1.Python Assignment

April 20, 2020

Python: without numpy or sklearn
Q1: Given two matrices please print the product of those two matrices

In [1]:
```
# 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 examp

A1 = [[1, 3, 4],
      [2, 5, 7],
      [5, 9, 6]]
B1   = [[1, 0, 0],
      [0, 1, 0],
      [0, 0, 1]]
A1B1 = [[1, 3, 4],
      [2, 5, 7],
      [5, 9, 6]]


A2   = [[1, 2],
      [3, 3]]
B2   = [[1, 2, 3, 4, 5],
      [5, 6, 7, 8, 9]]
A2B2 = [[11, 14, 17, 20, 23],
      [18, 24, 30, 36, 42]]

A3   = [[1, 2],
      [3, 4]]
B3   = [[1, 4],
      [5, 6],
      [7, 8],
      [9, 6]]
A3B3 = "Not possible"

# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(A, B):
    """ Assuming that bad matrices will not be given as input
```

```
        m x n
            n x p
        m x p matrix is the result
        """

        m = len(A)
        n = len(A[0])
        columns_A = n
        rows_B = len(B)
        p = len(B[0])

        if columns_A != rows_B:
            return "Not possible"


        C = []

        for i in range(m):
            C.append([0] * p)
            for j in range(p):
                s = 0
                for k in range(n):
                    s += A[i][k] * B[k][j]
                C[i][j] = s

        return C
```

```
In [2]: matrix_mul(A1, B1)
```

```
In [3]: matrix_mul(A2, B2)
```

```
In [4]: matrix_mul(A3, B3)
```

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.

https://stackoverflow.com/questions/16489449/select-element-from-array-with-probability-proportional-to-its-value

Idea is taken from this link

```
In [5]: from random import uniform
        # 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 examp


        # 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 i
            cumsum = 0
            cum_sum_array = []
            for i in A:
                cumsum += i
                cum_sum_array.append(cumsum)

            random_number = uniform(cum_sum_array[0], cum_sum_array[-1])

            if len(A) == 1:
                return A[0]
            if len(A) == 0:
                raise Exception("Empty list")

            index = None
            for i in range(0, len(A)):
                if random_number <= cum_sum_array[i]:
                    index = i
                    break

            return A[index]#selected_random_number

        A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
        values = []

        def sampling_based_on_magnitude():
            for i in range(1,int(1e6)):
                number = pick_a_number_from_list(A)
                values.append(number)

        sampling_based_on_magnitude()
```

```
In [6]: import pandas as pd

        pd.Series(values).value_counts()
```

```
Out[6]: 100    319750
        79     252251
        45     143848
        28      89260
        27      86351
        13      41626
        10      31875
        6       19192
        5       15846
        dtype: int64
```

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 #

```
In [7]: 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 examp

        # you can free to change all these codes/structure
        # String: it will be the input to your program

        digit = re.compile(r"[0-9]")
        non_digit = re.compile(r"[^0-9]")

        def replace_digits(s):
            # write your code
            s = non_digit.sub("", s)
            s = digit.sub("#", s)
            return s # modified string which is after replacing the # with digits

In [8]: test_strings = [("234", "###"),
        ("a2b3c4", "###"),
        ("abc", ""),
        ("#2a$#b%c%561#", "####")
        ]
        for test_string, expected_output in test_strings:
            assert replace_digits(test_string) == expected_output
```

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

```
In [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 examp
        Students=['student1','student2','student3','student4','student5','student6','student7'
        Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

        def get_top_5_students(Students, Marks):
            student_marks = list(zip(Students, Marks))
            student_marks.sort(key=lambda x: x[1], reverse=True)
            return student_marks[:5]

        def get_least_5_students(Students, Marks):
            student_marks = list(zip(Students, Marks))
```

```
        student_marks.sort(key=lambda x: x[1])
        return student_marks[:5]


    def get_students_within_25_and_75(Students, Marks):
        student_marks = list(zip(Students, Marks))
        student_marks.sort(key=lambda x: x[1])
        cumsum = 0
        n = len(Students)
        cumsum_values = []
        for student, mark in student_marks:
            cumsum += 1 / n
            cumsum_values.append((student, mark, cumsum))
        return [(student, mark, cumsum)
                for student, mark, cumsum in cumsum_values
                if cumsum >= .25 and cumsum < .75]

    # you can free to change all these codes/structure
    def display_dash_board(students, marks):
        # write code for computing top top 5 students
        top_5_students = get_top_5_students(Students, Marks)# compute this
        # write code for computing top least 5 students
        least_5_students = get_least_5_students(Students, Marks) # compute this
        # write code for computing top least 5 students
        students_within_25_and_75 = get_students_within_25_and_75(Students, Marks) # compu

        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(Stude

    def print_tuple(student_marks):
        if len(student_marks[0]) == 2:
            for student, mark in student_marks:
                print(student, mark)
        else:
            for student, mark, cumsum in student_marks:
                print(student, mark, cumsum)

    print("top 5 students")
    print_tuple(top_5_students)
    print("least 5 students")
    print_tuple(least_5_students)
    print("students within 25 and 75")
    print_tuple(students_within_25_and_75)
```

```
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 0.30000000000000004
student6 43 0.4
student1 45 0.5
student7 47 0.6
student5 48 0.7
```

