```python
"""
Textbook problem 18.17

Develop, debug, and test a program in either a high-level language or macro
language of your choice to implement Newton’s interpolating polynomial based
on Fig. 18.7.

@author: Jacob Needham
"""
#necessary libaries
import numpy as np

'''
Main fucntion takes input data from table with x and y values and runs Fdd function.
'''
def Main():
    #put numbers here
    x = []
    y = []
    n = len(x)

    Xi = 2                  #input x value to solve for here
    FddMatrix(x,y,n,Xi)

'''
FddMatrix takes in vectors x, y from table and solves for linear fit between points
'''
def FddMatrix(x,y,n,Xi):
    #initilizing Fdd matrix and placeholder matrix for y
    Fdd = np.matrix([[0 for x in range(n)] for y in range(n)],dtype = 'float')
    y_working = np.zeros((n,1), dtype = 'float')
    for i in range(n):                              #For loop coppies y value into first
        Fdd[i,0] = y[i]                             #column.
    for j in range(1,n):                            #For loop fills A matrix using modified
        for i in range(0,n-j):                      #Eq. 18.2 for linear interpolation from
            Fdd[i,j] = (Fdd[i+1,j-1] - Fdd[i,j-1])/(x[i+j]-x[i])           #the book.

    #initializing necessary place-holding matricies
    x_temp = [1]
    y_working[0] = Fdd[0,0]
    for order in range(1,n):
        x_temp.append(x_temp[order-1]*(Xi - x[order-1]))
        y_working[order] = y_working[order-1] + Fdd[0,order]*x_temp[order]

    #formating output and printing solution
    final_y = str(y_working[n-1])
    final_y = final_y.strip(' /[/]')
    print('At x =',Xi,' the interpolated y value is: ', final_y, sep = '')

Main()
```

```python
"""
Textbook problem 18.18

Test the program you developed in Prob. 18.17 by duplicating
the computation from Example 18.5.

@author: Jacob Needham
"""

#necessary libaries
import numpy as np

'''
Main fucntion takes input data from table with x and y values and runs Fdd function.
'''
def Main():
    #put numbers here
    x = [1,4,6,5,3,1.5,2.5,3.5]
    y = [0,1.3862944,1.7917595,1.6094379,1.0986123,0.4054641,0.9162907,1.2527630]
    n = len(x)

    Xi = 2
    FddMatrix(x,y,n,Xi)


'''
FddMatrix takes in vectors x, y from table and solves for linear fit between points
'''
def FddMatrix(x,y,n,Xi):
    #initilizing Fdd matrix and placeholder matrix for y
    Fdd = np.matrix([[0 for x in range(n)] for y in range(n)],dtype = 'float')
    y_working = np.zeros((n,1), dtype = 'float')
    for i in range(n):                               #for loop coppies y value into first
        Fdd[i,0] = y[i]                              #column
    for j in range(1,n):                             #for loop fills A matrix using modified
        for i in range(0,n-j):                       #Eq. 18.2 for linear interpolation from
            Fdd[i,j] = (Fdd[i+1,j-1] - Fdd[i,j-1])/(x[i+j]-x[i])       #the book

    #initializing necessary place-holding matricies
    x_temp = [1]
    y_working[0] = Fdd[0,0]
    for order in range(1,n):
        x_temp.append(x_temp[order-1]*(Xi - x[order-1]))
        y_working[order] = y_working[order-1] + Fdd[0,order]*x_temp[order]

    #formating output and printing solution
    final_y = str(y_working[n-1])
    final_y = final_y.strip(' /[/]')
    print('OUTPUT')
    print('At x=',Xi,' the interpolated y value is: ', final_y, sep = '')

Main()
```

Editor - C:\Users\Needh\Documents\Spring 2019\Numerical Methods\HW 6\18.18.py

18.17.py | 18.18.py | 18.23.py | 18.24.py | 17.17.py

```python
"""
Textbook problem 18.18

Test the program you developed in Prob. 18.17 by duplicating
the computation from Example 18.5.

@author: Jacob Needham
"""

#necessary libaries
import numpy as np

'''
Main fucntion takes input data from table with x and y values and runs Fdd function.
'''
def Main():
    #put numbers here
    x = [1,4,6,5,3,1.5,2.5,3.5]
    y = [0,1.3862944,1.7917595,1.6094379,1.0986123,0.4054641,0.9162907,1.2527630]
    n = len(x)

    Xi = 2
    FddMatrix(x,y,n,Xi)


'''
FddMatrix takes in vectors x, y from table and solves for linear fit between points
'''
def FddMatrix(x,y,n,Xi):
    #initializing Fdd matrix and placeholder matrix for y
    Fdd = np.matrix([[0 for x in range(n)] for y in range(n)],dtype = 'float')
    y_working = np.zeros((n,1), dtype = 'float')
    for i in range(n):                         #for loop coppies y value into first column
        Fdd[i,0] = y[i]
    for j in range(1,n):                       #for loop fills A matrix using modified Eq.
        for i in range(0,n-j):                 #linear interpolation from the book
            Fdd[i,j] = (Fdd[i+1,j-1] - Fdd[i,j-1])/(x[i+j]-x[i])

    #initializing necessary place-holding matricies
    x_temp = [1]
    y_working[0] = Fdd[0,0]
```

