programs

April 13, 2023

https://github.com/Sachin-D-N/Python_solved_problems/tree/master/Python_Problem_Practise

Python: without numpy or sklearn

Q1: Given two matrices please print the product of those two matrices

[7]:
```python
from random import randint
from random import randrange
from random import choices
from collections import Counter
from itertools import accumulate
from bisect import bisect
import math
```

[21]:
```python
# return a single value in range [0,2) i.e., exclusive of 2
print(randrange(0,2))
```

```
0
```

[32]:
```python
# return a single value in range [0,2] i.e., inclusive of 2
print(randint(0,2))
```

```
2
```

[11]:
```python
mat_A = [[i for i in range(10)] for j in range(10)]
print(mat_A)
```

```
[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4,
5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4,
5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]]
```

[24]:
```python
mat_A = [[randrange(0,5) for i in range(10)] for j in range(10)]
print(mat_A)
```

```
[[3, 1, 2, 4, 4, 4, 0, 3, 0, 0], [2, 2, 4, 4, 2, 3, 1, 1, 3, 4], [1, 1, 0, 2, 4,
0, 0, 4, 1, 1], [0, 1, 3, 4, 2, 4, 4, 4, 2, 0], [3, 0, 0, 2, 0, 3, 1, 0, 4, 1],
[0, 4, 2, 2, 1, 4, 4, 0, 1, 4], [4, 2, 4, 1, 4, 3, 1, 4, 4, 2], [3, 0, 1, 2, 4,
2, 1, 1, 4, 1], [1, 1, 2, 4, 0, 3, 2, 3, 3, 3], [4, 1, 0, 0, 3, 4, 1, 3, 1, 2]]
```

[50]:
```python
for row in mat_A:
    a = '\t'.join([str(cell) for cell in row])
print(a)
print(type(a))
```

```
4       1       0       0       3       4       1       3       1       2
<class 'str'>
```

[48]:
```python
b = []
for row in mat_A:
    a = '\t'.join([str(cell) for cell in row])
    b.append(a)
    break
print(b)
```

```
['3\t1\t2\t4\t4\t4\t0\t3\t0\t0']
```

[55]:
```python
b = []
for row in mat_A:
    a = '\t'.join([str(cell) for cell in row])
    b.append(a)
print(b)
a = '\n'.join(b)
print(a)
```

```
['3\t1\t2\t4\t4\t4\t0\t3\t0\t0', '2\t2\t4\t4\t2\t3\t1\t1\t3\t4',
'1\t1\t0\t2\t4\t0\t0\t4\t1\t1', '0\t1\t3\t4\t2\t4\t4\t4\t2\t0',
'3\t0\t0\t2\t0\t3\t1\t0\t4\t1', '0\t4\t2\t2\t1\t4\t4\t0\t1\t4',
'4\t2\t4\t1\t4\t3\t1\t4\t4\t2', '3\t0\t1\t2\t4\t2\t1\t1\t4\t1',
'1\t1\t2\t4\t0\t3\t2\t3\t3\t3', '4\t1\t0\t0\t3\t4\t1\t3\t1\t2']
3       1       2       4       4       4       0       3       0       0
2       2       4       4       2       3       1       1       3       4
1       1       0       2       4       0       0       4       1       1
0       1       3       4       2       4       4       4       2       0
3       0       0       2       0       3       1       0       4       1
0       4       2       2       1       4       4       0       1       4
4       2       4       1       4       3       1       4       4       2
3       0       1       2       4       2       1       1       4       1
1       1       2       4       0       3       2       3       3       3
4       1       0       0       3       4       1       3       1       2
```

[33]:
```python
print('\n'.join(['\t'.join([str(cell) for cell in row]) for row in mat_A]),'\n')
```

```
3       1       2       4       4       4       0       3       0       0
2       2       4       4       2       3       1       1       3       4
1       1       0       2       4       0       0       4       1       1
0       1       3       4       2       4       4       4       2       0
3       0       0       2       0       3       1       0       4       1
```

```
0       4       2       2       1       4       4       0       1       4
4       2       4       1       4       3       1       4       4       2
3       0       1       2       4       2       1       1       4       1
1       1       2       4       0       3       2       3       3       3
4       1       0       0       3       4       1       3       1       2
```

