Aayush Shah,
D1 - 19BCE245
20 March 2021

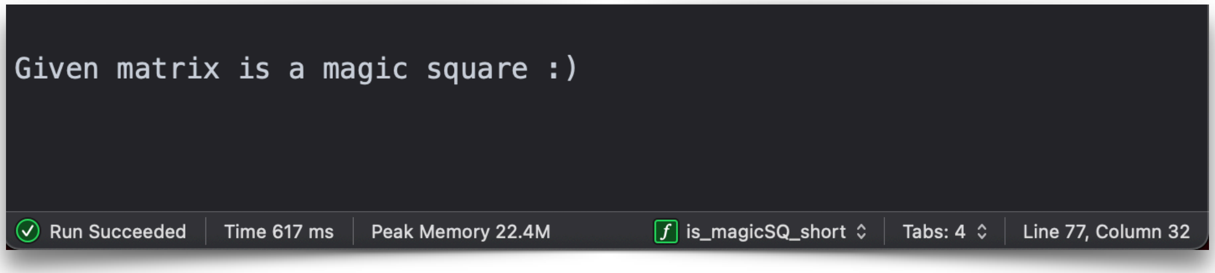
Practical 5

A. As you know, a magic square is a matrix all of whose row sums, column sums and the sums of the two diagonals are the same. (One diagonal of a matrix goes from the top left to the bottom right, the other diagonal goes from top right to bottom left.) Show by direct computation that if the matrix A is given by…

Code :

```
import numpy

def is_magicSQ(sq_matrix,size):
    min_sum_of_rows,max_sum_of_rows = 1e9,0

    #min and max of sum_of_elements_in_rows
    for row in range(size):
        row_sum = 0
        for column in range(size):
            row_sum += sq_matrix[row][column]
        min_sum_of_rows = min(min_sum_of_rows,row_sum)
        max_sum_of_rows = max(max_sum_of_rows,row_sum)

    if(min_sum_of_rows != max_sum_of_rows):
        return False

    min_sum_of_cols,max_sum_of_cols = 1e9,0

    #min and max of sum_of_elements_in_cols
    for column in range(size):
        col_sum = 0
        for row in range(size):
            col_sum += sq_matrix[row][column]
        min_sum_of_cols = min(min_sum_of_cols,col_sum)
        max_sum_of_cols = max(max_sum_of_cols,col_sum)

    if(min_sum_of_cols != max_sum_of_cols):
        return False


    principal_diagonal_sum,secondary_diagonal_sum =  0,0

    #checking diagonal sums
    for row_col in range(size):
        principal_diagonal_sum += sq_matrix[row_col][row_col]
        secondary_diagonal_sum += sq_matrix[size-row_col-1][size-row_col-1]

    if(principal_diagonal_sum != secondary_diagonal_sum):
        return False

    if(min_sum_of_rows == max_sum_of_rows ==  min_sum_of_cols == max_sum_of_cols == principal_diagonal_sum == secondary_diagonal_sum):
        return True
    return False

def is_magicSQ_short(sq_matrix,size):
    min_sum_of_rows = min(numpy.sum(sq_matrix, axis = 0))
    max_sum_of_rows = max(numpy.sum(sq_matrix, axis = 0))
    min_sum_of_cols = min(numpy.sum(sq_matrix, axis = 1))
    max_sum_of_cols = max(numpy.sum(sq_matrix, axis = 1))
    principal_diagonal_sum = sum(numpy.diagonal(sq_matrix))
    secondary_diagonal_sum = sum(numpy.fliplr(sq_matrix).diagonal())
    if(min_sum_of_rows == max_sum_of_rows ==  min_sum_of_cols == max_sum_of_cols == principal_diagonal_sum == secondary_diagonal_sum):
        return True
    else:
        return False

if __name__ == "__main__":

    #MANUAL PROCESS [takes input from user]
# size = int(input("Enter size of the square matrix : "))
# sq_matrix = numpy.zeros((size,size))
#
# for row in range(size):
#       for column in range(size):
#           str_for_input = "Enter value for matrix[" + str(row+1) + "][" + str(column+1) + "] : "
#           sq_matrix[row,column] = int(input(str_for_input))


    #Pre-defined [matrix as per question]
    sq_matrix = numpy.array([[17, 24, 1, 8, 15], [23, 5, 7, 14, 16],[ 4, 6, 13, 20, 22], [10, 12, 19, 21, 3], [11, 18, 25, 2, 9]])
    size = numpy.shape(0)

    if(is_magicSQ_short(sq_matrix, size)):
        print("\nGiven matrix is a magic square :)")
    else:
        print("\nGiven matrix is not a magic square :(")
```

