P346 (Computational Physics Lab)
Assignment 2
Name: Ratul Das
Roll No.: 1911128

[33]:
```
from math import tan, sqrt, atan, pi
import random
```

Question 1

[2]:
```
from file1 import mat_A
from file2 import mat_B
from file3 import mat_C
from file4 import mat_D

def displayMatrix(matrix): #display given matrix
    for row in matrix:
            print(row)

def matrixProduct(A,B): #finds product of two matrices
    if len(A[0])==len(B):
        C = []
        dim_C = (len(A),len(B[0]))
        for i in range(dim_C[0]):
            row_C = []
            for j in range(dim_C[1]):
                elem_C = 0
                for k in range(len(B)):
                    elem_C += (A[i][k])*(B[k][j])
                row_C.append(elem_C)
            C.append(row_C)
        displayMatrix(C)
    else:
        print("Matrix multiplication is undefined for the given matrices!")

def dotProduct(A,B):  #finds dot product of two matrices
    if len(A)==len(B):
        A_transpose = []
        for k in range(len(A[0])):
            row_At = []
            for l in range(len(A)):
                row_At.append(A[l][k])
            A_transpose.append(row_At)
        C = []
        dim_C = (len(A_transpose),len(B[0]))
        for i in range(dim_C[0]):
            row_C = []
            for j in range(dim_C[1]):
                elem_C = 0
                for k in range(len(B)):
                    elem_C += (A_transpose[i][k])*(B[k][j])
                row_C.append(elem_C)
            C.append(row_C)
        displayMatrix(C)
    else:
        print("Matrix multiplication is undefined for the given matrices!")
```

[3]:
```
displayMatrix(mat_A)
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

[4]:
```
displayMatrix(mat_B)
```

```
[3, 2, 1]
[6, 5, 4]
[9, 8, 7]
```

[5]:
```
displayMatrix(mat_C)
```

```
[1, 2, 3]
```

[6]:
```
displayMatrix(mat_D)
```

```
[30, 20, 10]
```

[7]:
```
dotProduct(mat_C,mat_D)
```

```
[30, 20, 10]
[60, 40, 20]
[90, 60, 30]
```

[8]:
```
matrixProduct(mat_A,mat_B)
```

```
[42, 36, 30]
[96, 81, 66]
[150, 126, 102]
```

[9]:
```
matrixProduct(mat_C,mat_A)
```

```
[30, 36, 42]
```

[10]:
```
matrixProduct(mat_B,mat_D)
```

```
Matrix multiplication is undefined for the given matrices!
```

Question 2

[11]:
```python
class myComplex:
    def initNumber(self): #initiates a myComplex instance, asks for input
        self.real = float(input("Enter real component of complex number: "))
        self.imag = float(input("Enter imaginary component of complex number: "))

    def display(self):    #displays the instance of myComplex
        print(str(self.real) + " + " + str(self.imag) + "i")

    def conjugate(self):  #displays conjugate of myComplex
        self.real = self.real
        self.imag = -self.imag
        print("Conjugate of given complex number is: ")
        self.display()

    def add(self, c1, c2): #adds 2 instances of myComplex
        self.real = c1.real + c2.real
        self.imag = c1.imag + c2.imag

    def product(self, c1, c2): #multiplies 2 instances of myComplex
        self.real = (c1.real)*(c2.real) - (c1.imag)*(c2.imag)
        self.imag = (c1.real)*(c2.imag) + (c1.imag)*(c2.real)

    def divide(self, c1, c2): #divides 2 instances of myComplex
        self.real = ((c1.real)*(c2.real)
                     + (c1.imag)*(c2.imag))/((c2.real)**2 + (c2.imag)**2)
        self.imag = ((c1.imag)*(c2.real)
                     - (c1.real)*(c2.imag))/((c2.real)**2 + (c2.imag)**2)

    def modulus(self):     #finds modulus of myComplex instance
        mod = sqrt((self.real)**2 + (self.imag)**2)
        print("Modulus of the given complex number is " + str(mod))

    def phase(self):       #finds phase angle of myComplex instance
        if (self.real > 0):
            print("Phase angle of complex number = "
                  + str(atan(self.imag/self.real)) + " radians")

        elif (self.real < 0) and (self.imag >= 0):
            print("Phase angle of complex number = "
                  + str(atan(self.imag/self.real) + pi) + " radians")

        elif (self.real < 0) and (self.imag < 0):
            print("Phase angle of complex number = "
                  + str(atan(self.imag/self.real) - pi) + " radians")
        elif (self.real == 0) and (self.imag > 0):
            print("Phase angle of complex number = "
                  + str(pi/2) + " radians")

        elif (self.real == 0) and (self.imag < 0):
            print("Phase angle of complex number = "
                  + str(-pi/2) + " radians")

        else:
            print("Phase angle of given complex number is undefined")
```

[12]:
```
#7 instances of myComplex are initiated
#4 of them are initiated with values
z1 = myComplex()
z2 = myComplex()
z3 = myComplex()
z4 = myComplex()
#the last 3 are obtained from computations with previous 4 instances
z5 = myComplex()
z6 = myComplex()
z7 = myComplex()
```

[13]:
```
z1.initNumber()
```