[1]:
```python
from random import randrange

n_rows_A = int(input('Number of rows for matrix A: '))
n_cols_A = int(input('Number of columns for matrix A: '))

# first arg can be omitted in range func, randrange func provides a single␣
 ↪value at a time
mat_A = [[randrange(1, 100) for i in range(n_cols_A)] for j in range(n_rows_A)]␣
 ↪# i   [0,n_cols_A)
print("your matrix A:")
print('\n'.join(['\t'.join([str(cell) for cell in row]) for row in␣
 ↪mat_A]),'\n') # for printing in format

n_rows_B = int(input('Number of rows for matrix B: '))
n_cols_B = int(input('Number of columns for matrix B: '))

mat_B = [[randrange(1, 100) for i in range(n_cols_B)] for j in range(n_rows_B)]
print("your matrix B:")
print('\n'.join(['\t'.join([str(cell) for cell in row]) for row in mat_B]),'\n')

def matrix_mul(A, B):
    result = []
    for m in range(len(A)): #  picking a row of A, m   [0,len(A))
        rows = []
        for i in range(len(B[0])): # picking a column of B
            row_ele_R = 0
            for j in range (len(A[0])): # picking row element of A and column␣
 ↪element of B
                row_ele_R += A[m][j] * B[j][i] # row element of result
            rows.append(row_ele_R)
        result.append(rows)
    return result

if (len(mat_A[0]) != len(mat_B)): # (A cols size != B rows size)
    print('The two matrices cannot be multiplied. Columns-rows mismatch.')
else:
    print("Result of matrix_A x matrix_B:")
    print('\n'.join(['\t'.join([str(cell) for cell in row]) for row in␣
 ↪matrix_mul(mat_A,mat_B)]))
```

```
Number of rows for matrix A: 5
Number of columns for matrix A: 5
your matrix A:
32      9       26      57      5
32      39      89      96      1
84      61      56      99      84
55      13      14      46      60
6       70      27      7       32

Number of rows for matrix B: 5
Number of columns for matrix B: 5
your matrix B:
35      94      4       62      67
97      81      26      21      79
56      63      35      57      10
22      18      16      88      43
67      87      82      16      22

Result of matrix_A x matrix_B:
5038    6836    2594    8751    5676
12066   13589   5875    16340   10265
19799   25455   12354   19737   17112
9002    13153   6704    9489    8150
10810   10845   5525    4509    7207
```

Q2: Proportional Sampling - Select a number randomly with probability proportional to its magnitude from the given array of n elements

Consider an experiment, selecting an element from the list A randomly with probability proportional to its magnitude. assume we are doing the same experiment for 100 times with replacement, in each experiment you will print a number that is selected randomly from A.

[34]:
```
A = [0,5,27,6,13,28,100,45,10,79]

def propotional_sampling(A, n=100):
    # calculate cumulative sum from A:
    # accumulate(A) => <itertools.accumulate object at 0x0000016E201E0A88>
    # *accumulate(A) => 0 5 32 38 51 79 179 224 234 313
    cum_sum = [*accumulate(A)]
    # cum_sum = [0, 5, 32, 38, 51, 79, 179, 224, 234, 313]

    out = []
    for _ in range(n):
        # returns a next floating number in the range [0.0, 1.0)
        i = random.random()                     # i = [0.0, 1.0)

        # bisect.bisect(a, x, ...) =>
        # locate the insertion point for x in a to maintain sorted order.
        # returns an insertion point which comes after (to the right of) any␣
↪existing entries of x in a.
        # i*cum_sum[-1] => 193.1, 41, 36, 80, ...
        idx = bisect(cum_sum, i*cum_sum[-1])    # get index to list A
        # idx => 7, 4, 3, 6, ...
        out.append(A[idx])

    tmp = Counter(out)
    # tmp => Counter({79: 30, 100: 23, 45: 15, 28: 11, 13: 7, 27: 7, 10: 4, 6:␣
↪2, 5: 1})
    # sorted(tmp) => [5, 6, 10, 13, 27, 28, 45, 79, 100]
    # tmp[5] => 1

    print("Frequency( Number ) ==>")
    # sort according to dictionary values; ascending
    for item in sorted(tmp):
        print(str(tmp[item])+"("+str(item)+")", end=" > ") # loop print in a␣
↪line


print("Clearly the number of times each elements appears is proportional to its␣
↪magnitude.")
print(propotional_sampling(A))
```

```
Clearly the number of times each elements appears is proportional to its
magnitude.
Frequency( Number ) ==>
2(5) > 1(6) > 3(10) > 4(13) > 10(27) > 8(28) > 15(45) > 25(79) > 32(100) > None
```