Output :



```
Given matrix is a magic square :)
```

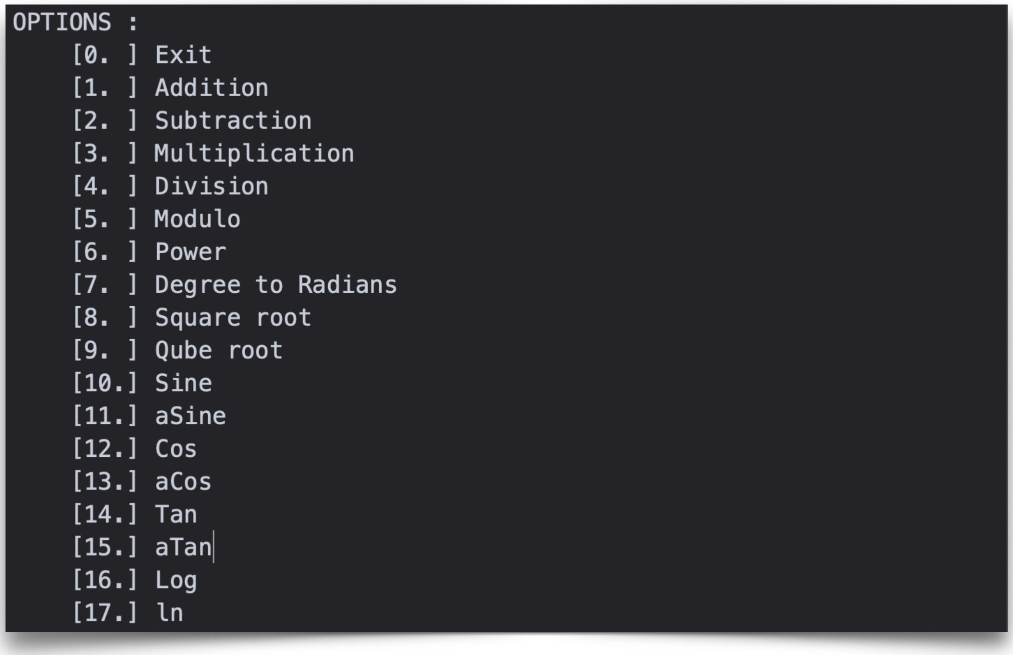
B. Create scientific calculator using numpy API.

Code :

