CS 320

Programming Assignment #3
Matrix Multiplication in Python
40 points

Due Date/Time:

The program is due on Monday, October 19th at 11:59 pm on edoras. The name of your program must be `p3.py`

The Program:

For this assignment, you will multiply two matrices and print the result. If A is an *m* x *n* matrix, and B is an *n* x *p* matrix, then the product matrix C, which is A x B, is defined to be the *m* x *p* matrix whose entry in the i^{th} row and the j^{th} column is the sum of the products of corresponding entries of the i^{th} row of A and the j^{th} column of B. A standard algorithm is:

```
for(int i=0; i < m; i++)
    for(int j=0; j < p; j++) {
        C[i][j] = 0;
        for(int k=0; k < n; k++)
            C[i][j] += A[i][k] * B[k][j];
        }
```

Input:

will take the following form. The first three lines of the file contain the values *m*, *n*, and *p*, each on a line by themselves. Following are two matrices, *A* followed by *B* with no blank lines, of the dimensions specified, one row per line. Each row entry is separated by a space. Example: Given the following two matrices *A* and *B*:

```
A = |1 2|     B = |3|
                  |4|
```

The datafile will then have this format:

```
1
2
1
1 2
3
4
```

sys library:

For this assignment, the only library you may is sys. (Python has a built-in `array` module, which you may *not* use. Instead, the matrices will be lists which are very similar to arrays.) Immediately following the header comment section at the top of your program, add the line:

```
import sys
```

Output:

As usual, your program will first print a title line, consisting of the assignment number, your class account, and your name, all on one line. Then your program will print the two matrices to be multiplied, and finally print the resulting matrix (pretty formatting is appreciated, but not necessary). Each of the three matrices should be labeled. Example:

```
Program #3, csscxxxx, Student Name
Matrix A contents:
    1    2
    3    4
    5    6

Matrix B contents:
    7    8    9   10
   11   12   13   14

Matrix A * B is:
   29   32   35   38
   65   72   79   86
  101  112  123  134
```

The datafile read for this example is:

```
3
2
4
1 2
3 4
5 6
7 8 9 10
11 12 13 14
```

Additional Requirements:

- You may use brackets [ ] in your code.
- The matrices A and B must be declared as empty lists: `A = []` and `B = []`
- No error checking of input data file content is required for this assignment. You may assume that any datafile used for testing will contain only integer values in the format specified in the assignment.
- Your source code file will be run on edoras, using the command:
  `python3 p3.py datafileName`
- Your `p3.py` program file must contain exactly four functions, whose prototypes are:
  - `main()`
  - `read_matrices(A,B)` which returns matrix C
  - `mult_matrices(A,B,C)`
  - `print_matrix(arr)`

p3.py outline

```
#   Header comment section (typical assignment info)
import sys

# This function begins execution of program.
# Verify data input filename provided on command line: len(sys.argv)
# If error, output message for user: Usage: p3.py dataFileName'
# and quit, using sys.exit()
#
# Declare A, B, call read_matrices to initialize A, B, and store
#   return value as C
#
# Print A and B contents
#
# Call mult_matrices
#
# Print result contents
#
def main():

#   This function reads m, n, and p from the datafile.
#   Then matrices A and B are filled from the datafile.
#   Matrix C is then allocated size m x p.
#   The values for m, n, and p are local values filled in by this function
#   PARAMETERS in order are:
#   list      matrix A
#   list      matrix B
#   RETURN    matrix C
#
def read_matrices(A,B):


#   This function prints a matrix.  Rows and columns should be preserved.
#   PARAMETERS in order are:
#   list      The matrix to print
#
def print_matrix(matrix):


#  The two matrices A and B are multiplied, and matrix C contains the
#   result.
#   PARAMETERS in order are:
#   list       Matrix A
#   list       Matrix B
#   list       Matrix C (all zeros at this point)
#
def mult_matrices(A,B,C):

#  Begin program
if __name__ == '__main__':
    main()
```

Python scripting language basics

Python Tutorial

https://docs.python.org/3/tutorial/

Invoking the Interpreter

Login to edoras, then type:

```
python3
```

Since this is an interpreted (not compiled) language, you can start entering commands. In interactive mode, the prompt is ">>>". For practice, type:

```
>>> pi = 3.14
>>> print("pi is", pi)
pi is 3.14
>>> pi = "good"
>>> print("pi is", pi)
pi is good
>>> quit()
```

Output

```
print(""), print(), print('')    output blank line
print(x,end=" ")      output x followed by space
print(x,y,z)          output x y z
```

Loops

Indentation indicates the body portion of control structures (if-else, loops, functions). Note the use of the colon.

```
for item in list:
    print(item)
for row in 2Dlist:
    for column in row:
        print(column)
    print("")
for i in range(n)     loop i is 0,1,2,…,n-1
for j in range(3,8)   loop j is 3,4,5,6,7
```

Lists

```
len(myList)                 # get number of elements in list
myList[0]                   # first element in list
len(myList[0])              # get number of elements in first row of matrix
myList = []                 # empty list
myList = [ 1, 2, 3, 4 ]     # list of four elements
```

```
mylist = [ [1,2,3],          #  initialize a 3x3 matrix
           [4,5,6],
           [7,8,9] ]
myList = [ [0 for i in range(3)] for j in range(4)]   # initialize to zero a 3x4 matrix
```

Files

While there are many file mechanisms, we will illustrate one. To read the entire file, one line at a time:

```
with open("myInputFile.dat") as f:
    for item in f:
        print(item)
```

Once we have a line, use split() to extract elements. Suppose our line is `Big red balloon`, then the above code would print:

```
Big red balloon
```

But using split(), we can get individual values, and even assign them to variables:

```
  first, second, third = item.split()
  print(first)
  print(second)
  print(third)
```

This output is:

```
Big
red
balloon
```

Reading one integer per line, use next(f).split() to get the list, then cast and extract the first (and only) element:

```
  iVal = [int(x) for x in next(f).split()][0] # read line
```

Reading a certain number of lines of integers into a preexisting list:

```
  for i in range(iVal):
      myList.append([int(x) for x in next(f).split()])
```

sys Library: Command line arguments and exit()

The sys library contains functions needed for accessing command line arguments. Suppose we invoke the interpreter with:

```
[cssc0000 sandbox]$ python3 somePythonProg.py myInputFile.dat
```

Assume our program contains the following statements.

```
print('Number of arguments:', len(sys.argv), 'arguments.')
print('Argument List:', str(sys.argv))
```

Then the output from these statements is:

```
Number of arguments: 2 arguments.
Argument List: ['somePythonProg.py', 'myInputFile.dat']
```

We can use sys library's exit() function to terminate our program.

sys.exit()

Functions

Define functions before calling them. Use indentation and colons. Provide return (if desired).

```
def aFunc(parm1, parm2):
   # Do stuff
   return ' '  # may need to prevent outputting done
def anotherFunc():
   # Create something
   return theThingCreated
def yetAnotherFunc(parm1):
   # Do something with parm1
```

Call functions this way:

```
aFunc(x,y)
storeIt = anotherFunc()
yetAnotherFunc(x)
```