```
Enter real component of complex number: 1.5
Enter imaginary component of complex number: 4.5
```

[14]:
```
z2.initNumber()
```

```
Enter real component of complex number: 5
Enter imaginary component of complex number: -2
```

[15]:
```
z3.initNumber()
```

```
Enter real component of complex number: 0
Enter imaginary component of complex number: 2
```

[16]:
```
z4.initNumber()
```

```
Enter real component of complex number: 3.5
Enter imaginary component of complex number: 0
```

[17]:
```
z1.display(),z2.display(),z3.display(),z4.display()
```

```
1.5 + 4.5i
5.0 + -2.0i
0.0 + 2.0i
3.5 + 0.0i
```

[17]: (None, None, None, None)

[18]:
```
z5.add(z1,z2),z5.display()
```

```
6.5 + 2.5i
```

[18]: (None, None)

[19]:
```
z6.product(z1,z3),z6.display()
```

```
-9.0 + 3.0i
```

[19]: (None, None)

[20]:
```
z7.divide(z1,z4),z7.display()
```

```
0.42857142857142855 + 1.2857142857142858i
```

[20]: (None, None)

[21]:
```
z1.conjugate()
```

```
Conjugate of given complex number is:
1.5 + -4.5i
```

[22]:
```
z1.modulus()
```

```
Modulus of the given complex number is 4.743416490252569
```

[23]:
```
z1.phase()
```

```
Phase angle of complex number = -1.2490457723982544 radians
```

Question 3

[24]:
```
def avgDistance(N): #finds average distances between N discrete points
    if N <= 1: #less/equal to 1 eliminated
        print("Please enter a natural number greater than 1")
    elif N%1 != 0: #fractional numbers eliminated
        print("Please enter a natural number greater than 1")
    else:
        s = 0
        list = []

        for i in range(N):
            s+=i
        m = s

        for j in range(1, N):
            m += j - (N - j)
            list.append(m)
        list.append(s)
        x = 0

        for k in range(0, len(list)):
            x += list[k]

        dist = x/(N*N)

        print("Average distance between 2 points on a line with")
        print(str(N) + " discrete points is " + str(dist))
```

[25]:
```
avgDistance(2)
```

```
Average distance between 2 points on a line with
2 discrete points is 0.5
```

[26]:
```
avgDistance(10)
```

```
Average distance between 2 points on a line with
10 discrete points is 3.3
```

[27]:
```
avgDistance(5.5)
```

```
Please enter a natural number greater than 1
```

[28]:
```
avgDistance(-2)
```

```
Please enter a natural number greater than 1
```

[29]:
```
avgDistance(1)
```

```
Please enter a natural number greater than 1
```

Question 4

[35]:
```
#capitals of the 24 countries qualified for FIFA 2019 Women's World Cup
capitals =['paris', 'washington dc', 'berlin', 'london',
           'ottawa', 'canberra', 'amsterdam', 'tokyo',
           'stockholm', 'brasilia', 'madrid', 'oslo',
           'seoul', 'beijing', 'rome', 'auckland', 'edinburgh',
           'bangkok', 'buenos aires', 'santiago', 'abuja',
           'yaounde', 'cape town', 'kingston']

def displayStatus(string): #displays current state of the game
    for char in string:
        print(char, end=" ")
```

```
def updateWordState(corr_state,state,guessed): #revises state
    new_state = []
    for i in range(len(state)):
        if corr_state[i]==guessed:
            new_state.append(guessed)
        elif corr_state[i]==state[i]:
            new_state.append(state[i])
        elif corr_state[i]==" ":
            new_state.append(" ")
        else:
            new_state.append("_")
    return new_state

def setLives(string): #obtains number of chances available
    if len(string)<5: #if word is smaller than 5 letters
        return 2      #then min number of chances is 2
    else: #closest integer to 40% of string length is chosen
        return int(0.4*len(string))

def checkWin(string1,string2):#checks if correct state and
    count = 0                 #current state of the game are equal
    for m in range(len(string1)):
        if string1[m]==string2[m]:
            count+=1
    if count==len(string1):
        return 1
    else:
        return 0

def playHangman(): #initiates a game of Hangman
    lowercase = 'abcdefghijklmnopqrstuvwxyz' #lowercase alphabet
    print("Welcome! Let's play a game of Hangman!")
    decision = input("Enter 'y' to start a new round: ")