Q3: Replace the digits in the string with #

consider a string that will have digits in that, we need to remove all the not digits and replace the digits with #

[70]:
```
import re

def replace_digits(st):
    str_2 = ""
    result = re.findall(r'\d', st) # finding Unicode decimal digit [0-9] in␣
↪string
    print(result)
    for m in result:
        str_2 += "#"
    return str_2 # modified string which is after replacing the # with digits

str_1 = input("Enter the string : ")
replace_digits(str_1)
```

```
Enter the string : #2a$#b%c%561#
['2', '5', '6', '1']
```

```
[70]: '####'
```

Q4: Students marks dashboard

consider the marks list of class students given two lists Students = ['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10'] Marks = [45, 78, 12, 14, 48, 43, 45, 98, 22, 80] from the above two lists the Student[0] got Marks[0], Student[1] got Marks[1] and so on your task is to print the name of students a. Who got top 5 ranks, in the descending order of marks b. Who got least 5 ranks, in the increasing order of marks d. Who got marks between >25th percentile <75th percentile, in the increasing order of marks

```
[58]: students = ['sam','ram','tom','pam','gor','kim','jim','toh','gig','abe']
marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]

# zip() will bind together corresponding elements of students and marks
# e.g. [('student1', 45), ('student2', 78), ...]
grades = list(zip(students, marks))

# once that's all in one list of 2-tuples, sort it by calling .sort() or using␣
↪sorted()
# give it a "key", which specifies what criteria it should sort on
# in this case, it should sort on the mark, so the second element (index 1) of␣
↪the tuple
_ = grades.sort(key=lambda e:e[1])
# [('tom', 12), ('pam', 14), ('gig', 35), ('kim', 43), ('sam', 45), ('jim',␣
↪45), ('gor', 48), ('ram', 78), ('abe', 80), ('toh', 98)]

# now, just slice out the 25th and 75th percentile based on the length of that␣
↪list.
# .ceil(); rounds a number UP to the nearest integer.
# .floor(); rounds a number DOWN to the nearest integer.
# 1/4 is 25th percentile and 3/4 is 75th percentile.
twentyfifth = math.ceil(len(grades) / 4)
seventyfifth = math.floor(3 * len(grades) / 4)

marks = sorted(set(marks))


# [::-1]; to reverse a list.
# .reverse(); do not work when list inner elements are tuple.
top5 = marks[-5:][::-1]
print("Top 5 ===> \n", [grades[i] for i in range(len(grades)) if grades[i][1]␣
↪in top5][::-1])
print("")
print("Between 25 and 75 percentiles ===> \n", grades[twentyfifth :␣
 ↪seventyfifth])
# [('student6', 43), ('student1', 45), ('student7', 45), ('student5', 48)]
print("")
bot5 = marks[:5]
print("Bottom 5 ===> \n", [grades[i] for i in range(len(grades)) if␣
 ↪grades[i][1] in bot5])
```

```
Top 5 ===>
 [('toh', 98), ('abe', 80), ('ram', 78), ('gor', 48), ('jim', 45), ('sam', 45)]

Between 25 and 75 percentiles ===>
 [('kim', 43), ('sam', 45), ('jim', 45), ('gor', 48)]

Bottom 5 ===>
 [('tom', 12), ('pam', 14), ('gig', 35), ('kim', 43), ('sam', 45), ('jim', 45)]
```

[25]:
```
s = int(input('How many students record you want to store?? '))
record={}
for i in range(1, s+1):
    #record[str(input("Name of {0} student: ".format(i)))]=int(input("Marks of␣
 ↪{0} student : ".format(i)))
    while  True:
        try:
            num=int(input("Marks of {0} student : ".format(i)))
            assert num < 100
            record[str(input("Name of {0} student: ".format(i)))]=num
            print('')
        except ValueError:
            print("Not an integer! Please enter an integer.")
        except AssertionError:
            print("Please enter an integer below 100")
        else:
            break
```

```
How many students record you want to store?? 1
Marks of 1 student : 10
Name of 1 student: a
```

Q5: Find the closest points

Consider you have given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q) Your task is to find 5 closest points(based on cosine distance) in S from P Cosine distance between two points (x,y) and (p,q) is defind as $cos^{-1}(\frac{(x \cdot p+y \cdot q)}{\sqrt(x^2+y^2) \cdot \sqrt(p^2+q^2)})$

Hint - If you write the formula correctly you'll get the distance between points (6,-7) and (3,-4) = 0.065