IPython console

Console 1/A

```
In [123]: runfile('C:/Users/Needh/Documents/Spring 2019/Numerical Methods/HW 6/18.18.py', wdir='C:/Users/Needh/Documents/Spring 2019/Numerical Methods/HW 6')
OUTPUT
At x=2 the interpolated y value is: 0.69343835

In [124]:
```

Fast switch between files

Permissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 50 | Column: 16 | Memory: 50 %

```python
"""
Textbook problem 18.23

Develop, debug, and test a program in either a high-level language or macro
language of your choice to implement cubic spline interpolation based on
Fig. 18.18. Test the program by duplicating
Example 18.10. (table 18.1)

@author: Jacob Needham
"""

import numpy as np

'''
Main function defines data series of points and what X is to be found and calls
the cubic spline interpolation to find corresponding Y value.
'''
def Main():
    #put data here:
    x = np.matrix([3,4.5,7,9], dtype = 'float')
    y = np.matrix([2.5,1,2.5,.5], dtype = 'float')
    n = x.size

    find_x = 5
    CubicSplineInt(x,y,n,find_x)

'''
Function takes data series x,y and the length of the data set, n, and finds a
coresponding Y value (y_eval) for an input X value (x_eval).
'''
def CubicSplineInt(x,y,n,x_eval):
    #initilzing matrix A, solution vector b, and 2nd derivates vector Fpp
    A = np.matrix([[0 for x in range(n)] for y in range(n)], dtype = 'float')
    A[0,0] = 1
    A[n-1,n-1] = 1

    Fpp = np.matrix([None]*n, dtype = 'float')

    b = np.matrix([None]*n, dtype = 'float')
    b[0,0] = 0
    b[0,n-1] = 0

    #Populating the A matrix with left hand side of equation 18.37 from book
    #and b matrix with right hand side of same equation.
    for i in range(1,n-1):
        A[i,i-1] = x[0,i] - x[0,i-1]
        A[i,i] = 2*(x[0,i+1] - x[0,i-1])
        A[i,i+1] = x[0,i+1] - x[0,i]
        b[0,i] = (6*(y[0,i+1]-y[0,i]))/(x[0,i+1]-x[0,i]) \
                 + (6*(y[0,i-1]-y[0,i]))/(x[0,i]-x[0,i-1])

    #linear algebra to solve for Fpp vector using Fpp = A^-1*b
    Fpp = np.dot(np.linalg.inv(A),np.transpose(b))

    #calculating the interpolated point for the chosen x value x_eval
    for i in range(n-1):
        if (x[0,i-1] <= x_eval and x[0,i] >= x_eval):
```

```
            y_eval = (Fpp[i-1,0]*(x[0,i]-x_eval)**3)/(6*(x[0,i]-x[0,i-1])) \
                    + (Fpp[i,0]*(x_eval - x[0,i-1])**3)/(6*(x[0,i]-x[0,i-1])) \
                    + ((y[0,i-1]/(x[0,i]-x[0,i-1]))-(Fpp[i-1,0]*(x[0,i]-x[0,i-1]))/6)*\
                    (x[0,i]-x_eval)
                    + ((y[0,i]/(x[0,i]-x[0,i-1]))-(Fpp[i,0]*(x[0,i]-x[0,i-1]))/6)*\
                    (x_eval-x[0,i-1])
    print('OUTPUT')
    print('For the given data set and x value=',x_eval,', the interpolated y value is: ',\
          round(y_eval,4), sep = '')

Main()
```

```python
"""
Textbook problem 18.24

@author: Jacob Needham
"""

import numpy as np

#two data sets from problmes 18.5 and 18.6
problem5 = 18.5
x_185 = np.matrix([1.6,2,2.5,3.6,4,4.5], dtype = "float")
y_185 = np.matrix([2,8,14,15,8,2], dtype = "float")

problem6 = 18.6
x_186 = np.matrix([1,2,3,5,7,8], dtype = "float")
y_186 = np.matrix([3,6,19,99,291,444], dtype = "float")

'''
Main function defines data series of points and what X is to be found and calls
the cubic spline interpolation to find corresponding Y value.
'''
def Main(x,y,find_x,problem):
    #put data here:
    n = x.size
    CubicSplineInt(x,y,n,find_x,problem)

'''
Function takes data series x,y and the length of the data set, n, and finds a
coresponding Y value (y_eval) for an input X value (x_eval).
'''
def CubicSplineInt(x,y,n,x_eval,problem):
    #initilzing matrix A, solution vector b, and 2nd derivates vector Fpp
    A = np.matrix([[0 for x in range(n)] for y in range(n)], dtype = 'float')
    A[0,0] = 1
    A[n-1,n-1] = 1

    Fpp = np.matrix([None]*n, dtype = 'float')

    b = np.matrix([None]*n, dtype = 'float')
    b[0,0] = 0
    b[0,n-1] = 0

    #Populating the A matrix with left hand side of equation 18.37 from book
    #and b matrix with right hand side of same equation.
    for i in range(1,n-1):
        A[i,i-1] = x[0,i] - x[0,i-1]
        A[i,i] = 2*(x[0,i+1] - x[0,i-1])
        A[i,i+1] = x[0,i+1] - x[0,i]
        b[0,i] = (6*(y[0,i+1]-y[0,i]))/(x[0,i+1]-x[0,i]) \
                 + (6*(y[0,i-1]-y[0,i]))/(x[0,i]-x[0,i-1])

    #linear algebra to solve for Fpp vector using Fpp = A^-1*b
    Fpp = np.dot(np.linalg.inv(A),np.transpose(b))

    #calculating the interpolated point for the chosen x value x_eval
    for i in range(n-1):
        if (x[0,i-1] <= x_eval and x[0,i] >= x_eval):
```

```python
            y_eval = (Fpp[i-1,0]*(x[0,i]-x_eval)**3)/(6*(x[0,i]-x[0,i-1])) \
                    + (Fpp[i,0]*(x_eval - x[0,i-1])**3)/(6*(x[0,i]-x[0,i-1])) \
                    + ((y[0,i-1]/(x[0,i]-x[0,i-1]))-(Fpp[i-1,0]*(x[0,i]-x[0,i-1]))/6)*\
                    (x[0,i]-x_eval)
                    + ((y[0,i]/(x[0,i]-x[0,i-1]))-(Fpp[i,0]*(x[0,i]-x[0,i-1]))/6)*\
                    (x_eval-x[0,i-1])

    print('For the data set from problem ',problem,', and x value=',x_eval,\
          ', the interpolated y value is: ',round(y_eval,4), sep = '')

#calling the function for the two data sets from 18.5 and 18.6
Main(x_185,y_185,2.25,problem5)
Main(x_186,y_186,2.25,problem6)
```