    #game is started only on affirmative input
    while decision == 'y' or decision == 'Y':
        random.seed(a=None, version=2) #to randomise choice each time
        word = random.choice(capitals)
        guess_list = []
        correct_guess = []
        win=False

        available_lives = setLives(word)

        word_state = []    #current state of game
        correct_state = [] #state needed to win
        for letter in word:
            if letter == " ":
                word_state.append(" ")
                correct_state.append(" ")
            else:
                word_state.append("_")
                correct_state.append(letter)

        print()
        print("The word for this round is: ")
        displayStatus(word_state) #shows word with dashes and spaces
        print()
        print("You have " + str(available_lives) + " wrong guesses to start with.␣
↪")


        while available_lives!=0 and win!=True: #
            guess = input("Guess a letter in the name of the city: ")
            guess = guess.lower()     #all inputs are converted to lowercase
            guess_list.append(guess)
            if (guess in correct_state) and (guess in lowercase) and (guess not␣
↪in correct_guess):
                #letter is added to current state only if
                #above 3 conditions are satisfied
                print("Yay! That letter is in the name of the city!")
                word_state = updateWordState(correct_state,word_state,guess)
                #current state is updated with correct letters
                correct_guess.append(guess) #repeated letters kept in memory
                if checkWin(correct_state,word_state)==1: #checks current state␣
↪for win condition
                    print("Congratulations! You won the round!")
                    win=True
            elif guess in correct_guess: #repeated letters excluded
                print("Uh oh. That letter has already been guessed correctly!")
            else:
                #applies decrement to number of
                #chances left in case of failed guess
                print("Sorry :( That letter is not in the name of the city!")
                available_lives = available_lives - 1
                print("You have " + str(available_lives) + " wrong guesses left!
↪")

            displayStatus(word_state) #shows updated current state
            print()
            print()

        if available_lives==0:
            #provides a game over message to the user upon
            #exit from the while loop due to 0 chances left
            print("Tough luck :( You lost the game.")
            print("The city's name is: " + str(word))

        print("Do you want to start another round?")
        #repeats the outer while loop to continue/stop playing
        decision = input("Enter 'y' to start a new round: ")

    print()
    print("Ah, you must be busy. Maybe another time!")
```

[37]:
```
playHangman()
```

```
Welcome! Let's play a game of Hangman!
Enter 'y' to start a new round: y

The word for this round is:
_ _ _ _ _
You have 2 wrong guesses to start with.
Guess a letter in the name of the city: P
Yay! That letter is in the name of the city!
p _ _ _ _

Guess a letter in the name of the city: %
Sorry :( That letter is not in the name of the city!
You have 1 wrong guesses left!
p _ _ _ _

Guess a letter in the name of the city: r
Yay! That letter is in the name of the city!
p _ r _ _

Guess a letter in the name of the city: s
Yay! That letter is in the name of the city!
p _ r _ s

Guess a letter in the name of the city: i
Yay! That letter is in the name of the city!
p _ r i s

Guess a letter in the name of the city: a
Yay! That letter is in the name of the city!
Congratulations! You won the round!
p a r i s

Do you want to start another round?
```

```
Enter 'y' to start a new round: n

Ah, you must be busy. Maybe another time!
```

[48]:
```
playHangman()
```

```
Welcome! Let's play a game of Hangman!
Enter 'y' to start a new round: y

The word for this round is:
_ _ _ _ _ _ _ _
You have 3 wrong guesses to start with.
Guess a letter in the name of the city: a
Yay! That letter is in the name of the city!
_ a _ _ _ a _ _

Guess a letter in the name of the city: I
Yay! That letter is in the name of the city!
_ a _ _ i a _ _

Guess a letter in the name of the city: 6
Sorry :( That letter is not in the name of the city!
You have 2 wrong guesses left!
_ a _ _ i a _ _

Guess a letter in the name of the city: G
Yay! That letter is in the name of the city!
_ a _ _ i a g _

Guess a letter in the name of the city: s
Yay! That letter is in the name of the city!
s a _ _ i a g _

Guess a letter in the name of the city: N
Yay! That letter is in the name of the city!
s a n _ i a g _

Guess a letter in the name of the city: t
Yay! That letter is in the name of the city!
s a n t i a g _

Guess a letter in the name of the city: O
Yay! That letter is in the name of the city!
Congratulations! You won the round!
s a n t i a g o

