Project 2, Raktim Biswas

1. Methods for Divided Differences, Coefficients, and Evaluating Newton Form of Interpolating Polynomial

```
import numpy as np

def divdiff(x, y):
    #m = len(x) let m=n+1, for some reason value of m is set to 1 instead of 21 when the method is called later even though arrays of x and y values are of length 21
    a = np.zeros([21,22]) #create table, deg n polynomial gives n+1 points and n+1 divdiffs
    #first column holds y values aka 0th divdiffs
    #note the last column holds x values so we add a column instead of creating a square matrix, this lets us calculate divdiff in forloop with ease instead
    #if first column contained x values, we would have to offset our j values by 1 in the forloop which is confusing
    a[:,0] = y
    a[:,21] = x
    #print(*a)
    for j in range(1, 21): #nested forloop calculates entries for table by column, we start with 1st divdiff and end with nth divdiff
        for i in range(j,21): #i is equal to or greater than j to fill table along and below the diagonal
            a[i,j] = ((a[i,j-1] - a[i-1,j-1])/(a[i,21] - a[i-j,21]))
    co = np.zeros(21)
    for k in range(0, 21):
        co[k] = a[k,k] #coefficients are elements of main diagnoal
    print(*a)
    return co #coefficients, if we wanted we could return the table as well but it gets messy and program gets confused when we evaluate

def newpoly(ak, xk, evalpoint):
    n = len(xk)-1
    p = ak[n]
    for k in range(1,n+1):
        p = ak[n-k] + (evalpoint - xk[n-k])*p #nested form, similar to horners, allows us to generate the polynomial recursively
        #if we consider the nested form, the nth coefficient is buried furthest in the nest so our forloop begins with the highest degree coefficient
        #by the nested form, if we evaluate at x thats relatively close to x0, we will be left with a value close to a0
    return p
```

2. Error Evaluation

a. Program Error

I observed that when the coefficients and x values were kept the same for evaluating at any given value of x, I received the same value each time; however the values did change based on the coefficients and nodes passed corresponding to uniform or Chebyshev nodes. If we consider the nested form of the interpolating polynomial, as x approaches x0, we would be multiplying by a number that approaches 0, which would leave us with a value relatively close to a0. My evaluation method seems to return the first coefficient a0, so my evaluations at 0.985 yielded significantly less error than at 0.1. This is not ideal since my forloop calculates pn(x) in its nested form recursively.

b. Evenly Spaced Nodes

Although we've discussed in class that uniform nodes provide us with larger error at the ends of the interval with interpolating polynomials, we still see a relatively accurate approximation of f(x) for x = 0.985; this is likely due to having a 20^{th} degree polynomial