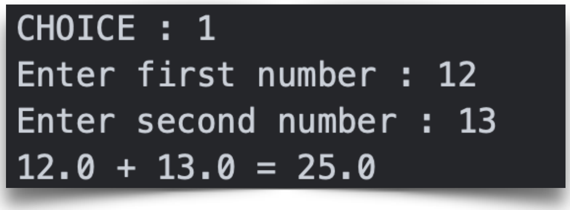
```
import numpy

def addition(n1,n2):
    return (n1+n2)

def subtraction(n1,n2):
    return (n1-n2)

def multiplication(n1,n2):
    return (n1*n2)

def division(n1,n2):
    while (n2==0):
          n2 = float(input("Denominator must be non-zero : (\nEnter again : "))
    return (n1/n2)

def modulo(n1,n2):
    return (n1%n2)

def power(n1,n2):
    return n1**n2

def qube_root(n1):
    while (n1<0):
          n1 = float(input("Non-negative value expected : (\nEnter again :"))
    return n1**(1/3)

def square_root(n1):
    while (n1<0):
          n1 = float(input("Non-negative value expected : (\nEnter again :"))
    return numpy.sqrt(n1)

def sine(n1):
    return numpy.sin(n1)
```

```

def a_sine(n1):
    return numpy.arcsine(n1)

def cos(n1):
    return numpy.cos(n1)

def a_cos(n1):
    return numpy.arccos(n1)

def tan(n1):
    return numpy.tan(n1)

def a_tan(n1):
    return numpy.arctan(n1)

def log_base10(n1):
    return numpy.log10(n1)

def log_baseE(n1):
    return numpy.log(n1)

def degree_to_rad(n1):
    return np.deg2rad(n1)

if __name__ == "__main__":
    while True:
        print("OPTIONS : ")
        print("\t[0. ] Exit")
        print("\t[1. ] Addition")
        print("\t[2. ] Subtraction")
        print("\t[3. ] Multiplication")
        print("\t[4. ] Division")
        print("\t[5. ] Modulo")
        print("\t[6. ] Power")
        print("\t[7. ] Degree to Radians")
        print("\t[8. ] Square root")
        print("\t[9. ] Qube root")
        print("\t[10.] Sine")
        print("\t[11.] aSine")
        print("\t[12.] Cos")
        print("\t[13.] aCos")
        print("\t[14.] Tan")
    print("\t[15.] aTan")
    print("\t[16.] Log")
    print("\t[17.] ln")
    choice = int(input("CHOICE : "))
    if(choice == 0):
        print("THANK YOU for using SCIENTIFIC CALCULATOR !")
        break
    elif(choice == 1):
        n1 = float(input("Enter first number : "))
        n2 = float(input("Enter second number : "))
        print(n1,"+",n2,"=",addition(n1,n2))
    elif(choice == 2):
        n1 = float(input("Enter first number : "))
        n2 = float(input("Enter second number : "))
        print(n1,"-",n2,"=",subtraction(n1,n2))
    elif(choice == 3):
        n1 = float(input("Enter first number : "))
        n2 = float(input("Enter second number : "))
        print(n1,"*",n2,"=",multiplication(n1,n2))
    elif(choice == 4):
        n1 = float(input("Enter first number : "))
        n2 = float(input("Enter second number : "))
        print(n1,"/",n2,"=",division(n1,n2))
    elif(choice == 5):
        n1 = float(input("Enter first number : "))
        n2 = float(input("Enter second number : "))
        print(n1,"%",n2,"=",modulo(n1,n2))
    elif(choice == 6):
        n1 = float(input("Enter base : "))
        n2 = float(input("Enter power : "))
        print(n1,"^(",n2,") =",power(n1,n2))
    elif(choice == 7):
        n1 = float(input("Enter degree : "))
        print("radians of",n1," : ",degree_to_rad(n1))
    elif(choice == 8):
        n1 = float(input("Enter number : "))
        print("square root of",n1,"=",square_root(n1))
    elif(choice == 9):
        n1 = float(input("Enter value : "))
        print("qube of",n1,"=",qube_root(n1))
    elif(choice == 10):
        n1 = float(input("Enter value : "))
        print("a sine of",n1,"=",sine(n1))
    elif(choice == 11):
        n1 = float(input("Enter value : "))
        print("a sine of",n1,"=",a_sine(n1))
    elif(choice == 12):
        n1 = float(input("Enter value : "))
        print("a cos of",n1,"=",cos(n1))
    elif(choice == 13):
        n1 = float(input("Enter value : "))
        print("a acos of",n1,"=",a_cos(n1))
    elif(choice == 14):
        n1 = float(input("Enter value : "))
        print("a tan of",n1,"=",tan(n1))
    elif(choice == 15):
        n1 = float(input("Enter value : "))
        print("a atan of",n1,"=",a_tan(n1))
    elif(choice == 16):
        n1 = float(input("Enter value : "))
        print("a log10 of",n1,"=",log_base10(n1))
    elif(choice == 17):
        n1 = float(input("Enter value : "))
        print("a logE of",n1,"=",log_baseE(n1))
    else:
        print("Invalid choice :(")
```

