NaN	NaN	NaN	NaN	None
2	15.930291	11.720616	NaN	NaN	NaN	NaN	NaN	NaN	None
3	21.765062	19.991823	10.282245	NaN	NaN	NaN	NaN	NaN	None
4	14.611678	15.929246	14.781787	27.159064	NaN	NaN	NaN	NaN	None
5	7.852979	19.206977	29.962699	8.695610	15.433648	NaN	NaN	NaN	None
6	5.982626	22.403375	17.229439	14.970117	29.285216	14.137616	NaN	NaN	None
7	17.904903	15.828718	8.222765	26.737853	19.692791	6.701636	44.116154	NaN	None
8	9.332893	18.401512	7.071504	21.959631	20.072881	20.306532	31.893225	35.917308	None

p

	0	1	2	3	4	5	6	\
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	0.028946	NaN	NaN	NaN	NaN	NaN	NaN	
2	0.068352	0.229522	NaN	NaN	NaN	NaN	NaN	
3	0.040239	0.067241	0.591214	NaN	NaN	NaN	NaN	
4	0.263362	0.194498	0.253587	0.039749	NaN	NaN	NaN	
5	0.796511	0.083654	0.002829	0.925417	0.493136	NaN	NaN	
6	0.916955	0.033240	0.141169	0.526828	0.022082	0.588462	NaN	
7	0.118609	0.199208	0.767489	0.044490	0.234377	0.978592	0.000189	
8	0.674265	0.104032	0.852852	0.144500	0.216956	0.206731	0.010325	

df

	0	1	2	3	4	5	6	7	8
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	None
1	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	None
2	9.0	9.0	NaN	NaN	NaN	NaN	NaN	NaN	None
3	12.0	12.0	12.0	NaN	NaN	NaN	NaN	NaN	None
4	12.0	12.0	12.0	16.0	NaN	NaN	NaN	NaN	None
5	12.0	12.0	12.0	16.0	16.0	NaN	NaN	NaN	None
6	12.0	12.0	12.0	16.0	16.0	16.0	NaN	NaN	None
7	12.0	12.0	12.0	16.0	16.0	16.0	16.0	NaN	None
8	12.0	12.0	12.0	16.0	16.0	16.0	16.0	16.0	None