Editor - C:\Users\Needh\Documents\Spring 2019\Numerical Methods\HW 6\18.24.py

18.17.py | 18.18.py | 18.23.py | 18.24.py

```python
"""
Textbook problem 18.24

@author: Jacob Needham
"""

import numpy as np

#two data sets from problmes 18.5 and 18.6
problem5 = 18.5
x_185 = np.matrix([1.6,2,2.5,3.6,4,4.5], dtype = "float")
y_185 = np.matrix([2,8,14,15,8,2], dtype = "float")

problem6 = 18.6
x_186 = np.matrix([1,2,3,5,7,8], dtype = "float")
y_186 = np.matrix([3,6,19,99,291,444], dtype = "float")

'''
Main function defines data series of points and what X is to be found and calls
the cubic spline interpolation to find corresponding Y value.
'''
def Main(x,y,find_x,problem):
    #put data here:
    n = x.size
    CubicSplineInt(x,y,n,find_x,problem)

'''
Function takes data series x,y and the length of the data set, n, and finds a
coresponding Y value (y_eval) for an input X value (x_eval).
'''
def CubicSplineInt(x,y,n,x_eval,problem):
    #initilzing matrix A, solution vector b, and 2nd derivates vector Fpp
    A = np.matrix([[0 for x in range(n)] for y in range(n)], dtype = 'float')
    A[0,0] = 1
    A[n-1,n-1] = 1

    Fpp = np.matrix([None]*n, dtype = 'float')

    b = np.matrix([None]*n, dtype = 'float')
    b[0,0] = 0
    b[0,n-1] = 0
```

IPython console

Console 1/A

```
In [139]: runfile('C:/Users/Needh/Documents/Spring 2019/Numerical Methods/HW 6/18.24.py', wdir='C:/Users/Needh/Documents/Spring 2019/Numerical Methods/HW 6')
OUTPUT
For the data set from problem 18.5, and x value=2.25, the interpolated y value is: 11.2289
For the data set from problem 18.6, and x value=2.25, the interpolated y value is: 8.0079

In [140]:
```

Permissions: **RW** | End-of-lines: **CRLF** | Encoding: **UTF-8** | Line: **64** | Column: **1** | Memory: **48 %**