Do you want to start another round?
Enter 'y' to start a new round: n
```

```
Ah, you must be busy. Maybe another time!
```

[49]:
```
playHangman()
```

```
Welcome! Let's play a game of Hangman!
Enter 'y' to start a new round: no

Ah, you must be busy. Maybe another time!
```

[50]:
```
playHangman()
```

```
Welcome! Let's play a game of Hangman!
Enter 'y' to start a new round: y

The word for this round is:
_ _ _ _ _ _   _ _ _ _ _
You have 4 wrong guesses to start with.
Guess a letter in the name of the city: b
Yay! That letter is in the name of the city!
b _ _ _ _ _   _ _ _ _ _

Guess a letter in the name of the city: E
Yay! That letter is in the name of the city!
b _ e _ _ _   _ _ _ e _

Guess a letter in the name of the city: n
Yay! That letter is in the name of the city!
b _ e n _ _   _ _ _ e _

Guess a letter in the name of the city: O
Yay! That letter is in the name of the city!
b _ e n o _   _ _ _ e _

Guess a letter in the name of the city: o
Uh oh. That letter has already been guessed correctly!
b _ e n o _   _ _ _ e _

Guess a letter in the name of the city: s
Yay! That letter is in the name of the city!
b _ e n o s   _ _ _ e s

Guess a letter in the name of the city: e
Uh oh. That letter has already been guessed correctly!
b _ e n o s   _ _ _ e s

Guess a letter in the name of the city: i
Yay! That letter is in the name of the city!
```

```
b _ e n o s   _ i _ e s

Guess a letter in the name of the city: u
Yay! That letter is in the name of the city!
b u e n o s   _ i _ e s

Guess a letter in the name of the city: u
Uh oh. That letter has already been guessed correctly!
b u e n o s   _ i _ e s

Guess a letter in the name of the city: a
Yay! That letter is in the name of the city!
b u e n o s   a i _ e s

Guess a letter in the name of the city: w
Sorry :( That letter is not in the name of the city!
You have 3 wrong guesses left!
b u e n o s   a i _ e s

Guess a letter in the name of the city: t
Sorry :( That letter is not in the name of the city!
You have 2 wrong guesses left!
b u e n o s   a i _ e s

Guess a letter in the name of the city: k
Sorry :( That letter is not in the name of the city!
You have 1 wrong guesses left!
b u e n o s   a i _ e s

Guess a letter in the name of the city: r
Yay! That letter is in the name of the city!
Congratulations! You won the round!
b u e n o s   a i r e s

Do you want to start another round?
Enter 'y' to start a new round: n

Ah, you must be busy. Maybe another time!
```

[53]:
```
playHangman()
```

```
Welcome! Let's play a game of Hangman!
Enter 'y' to start a new round: y

The word for this round is:

_ _ _ _
You have 2 wrong guesses to start with.
Guess a letter in the name of the city: o
```

```
Yay! That letter is in the name of the city!
_ o _ _

Guess a letter in the name of the city: r
Yay! That letter is in the name of the city!
r o _ _

Guess a letter in the name of the city: m
Yay! That letter is in the name of the city!
r o m _

Guess a letter in the name of the city: e
Yay! That letter is in the name of the city!
Congratulations! You won the round!
r o m e

Do you want to start another round?
Enter 'y' to start a new round: y

The word for this round is:
_ _ _ _ _
You have 2 wrong guesses to start with.
Guess a letter in the name of the city: r
Sorry :( That letter is not in the name of the city!
You have 1 wrong guesses left!
_ _ _ _ _

Guess a letter in the name of the city: o
Sorry :( That letter is not in the name of the city!
You have 0 wrong guesses left!
_ _ _ _ _

Tough luck :( You lost the game.
The city's name is: abuja
Do you want to start another round?
Enter 'y' to start a new round: y

The word for this round is:
_ _ _ _ _
You have 2 wrong guesses to start with.
Guess a letter in the name of the city: o
Sorry :( That letter is not in the name of the city!
You have 1 wrong guesses left!
_ _ _ _ _

Guess a letter in the name of the city: b
Yay! That letter is in the name of the city!
_ b _ _ _
```

```
Guess a letter in the name of the city: u
Yay! That letter is in the name of the city!
_ b u _ _

Guess a letter in the name of the city: j
Yay! That letter is in the name of the city!
_ b u j _

Guess a letter in the name of the city: a
Yay! That letter is in the name of the city!
Congratulations! You won the round!
a b u j a

Do you want to start another round?
Enter 'y' to start a new round: n

Ah, you must be busy. Maybe another time!
```