In [ ]:
```
import numpy as np
from scipy.stats import uniform
from scipy.stats import norm
from scipy.stats import chi2
from scipy.special import kolmogorov

def LCG(xval, M, a, c, N):
    x = np.zeros(N)
    for i in range(N):
        xval = (a*xval + c) % M
        x[i] = xval
    x/=M
    return x

def KS_test(randn):
    Ftrue = np.arange(0,1,1/len(randn))
    #import uniform distribution cdf
    F = np.sort(uniform.cdf(randn))
    #calculate max difference
    D = np.max(np.abs(Ftrue - F))
    n = len(randn)
    D *= (np.sqrt(n)+0.12+0.11/np.sqrt(n))
    #calculate p value
    pval = 1 - np.exp(-2*n*D**2)
    pval = kolmogorov(D)
    return D, pval

#calculate chi squared
def chisquare_test(randn, k):
    n = len(randn)
    p = 1/k
    test = 0
    for i in range(k):
        xval = randn[(i/k < randn)*(randn < (i+1)/k)]
        ni = len(xval)
        test += (ni - n*p)**2/(n*p)
    pval = 1 - chi2.cdf(test, k-1)
    return test, pval

def run_test_1(randn):
    #median value
    median = np.median(randn)
    #number of observations below median
    possamps = randn < median
    negsamps = randn > median
    posruns = 0
    negruns = 0
    n1 = np.sum(possamps)
    n2 = np.sum(negsamps)
    n = n1+n2
    for i in range(len(randn)):
        if i == 0:
            continue
        if possamps[i] != possamps[i-1] and possamps[i] == True:
            posruns += 1
        if negsamps[i] != negsamps[i-1] and negsamps[i] == True:
            negruns += 1
    T = posruns + negruns
```

```
    #normal cdf

    mean = 2*n1*n2/n + 1
    var = np.exp(np.log(2)+np.log(n1)+np.log(n2)+np.log(2*n1*n2 - n)-np.log(n**2
    Z = (T-mean)/np.sqrt(var)
    pval = 1 - norm.cdf(Z)
    return T, pval

def run_test_2(randn):
    # up/down
    run_lengths = []
    current_run_length = 1
    for i in range(1, len(randn)):
        if randn[i] > randn[i-1]:
            current_run_length = min(current_run_length + 1, 6)
        else:
            run_lengths.append(current_run_length)
            current_run_length = 1
    #get vector of count for each run length
    R, _ = np.histogram(run_lengths, bins=6)

    A  = np.array([
        [4529.4, 9044.9, 13568, 18091, 22615, 27892],
        [9044.9, 18097, 27139, 36187, 45234, 55789],
        [13568, 27139, 40721, 54281, 67852, 83685],
        [18091, 36187, 54281, 72414, 90470, 111580],
        [22615, 45234, 67852, 90470, 113262, 139476],
        [27892, 55789, 83685, 111580, 139476, 172860]
        ])
    B = np.array([1/6, 5/24, 11/120, 19/720, 29/5040, 1/840])
    n = len(randn)
    Z = (R-B*n).T@A@(R-B*n)/(n-6)
    #chisquare test
    pval = 1 - chi2.cdf(Z, 6)
    return Z, pval

def run_test_3(randn):
    n = len(randn)
    updown = np.zeros(n-1, dtype=bool)
    for i in range(1, n):
        updown[i-1] = randn[i] > randn[i-1]
    #count number of runs
    runs = []
    current_run = 1
    for i in range(1, n-1):
        if updown[i] != updown[i-1]:
            runs.append(current_run)
            current_run = 1
        else:
            current_run += 1
    # number of unique runs
    X = len(runs)
    Z = (X - (2*n-1)/3)/np.sqrt((16*n-29)/90)
    p = 2*(1 - norm.cdf(abs(Z)))
    return Z, p

def correlation_coefficient(randn, h=2):
    c = np.sum(randn[:-h]*randn[h:])/len(randn)
    Z = (c - 1/4)/np.sqrt(7/(144*len(randn)))
    p = 2*(1 - norm.cdf(abs(Z)))
```

```
    return c, p

def do_all_tests(randn):
    D, pval1 = KS_test(randn)
    test1, pval2 = chisquare_test(randn, len(randn)//20)
    test2, pval3 = run_test_1(randn)
    test3, pval4 = run_test_2(randn)
    test4, pval5 = run_test_3(randn)
    c, pval6 = correlation_coefficient(randn)
    # print results
    print('KS test: D =', D, 'p-value =', pval1)
    print('Chi-square test: test =', test1, 'p-value =', pval2)
    print('Run test 1: test =', test2, 'p-value =', pval3)
    print('Run test 2: test =', test3, 'p-value =', pval4)
    print('Run test 3: test =', test4, 'p-value =', pval5)
    print('Correlation coefficient: c =', c, 'p-value =', pval6)
    # dictionary of results
    results = {'KS test': (D, pval1), 'Chi-square test': (test1, pval2), 'Run te
    return results
```