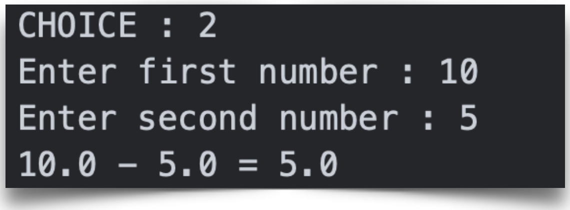
Output :



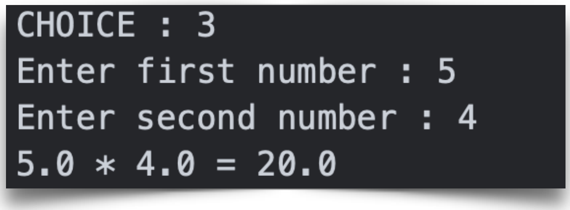
```
OPTIONS :
    [0. ] Exit
    [1. ] Addition
    [2. ] Subtraction
    [3. ] Multiplication
    [4. ] Division
    [5. ] Modulo
    [6. ] Power
    [7. ] Degree to Radians
    [8. ] Square root
    [9. ] Qube root
    [10.] Sine
    [11.] aSine
    [12.] Cos
    [13.] aCos
    [14.] Tan
    [15.] aTan
    [16.] Log
    [17.] ln
```



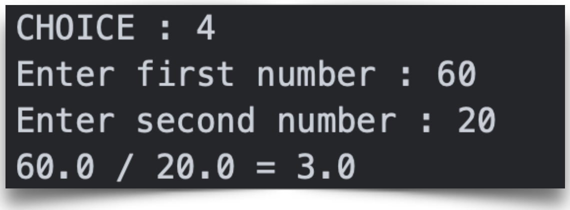
```
CHOICE : 1
Enter first number : 12
Enter second number : 13
12.0 + 13.0 = 25.0
```



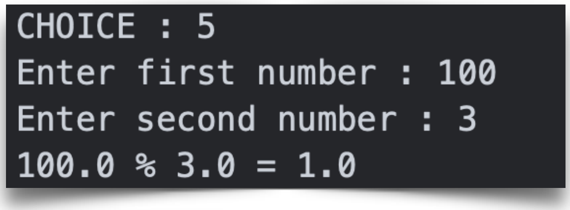
```
CHOICE : 2
Enter first number : 10
Enter second number : 5
10.0 - 5.0 = 5.0
```



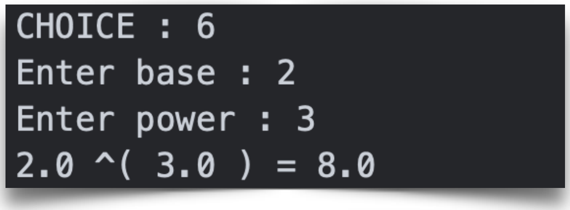
```
CHOICE : 3
Enter first number : 5
Enter second number : 4
5.0 * 4.0 = 20.0
```



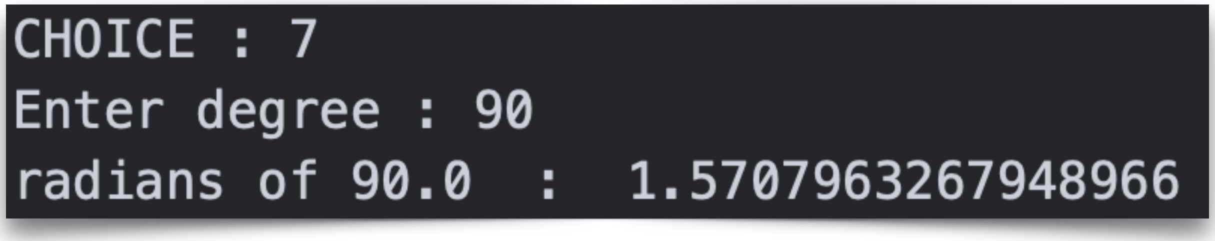
```
CHOICE : 4
Enter first number : 60
Enter second number : 20
60.0 / 20.0 = 3.0
```