c. Chebyshev Nodes

As we've discussed in class, the use of Chebyshev nodes helps mitigate the large error we observe near the bounds of our interval while evaluating interpolating polynomials. After using this program we do find that using Chebyshev nodes provides us with a slightly more accurate approximation for x = 0.985

```
for x= 0.1 f(x)= 0.8
for x= 0.985  f(x)= 0.03959513969660224
evenly spaced nodes [[-1.  -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1  0.   0.1  0.2  0.3
   0.4  0.5  0.6  0.7  0.8  0.9  1. ]]
coefficients of uniform nodes:
 [ 3.84615385e-02  8.59728507e-02  1.58371041e-01  2.86007001e-01
  5.33595151e-01  1.03775057e+00  2.00190181e+00  2.79677458e+00
 -7.54393144e+00 -1.01199080e+02 -6.43994147e+01  2.15278044e+03
 -7.26793395e+03  1.13937426e+04 -3.53842938e+03 -2.83074350e+04
  8.66915198e+04 -1.59229322e+05  2.23753623e+05 -2.60178631e+05
  2.60178631e+05]
evaluation at x= 0.1 :  0.038461538461538464 relative error (%)  95.1923076923077
evaluation at x= 0.985 :  0.038461538461538464 relative error (%)  2.862980769230765
chebyshev nodes [[ 9.97203797e-01  9.74927912e-01  9.30873749e-01  8.66025404e-01
   7.81831482e-01  6.80172738e-01  5.63320058e-01  4.33883739e-01
   2.94755174e-01  1.49042266e-01  6.12323400e-17 -1.49042266e-01
  -2.94755174e-01 -4.33883739e-01 -5.63320058e-01 -6.80172738e-01
  -7.81831482e-01 -8.66025404e-01 -9.30873749e-01 -9.74927912e-01
  -9.97203797e-01]]
coefficients of chebyshev nodes:
 [ 3.86691841e-02 -7.69933594e-02  1.19207718e-01 -1.73684669e-01
  2.56425451e-01 -4.00311244e-01  6.75451382e-01 -1.19713181e+00
  1.55126487e+00  8.17641885e+00 -6.92474217e+01 -2.97317827e+02
 -4.94176909e+02 -2.74949822e+02  6.15666801e+02  2.07014464e+03
  3.70188813e+03  5.10131293e+03  6.02813062e+03  6.44846925e+03
  6.46655104e+03]
evaluation at x= 0.1 :  0.03866918405598741 relative error (%)  95.16635199300157
evaluation at x= 0.985 :  0.03866918405598741 relative error (%)  2.3385588426002943
Press any key to continue . . .
```

```
for x= 0.0 f(x)= 1.0
for x= 1.0  f(x)= 0.038461538461538464
evenly spaced nodes [[-1.  -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1  0.   0.1  0.2  0.3
   0.4  0.5  0.6  0.7  0.8  0.9  1. ]]
coefficients of uniform nodes:
 [ 3.84615385e-02  8.59728507e-02  1.58371041e-01  2.86007001e-01
  5.33595151e-01  1.03775057e+00  2.00190181e+00  2.79677458e+00
 -7.54393144e+00 -1.01199080e+02 -6.43994147e+01  2.15278044e+03
 -7.26793395e+03  1.13937426e+04 -3.53842938e+03 -2.83074350e+04
  8.66915198e+04 -1.59229322e+05  2.23753623e+05 -2.60178631e+05
  2.60178631e+05]
evaluation at x= 0.0 :  0.038461538461538464 relative error (%)  96.15384615384616
evaluation at x= 1.0 :  0.038461538461538464 relative error (%)  0.0
chebyshev nodes [[ 9.97203797e-01  9.74927912e-01  9.30873749e-01  8.66025404e-01
   7.81831482e-01  6.80172738e-01  5.63320058e-01  4.33883739e-01
   2.94755174e-01  1.49042266e-01  6.12323400e-17 -1.49042266e-01
  -2.94755174e-01 -4.33883739e-01 -5.63320058e-01 -6.80172738e-01
  -7.81831482e-01 -8.66025404e-01 -9.30873749e-01 -9.74927912e-01
  -9.97203797e-01]]
coefficients of chebyshev nodes:
 [ 3.86691841e-02 -7.69933594e-02  1.19207718e-01 -1.73684669e-01
  2.56425451e-01 -4.00311244e-01  6.75451382e-01 -1.19713181e+00
  1.55126487e+00  8.17641885e+00 -6.92474217e+01 -2.97317827e+02
 -4.94176909e+02 -2.74949822e+02  6.15666801e+02  2.07014464e+03
  3.70188813e+03  5.10131293e+03  6.02813062e+03  6.44846925e+03
  6.46655104e+03]
evaluation at x= 0.0 :  0.03866918405598741 relative error (%)  96.13308159440126
evaluation at x= 1.0 :  0.03866918405598741 relative error (%)  0.5398785455672641
Press any key to continue . . .
```

3. References

https://pythonnumericalmethods.berkeley.edu/notebooks/chapter17.05-Newtons-Polynomial-Interpolation.html

http://www.jtrive.com/polynomial-interpolation-newtons-method.html