[3]:
```python
import math


# here S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(S, P):
    # write your code here
    dist_l={}
    p, q = P
    for i in S:
        x, y = i
        dist_l[i]=math.acos((x*p+y*q)/(math.sqrt(x*x + y*y)*math.sqrt(p*p +␣
↪q*q))) #dictionary of point and its distance

    i=0
    closest_points_to_p=[]
    for item in sorted(dist_l, key=dist_l.__getitem__): # sorting dictionary␣
↪according to values, not keys
        closest_points_to_p.append(item)
        i+=1
        if(i==5):
            break
    return closest_points_to_p  # its list of tuples

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)
points = closest_points_to_p(S, P)
print(points) #print the returned values
```

```
[(6, -7), (1, -1), (6, 0), (-5, -8), (-1, -1)]
```

Q6: Find Which line separates oranges and apples

consider you have given two set of data points in the form of list of tuples like

and set of line equations(in the string formate, i.e list of strings)

your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

[80]:
```python
Lines=["1x-1y-5"]

for l in Lines:
    # Split string by the occurrences of pattern. e.g. 1 -1 0 for Lines[0]
    # strip; removing the leading and trailing characters/spaces
    # extracting the coefficients alpha, beta and constant

    print(re.split('x|y', l)) # print: ['1', '-1', '-5']
    print(re.split('x|y', l)[1].strip()) # print: -1
    print(float(re.split('x|y', l)[1].strip())) # print: -1.0
    al, be, c = tuple(float(i.strip()) for i in re.split('x|y', l))

    print(type(al))
```

```
['1', '-1', '-5']
-1
-1.0
<class 'float'>
```

[82]:
```
import re

Red= [(1,1),(2,1),(4,2),(2,4),(-1,4)]
Blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]


def i_am_the_one(a, b, c):
    flag_R=[]
    for i in Red:
    #Checking if all the red points are seperated from the line
        x, y = i
        res = a*x + b*y
        if(res>c): # check all points for one side of line
            flag_R.append(True)
        else:
            flag_R.append(False)

    flag_B=[]
    for i in Blue:
    #Checking if all the blue points are seperated from the line
        x, y = i
        res = a*x + b*y
        if(res<c): # check all points for side opposite to red points
            flag_B.append(True)
        else:
            flag_B.append(False)


    #If all the red and blue points are seperated return Yes else No
    # all(flag_R) returns true when all elements in the list flag_R are ture
    # not any(flag_R) returns true when all elements in the list flag_R are␣
↪false
    if( (all(flag_R) and all(flag_B)) or (not any(flag_R) and not any(flag_B))␣
↪):
        print("Yes")
    else:
        print("No")
```

```
for l in Lines:
    # Split string by the occurrences of pattern. e.g. 1 -1 0 for Lines[0]
    # strip; removing the leading and trailing characters/spaces
    # extracting the coefficients alpha, beta and constant
    al, be, c = tuple(float(i.strip()) for i in re.split('x|y', l))
    i_am_the_one(al,be,c)
```

```
Yes
No
No
Yes
```

Q7: Filling the missing values in the specified formate

You will be given a string with digits and '_'(missing value) symbols you have to replace the '_' symbols as explained

for a given string with comma seprate values, which will have both missing values numbers like ex: "_, _, x, _, _, _" you need fill the missing values

Q: your program reads a string like ex: "_, _, x, _, _, _" and returns the filled sequence

Ex:

[74]:
```
S=  ["_,_,_,24","40,_,_,_,60","80,_,_,_,_","_,_,30,_,_,_,50,_,_"]

for j in range(len(S)):
    list_S = S[j].strip().split(",")
    print(list_S)
```

```
['_', '_', '_', '24']
['40', '_', '_', '_', '60']
['80', '_', '_', '_', '_']
['_', '_', '30', '_', '_', '_', '50', '_', '_']
```