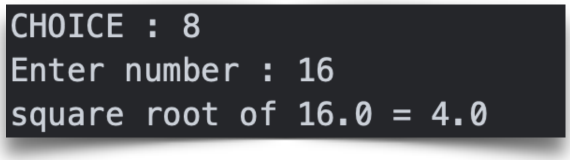
```
CHOICE : 5
Enter first number : 100
Enter second number : 3
100.0 % 3.0 = 1.0
```



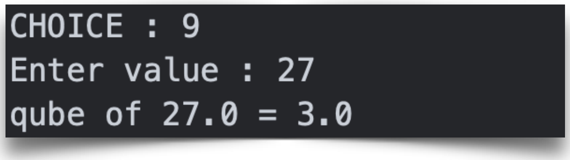
```
CHOICE : 6
Enter base : 2
Enter power : 3
2.0 ^( 3.0 ) = 8.0
```



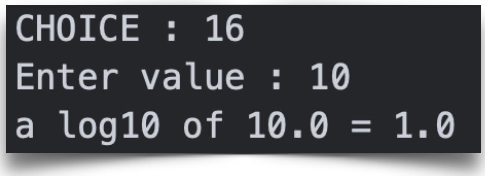
```
CHOICE : 7
Enter degree : 90
radians of 90.0  :  1.5707963267948966
```



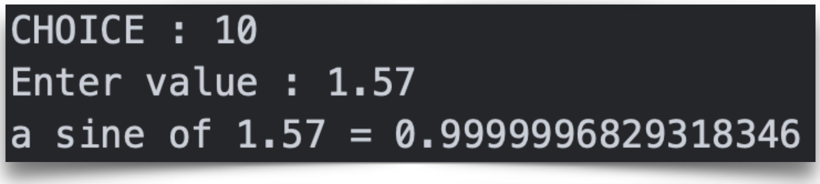
```
CHOICE : 8
Enter number : 16
square root of 16.0 = 4.0
```



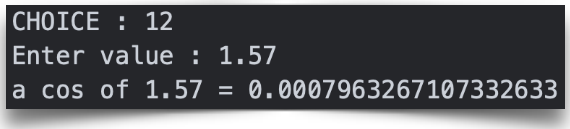
```
CHOICE : 9
Enter value : 27
qube of 27.0 = 3.0
```



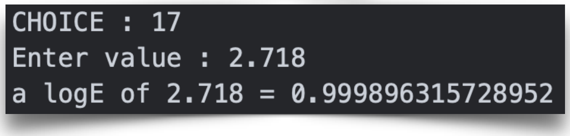
```
CHOICE : 16
Enter value : 10
a log10 of 10.0 = 1.0
```



```
CHOICE : 10
Enter value : 1.57
a sine of 1.57 = 0.9999996829318346
```

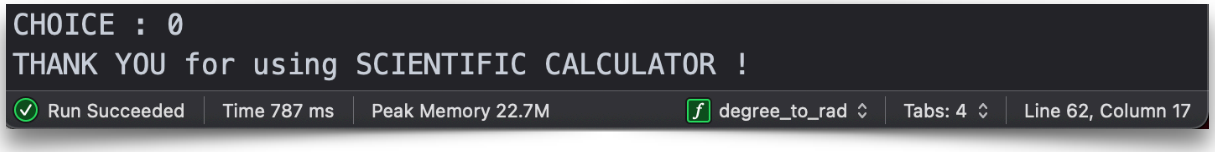


```
CHOICE : 12
Enter value : 1.57
a cos of 1.57 = 0.0007963267107332633
```



```
CHOICE : 17
Enter value : 2.718
a logE of 2.718 = 0.999896315728952
```

And more…



```
CHOICE : 0
THANK YOU for using SCIENTIFIC CALCULATOR !
```