

TA:

Q1:

* loop through Ks (or CallPuts) and calculate values
* make sure Ks and CallPuts are of the same dimension
* Ks and CallPuts can be just one value or it can be an array or a list

Q2:

In problem 2:

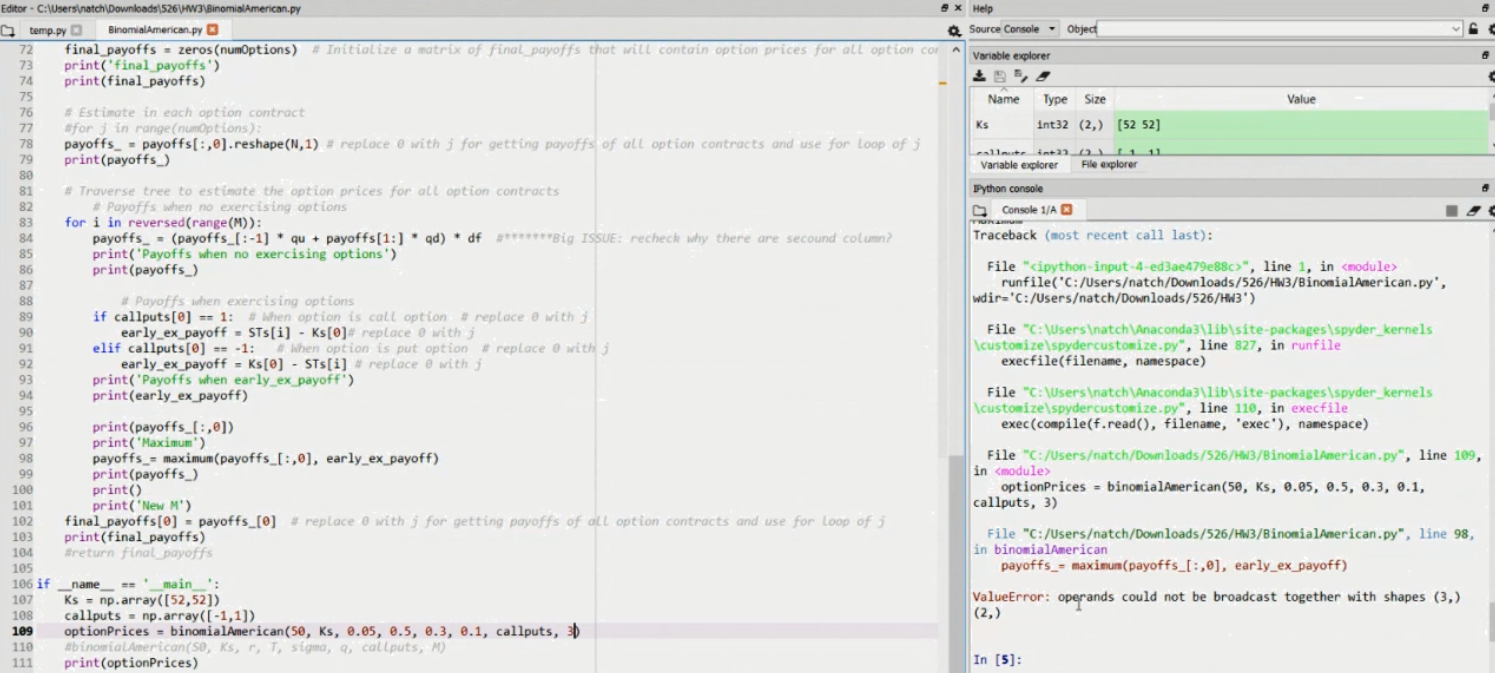
1. How to specify S\_max for test the code?

- Big value

2) How q, dividend yield, affects the equation in problem 2?

- price = bsformula(callput = callput, S0 = S0, K = K, r = r, T = T, sigma = sigma, q = q)[0]

Import bsformula?



Do the graphs in the report and console !!!!

10/15

Matthew Connelly – sked how to apply continuous dividend rate

04:40

def fdEuropean(callput, S0, K, r, T, sigma, q, M, N, S\_max):

dS = S\_max/M

dt = T/N

matval = np.zeros((M+1,N+1))

vetS = np.linspace(0,S\_max,M+1)

veti = np.array(range(M+1))

vetj = np.array(range(N+1))

if callput == 1:

matval[:,N] = np.maximum(vetS-K,0)

matval[0,:] = 0

matval[M,:] = S\_max\*np.exp(-q\*dt\*(N-vetj)) - K\*np.exp(-r\*dt\*(N-vetj))

else:

matval[:,N] = np.maximum(K-vetS,0)

matval[0,:] = K\*np.exp(-r\*dt\*(N-vetj))

matval[M,:] = 0

a = 0.5\*dt\*(sigma\*\*2\*veti - (r-q))\*veti

b = 1 - dt\*(sigma\*\*2\*veti\*\*2 + (r-q))

c = 0.5\*dt\*(sigma\*\*2\*veti + (r-q))\*veti

for j in range(N-1,-1,-1):

for i in range(1,M):

matval[i,j] = a[i]\*matval[i-1,j+1] + b[i]\*matval[i,j+1]+ \

c[i]\*matval[i+1,j+1]

return np.exp(-q\*T)\*np.interp(S0, vetS, matval[:,0])def fdEuropean(callput, S0, K, r, T, sigma, q, M, N, S\_max):

dS = S\_max/M

dt = T/N

matval = np.zeros((M+1,N+1))

vetS = np.linspace(0,S\_max,M+1)

veti = np.array(range(M+1))

vetj = np.array(range(N+1))

if callput == 1:

matval[:,N] = np.maximum(vetS-K,0)

matval[0,:] = 0

matval[M,:] = S\_max\*np.exp(-q\*dt\*(N-vetj)) - K\*np.exp(-r\*dt\*(N-vetj))

else:

matval[:,N] = np.maximum(K-vetS,0)

matval[0,:] = K\*np.exp(-r\*dt\*(N-vetj))

matval[M,:] = 0

a = 0.5\*dt\*(sigma\*\*2\*veti - (r-q))\*veti

b = 1 - dt\*(sigma\*\*2\*veti\*\*2 + (r-q))

c = 0.5\*dt\*(sigma\*\*2\*veti + (r-q))\*veti

for j in range(N-1,-1,-1):

for i in range(1,M):

matval[i,j] = a[i]\*matval[i-1,j+1] + b[i]\*matval[i,j+1]+ \

c[i]\*matval[i+1,j+1]

return np.exp(-q\*T)\*np.interp(S0, vetS, matval[:,0])