[21]:
```
def lastSeen_To_current(lastSeen, current, divisor, val1, val2):
    current+=1 # local variable current
    if divisor == 1:
        pass # pass is needed for empty code block to not produce an␣
 ↪IndentationError
    else:
        d = int((val1+val2)/divisor)
        for i in range(lastSeen, current):
            list_S[i] = d # updating list_S from lastseen to current
    return

S=  ["_,_,_,24","40,_,_,_,60","80,_,_,_,_","_,_,30,_,_,_,50,_,_"]
```

```
for j in range(len(S)):
    list_S = S[j].strip().split(",") # e.g. ['_', '_', '_', '24'] for j==0
    # strip: Remove spaces at the beginning and at the end of the string
    start = 0
    len_S = len(list_S)
    count = 1
    lastSeen = 0 # index of last seen number other than 0

    # converting input to integer: assigning 0 to blank places
    for i in range(len_S):
        if list_S[i] == "_":
            list_S[i] = 0
        else:
            list_S[i] = int(list_S[i])

    while(start < len_S):
        if list_S[start] != 0:
            current = start
            divisor = current - lastSeen + 1
            lastSeen_To_current(lastSeen, current, divisor, list_S[lastSeen],␣
 ↪list_S[current])
            lastSeen = current # assigning current to lastseen after updating␣
 ↪list_S
        if count == len_S:
            if list_S[start] == 0:
                current = len_S - 1
                divisor = current - lastSeen + 1
                lastSeen_To_current(lastSeen, current, divisor,␣
 ↪list_S[lastSeen], list_S[current])
        start += 1
        count +=1
    print('Input {0}: "{1}"'.format(j+1,S[j]))
    print("Output {0}: {1}".format(j+1,list_S), end ="\n")
    print()
```

```
Input 1: "_,_,_,24"
Output 1: [6, 6, 6, 6]

Input 2: "40,_,_,_,60"
Output 2: [20, 20, 20, 20, 20]

Input 3: "80,_,_,_,_"
Output 3: [16, 16, 16, 16, 16]

Input 4: "_,_,30,_,_,_,50,_,_"
Output 4: [10, 10, 12, 12, 12, 12, 4, 4, 4]
```

Q8: Find the conditional probabilities

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns 1. the first column F will contain only 5 uniques values (F1, F2, F3, F4, F5) 2. the second column S will contain only 3 uniques values (S1, S2, S3)

Ex:

[73]:
```
def compute_conditional_probabilites(A):
    all_F = []
    all_S = []
    len_A = len(A)
    for i in range(len_A):
        all_F.append(A[i][0])
        all_S.append(A[i][1])

    cnt_F=Counter(all_F) # e.g.: Counter({'F2': 3, 'F1': 2, 'F3': 2, 'F4': 2,␣
 ↪'F5': 1})
    cnt_S=Counter(all_S) # e.g.: Counter({'S1': 4, 'S2': 3, 'S3': 3})


    for item_f in cnt_F:
        for item_s in cnt_S:
            cnt_F_S=0;
            for i in range(len_A):
                if(A[i][0]==item_f and A[i][1]==item_s):
                    cnt_F_S+=1
            print("P("+item_f+"|"+item_s+") = {0}/{1}".
 ↪format(cnt_F_S,cnt_S[item_s]))
        print("")


A =␣
 ↪[['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4',


compute_conditional_probabilites(A)
```

```
Counter({'F2': 3, 'F1': 2, 'F3': 2, 'F4': 2, 'F5': 1})
P(F1|S1) = 1/4
P(F1|S2) = 1/3
P(F1|S3) = 0/3

P(F2|S1) = 1/4
P(F2|S2) = 1/3
P(F2|S3) = 1/3
```

```
P(F3|S1) = 0/4
P(F3|S2) = 1/3
P(F3|S3) = 1/3

P(F4|S1) = 1/4
P(F4|S2) = 0/3
P(F4|S3) = 1/3

P(F5|S1) = 1/4
P(F5|S2) = 0/3
P(F5|S3) = 0/3
```

Q9: Given two sentances S1, S2

You will be given two sentances S1, S2 your task is to find

Ex:

[72]:
```python
def string_features(s1, s2):
    list_s1=s1.split()
    list_s2=s2.split()

    comm_s1 = sorted(list(set(list_s1).intersection(list_s2)), reverse=True)
    #comm_s1 = sorted(list(set(list_s1) and set(list_s2)), reverse=True) #␣
 ↪'and' do not means intersection

    diff_s1 = sorted(list(set(list_s1) - set(list_s2)), reverse=True)
    diff_s2 = sorted(list(set(list_s2) - set(list_s1)), reverse=True)
    #diff = [w1 for w1 in list_s1 if w1 not in list_s2]


    print(f"Words both in S1 and S2 : {comm_s1}")
    print(f"Words in S1 but not in S2 are : {diff_s1}")
    print(f"Words in S2 but not in S1 are : {diff_s2}")
    return

S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
string_features(S1, S2)
```

```
Words both in S1 and S2 : ['will', 'values', 'uniques', 'the', 'only',
'contain', 'column']
Words in S1 but not in S2 are : ['first', 'F', '5']
Words in S2 but not in S1 are : ['second', 'S', '3']
```

