1.Python Assignment

January 6, 2024

Python: without numpy or sklearn

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

[1]:
```python
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input examples



# you can free to change all these codes/structure
# here A and B are list of lists
A = [[1, 2],
     [3, 4]]
B = [[1, 2, 3, 4, 5],
     [5, 6, 7, 8, 9]]

def matrix_mul(A, B):
    # write your code
    rowA, colA, rowB, colB = len(A), len(A[0]), len(B), len(B[0])
    result_shape = (len(A), len(B[0]))
    result = [[0 for _ in range(result_shape[1])]for _ in␣
 ↪range(result_shape[0])]
    if colA == rowB:
        for i in range(rowA):
            for j in range(colB):
                for k in range(colA):
                    result[i][j] += A[i][k] * B[k][j]

        return result
    return "Not possible"

result = matrix_mul(A, B)
print(result)
```

```
[[11, 14, 17, 20, 23], [23, 30, 37, 44, 51]]
```

Q2: 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.

[2]:
```python
from random import uniform
import random
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input examples

def weights_of_each_element(A):
    sums = sum(A)
    for i in range(len(A)):
        A[i] = (A[i] / sums) * A[i]
    return A

# you can free to change all these codes/structure
def pick_a_number_from_list(A):
    # your code here for picking an element from with the probability␣
 ↪propotional to its magnitude
    sums = sum(A)
    minm, maxm = min(A), max(A)
    weights = weights_of_each_element(A.copy())
    num = random.choices(A, cum_weights=weights, k=len(weights))[0]
    return num

def sampling_based_on_magnitude(A):
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        print(number)
#          break

A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
sampling_based_on_magnitude(A)
```

```
79
28
28
100
79
79
79
100
79
79
27
100
```

```
79
79
79
79
28
79
100
79
79
79
79
79
27
79
100
79
79
100
27
79
79
79
79
100
79
28
28
79
79
79
28
28
79
79
79
100
28
79
79
28
79
28
79
100
28
79
79
79
```

```
79
28
79
79
79
79
79
79
100
100
79
79
79
79
79
79
79
79
79
79
79
79
100
79
79
79
100
79
79
27
100
100
79
100
79
79
27
79
28
```

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 #

[3]:
```python
import re
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input examples

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    result = ''
    for i in range(len(String)):
        if String[i].isdigit():
            result += '#'
    return result # modified string which is after replacing the # with digits
```

[4]:
```
String = '#2a$#b%c%561#'
replace_digits(String)
```

[4]:
```
'####'
```

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, 35, 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

[5]:
```
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input examples
students=['student1','student2','student3','student4','student5','student6','student7','studen
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
def percentile(percentile, A):
    sorted_data = list(sorted(A.values()))
    index = (percentile / 100) * len(sorted_data) - 1
    if index.is_integer():
        return sorted_data[index]
    else:
        lower_index = int(index)
        upper_index = lower_index +1
        lower_val = sorted_data[lower_index]
        upper_val = sorted_data[upper_index]
        interpolated_value = index - lower_index
        final_val = (1 - interpolated_value) * lower_val + interpolated_value *␣
 ↪upper_val
        return final_val

# you can free to change all these codes/structure
```

```
def display_dash_board(students, marks):
    # write code for computing top top 5 students
    student_dict = dict()
    for student, mark in zip(students, marks):
        student_dict[student] = mark
    top_5_students = {key:val for key, val in sorted(student_dict.items(), key=␣
 ↪lambda kv: kv[1], reverse=True)}
    print("Top 5 students")
    count = 0
    for key, val in top_5_students.items():
        print(key, val)
        if count == 4:
            break
        count += 1
    # write code for computing top least 5 students
    least_5_students = {key:val for key, val in sorted(student_dict.items(),␣
 ↪key= lambda kv: kv[1])}
    print()
    print("Least 5 students")
    count = 0
    for key, val in least_5_students.items():
        print(key, val)
        if count == 4:
            break
        count += 1

    percentile_25, percentile_75 = percentile(25, student_dict), percentile(75,␣
 ↪student_dict)

    print()
    print('students_within_25_and_75')
    students_within_25_and_75 = dict()
    count = 0
    for key, val in least_5_students.items():
        if val > percentile_25 and val < percentile_75:
            students_within_25_and_75[key] = val
            print(key, val)
            if count == 4:
                break
            count += 1

    return top_5_students, least_5_students, students_within_25_and_75

top_5_students, least_5_students, students_within_25_and_75 =␣
 ↪display_dash_board(students, marks)
# print(# those values)
```

```
Top 5 students
student8 98
student10 80
student2 78
student5 48
student7 47

Least 5 students
student3 12
student4 14
student9 35
student6 43
student1 45

students_within_25_and_75
student9 35
student6 43
student1 45
student7 47
student5 48
```

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 $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