Q10: Find log loss of dataframe

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns

a. the first column Y will contain interger values
b. the second column Y_{score} will be having float values Your task is to find the value of $f(Y, Y_{score}) = -1 * \frac{1}{n} \Sigma_{foreach Y, Y_{score} pair}(Y log10(Y_{score}) + (1 - Y) log10(1 - Y_{score}))$ here n is the number of rows in the matrix
$\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + ... + (1 \cdot log_{10}(0.8) + 0 \cdot log_{10}(0.2)))$

[4]:
```
import math

# you can free to change all these codes/structure
def compute_log_loss(A):

    loss = 0
    for i in range(len(A)):
        y = A[i][0]
        y_s = A[i][1]
        loss += -(y*math.log10(y_s)+(1-y)*math.log10(1-y_s))/8
    return loss


A = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1,␣
↪0.8]]
loss = compute_log_loss(A)
print(loss)
```

```
0.42430993457031635
```

[10]:
```
# list within list cannot be extended by .extend().
tt = [0.5,0.6]
bb = [[]] * len(tt)
print(bb)
# extending every list within list instead index is mentioned.
_ = bb[1].extend(tt)
print(bb)
```

```
[[], []]
[[0.5, 0.6], [0.5, 0.6]]
```

[11]:
```
# correct way ==========================================
tt = [0.5,0.6]
bb = [[]] * len(tt)
print(bb)
bb[0] = bb[0] +[0.25564,0.24566]
bb[1] = bb[1] +[0.95564,0.94566]
print(bb)
```

```
[[], []]
[[0.25564, 0.24566], [0.95564, 0.94566]]
```

[72]:
```python
# program to print "prime twins/pairs" upto n.

def prime_finder(n):
    primes = []
    for possibleprime in range(2, n + 1):
        isprime = True
        for num in range(2, possibleprime):
            if possibleprime % num == 0:
                isprime = False
        if isprime:
            primes.append(possibleprime)
    return primes

def twin_prime(n):
    prime_list = prime_finder(n)
    count = 0
    for i in prime_list:
        if i + 2 in prime_list:
            print("{} and {} are prime twins".format(i,i+2))

twin_prime(100)
```

```
3 and 5 are prime twins
5 and 7 are prime twins
11 and 13 are prime twins
17 and 19 are prime twins
29 and 31 are prime twins
41 and 43 are prime twins
59 and 61 are prime twins
71 and 73 are prime twins
```

[8]:
```python
#Write a program to swap two numbers using bitwise operator.
#In bitwise operator 1 is converted to binary than x=0000

x=int(input('Enter the value of x:'))
y=int(input('Enter the value of y:'))
print('Before swaping the value of x is {} and y is {}'.format(x,y))
# bin(x) => 0b110000 (0b is binary code) => 48
# bin(y) => 0b1100010 (0b is binary code) => 98
x_ = x^y #in xor operation both are same it will gives zero , otherwise gives 1
'''
    0110000 (0 added before to 110000)
    1100010
=>  1010010
'''
# bin(x) => 0b1010010 (0b is binary code)
```

```
y_f = x_^y
# bin(y) => 0b110000 (0b is binary code)

x_f = x_^y_f
# bin(x) => 0b1100010 (0b is binary code)

print('After swaping the value of x is {} and y is {}'.format(x_f, y_f))
```

```
Enter the value of x:48
Enter the value of y:98
Before swaping the value of x is 48 and y is 98
After swaping the value of x is 98 and y is 48
```

[13]:
```
#check the Given string is Alphabet or not

# using Conditional operator
# ASCII 65-90 = Upper case(A-Z)
# ASCII 97-122= lower case (a-z)
# .ord(); function returns the number representing the unicode code of a␣
 ↪specified character.
char=input('Enter the character : ')
if ord(char)>=65 and ord(char)<=90 or ord(char)>=97 and ord(char)<=122:
    print(char,"is Alphabetic")
else:
    print(char,"is not Alphabetic")


#using isalpha() function
char=input('Enter the Character:')
if char.isalpha():
    print(char,"is Alphabetic")
else:
    print(char,"is Not Alphabetic")
```

```
Enter the character : 4
4 is not Alphabetic
Enter the Character:a
a is Alphabetic
```

[ ]:
```
```