[6]:
```python
import math

# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input examples
# you can free to change all these codes/structure

def cosine_dist(pt1, pt2):
    x, y = pt1
    p, q = pt2
    numerator = x*p + y*q
    denominator = (x ** 2 + y ** 2) ** 0.5 * (p ** 2 + q ** 2) ** 0.5
    value = numerator/denominator
    return math.acos(value)

# here S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(S, P):
    cosine_dict = dict()
    for coordinate in S:
        dist = cosine_dist(coordinate, P)
        cosine_dict[coordinate] = dist
    sorted_cosine_dict = {key:val for key,val in sorted(cosine_dict.items(),␣
 ↪key=lambda kv:kv[1])}
    closest_5_points = list()
    count = 0
    for key, val in sorted_cosine_dict.items():
        closest_5_points.append(key)
        count += 1
        if count == 5:
            break
    return closest_5_points  # 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

[7]:
```
import math, re
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input strings
def check_signs(values, x, y, c):
    prev_sign = 0
    for coordinates in values:
        xc, yc = coordinates
        sign = 1 if (xc * x + yc * y + c) > 0 else -1
        if prev_sign == 0:
            prev_sign = 1 if sign > 0 else -1
        elif prev_sign != sign:
            return False
    return True
# this code is taken from chatgpt
def get_coefficents(equation):
    # Define a regular expression pattern to match coefficients
    pattern = re.compile(r'([-+]?\d+)[xy]')
    # Use the findall method to extract matches
    matches = pattern.findall(equation)

    # Ensure that both x and y coefficients are present
    if len(matches) < 2:
        raise ValueError("Equation format is not valid")

    # Extract coefficients from the matches
    a, b = map(int, matches)  # Coefficients for x and y
    c = re.search(r'([-+]?\d+)\s*', equation)  # Constant term
    c = int(c.group(1))

    return a, b, c

# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
    # your code
    x, y, c = get_coefficents(line)
    if check_signs(red, x, y, c) and check_signs(blue, x, y, c):
        return 'yes'
    return 'no'

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+1=0","1x-1y+0","1x+0y-3","0x+1y-0.5"]

for line in Lines:
    yes_or_no = i_am_the_one(Red, Blue, line)
    print(yes_or_no) # the returned value
```

```
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:

[8]:
```
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input strings
def fill_value(lst, val, start, end):
    for i in range(start, end):
        lst[i] = val
    return lst

# you can free to change all these codes/structure
def curve_smoothing(string):
    # your code
    prev_val = 0
    count = 0
    start = 0
    splitted_string = string.split(',')
    result = list(range(len(splitted_string)))
    for i, val in enumerate(splitted_string):
        count += 1
        if val != '_':
            avg = (prev_val + int(val)) // count
            result = fill_value(result, avg, start, start+count)
            count = 1
            prev_val = avg
            start = i
        elif i == len(splitted_string) - 1:
            avg = prev_val // count
            result = fill_value(result, avg, start, start+count)
    result = map(str, result)
    return ','.join(result)

S=  "30,_,_,_,50,_"
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
16,16,16,16,8,8
```

Q8: Filling the missing values in the specified formate

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:

[9]:
```
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input strings

def prob(F, S, A):
    p_f = 0
    p_s = 0
    for row in A:
        if row[0] == F and row[1] == S:
            p_f += 1
        if row[1] == S:
            p_s += 1
    return p_f,p_s

# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
    # your code
    all_probs = list()
    f = set()
    s = set()
    for row in A:
        f.add(row[0])
        s.add(row[1])
    for fs in f:
        for ss in s:
            p_f, p_s = prob(fs, ss, A)
            print(f'P(F={fs}|S=={ss})={p_f}/{p_s}', end=' ')
        print()

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

compute_conditional_probabilites(A)
```

```
P(F=F1|S==S2)=1/3 P(F=F1|S==S1)=1/4 P(F=F1|S==S3)=0/3
P(F=F3|S==S2)=1/3 P(F=F3|S==S1)=0/4 P(F=F3|S==S3)=1/3
P(F=F2|S==S2)=1/3 P(F=F2|S==S1)=1/4 P(F=F2|S==S3)=1/3
P(F=F5|S==S2)=0/3 P(F=F5|S==S1)=1/4 P(F=F5|S==S3)=0/3
P(F=F4|S==S2)=0/3 P(F=F4|S==S1)=1/4 P(F=F4|S==S3)=1/3
```

Q9: Given two sentances S1, S2

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

Ex:

[10]:
```
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input strings
def common_word(S1, S2):
    s1 = dict()
    common = 0
    for word in S1.split(' '):
        s1[word] = s1.get(word, 0) + 1
    for word in S2.split(' '):
        if s1.get(word):
            common += 1
    return common

def uncommon_word(S1, S2):
    # this function returns all the words present in S1 but not in S2
    uncommon = list()
    s1 = dict()
    for word in S1.split(' '):
        s1[word] = s1.get(word, 0) + 1
    for word in S2.split(' '):
        if not s1.get(word):
            uncommon.append(word)
    return uncommon

# you can free to change all these codes/structure
def string_features(S1, S2):
    # your code
    a = common_word(S1, S2)
    b = uncommon_word(S2, S1)
    c = uncommon_word(S1, S2)
    return a, b, c

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

```
a, b, c 7 ['first', 'F', '5'] ['second', 'S', '3']
```

Q10: Given two sentances S1, S2

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 $f(Y,Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ here n is the number of rows in the matrix $\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + ... + (1 \cdot log_{10}(0.8) + 0 \cdot log_{10}(0.2)))$

[11]:
```
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
 ↪input strings
import math
```

```
# you can free to change all these codes/structure
def compute_log_loss(A):
    result = 0
    for val in A:
        y, y_score = val
        result += y * math.log10(y_score) + (1 - y) * math.log10(1 - y_score)
    loss = result * -1/len(A)
    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', loss)
```

```
loss 0.42430993457031635
```

```
[ ]:
```