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

```
In [10]: 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 exam
         # you can free to change all these codes/structure

         def cosine_distance(X, P):
             x, y = X
             p, q = P
             return math.acos((x * p + y * q)/ (math.sqrt(x**2 + y**2) * math.sqrt(p**2 + q**2

         # 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
             distances = [(s, cosine_distance(s, P)) for s in S]
             distances.sort(key=lambda x: x[1])
             return distances[:5]  # 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), 0.06512516333438509), ((1, -1), 0.14189705460416438), ((6, 0), 0.9272952180016123),
```

```
In [11]: for p, _ in points:
             print(p)

(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

```
In [12]: class Stack:
             def __init__(self):
                 self.stack = []

             def empty(self):
                 if len(self.stack) == 0:
                     return True
                 else:
                     return False

             def push(self, i):
                 self.stack.append(i)

             def pop(self):
                 if not self.empty():
                     return self.stack.pop()
                 else:
                     return None

             def get_list(self):
                 return self.stack

         "1x-1y-3"
         def get_line_equation(s):
             sign = Stack()
             variable = Stack()
             value = Stack()

             for i in s:
                 if i == 'x' or i == 'y':
                     variable.push(i)
                 elif i == '+':
```

```
                sign.push(1)
            elif i == '-':
                sign.push(-1)
            else:
                if i == '.':
                    value.push('.')
                else:
                    value.push(int(i))

            while not variable.empty():
                variable_ = variable.pop()

                if not value.empty():
                    value_ = float("".join(str(i) for i in value.get_list()))
                    while not value.empty():
                        value.pop()

                sign_value = sign.pop()
                if sign_value is not None:
                    value_ = value_ * sign_value
                if variable_ == 'x':
                    a = value_
                if variable_ == 'y':
                    b = value_

        if not value.empty():
            value_ = float("".join(str(i) for i in value.get_list()))
            while not value.empty():
                value.pop()

            sign_value = sign.pop()
            if sign_value is not None:
                value_ = value_ * sign_value
            c = value_

        return a, b, c


    get_line_equation("14x-13y-3.5")
```

```
Out[12]: (14.0, -13.0, -3.5)
```

https://math.stackexchange.com/questions/274712/calculate-on-which-side-of-a-straight-line-is-a-given-point-located

Idea is taken from this link. I take 2 points from the line and used the formula given in this link.

```
In [14]: 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 stri



def get_two_points(a, b, c):
    try:
        points = (0.0, -c/b), (1.0, (c-a)/b)
    except ZeroDivisionError:
        points = (-c/a, 0.0), (-c/a, 0.0)
    return points

def get_d_value(a, b, p):
    x, y = p
    x1, y1 = a
    x2, y2 = b
    d = (x-x1)*(y2-y1)-(y-y1)*(x2-x1)
    return d

def is_less_than_zero(x):
    return x < 0

def is_greater_than_zero(x):
    return x > 0


def get_sign(values):
    if len(values) < 0:
        raise Exception("Empty values")
    if values[0] < 0:
        if all(map(is_less_than_zero, values)):
            return -1
        else:
            return 0
    if values[0] > 0:
        if all(map(is_greater_than_zero, values)):
            return 1
        else:
            return 0
    if values[0] == 0:
        return 0


# you can free to change all these codes/structure
def i_am_the_one(reds,blues,line):

    a, b, c = get_line_equation(line)
```

```
        p1, p2 = get_two_points(a, b, c)
        red_d_values = []
        blue_d_values = []
        for red, blue in zip(reds, blues):
            red_d_values.append(get_d_value(p1, p2, red))
            blue_d_values.append(get_d_value(p1, p2, blue))

        return get_sign(blue_d_values) * get_sign(red_d_values)

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"]

for i in Lines:
    yes_or_no = i_am_the_one(Red, Blue, i)
    if yes_or_no < 0:
        print("YES")
    else:
        print("NO")
```

```
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:

```
In [15]: # 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 stri


         # you can free to change all these codes/structure
         def curve_smoothing(s):
             values = s.split(",")
             for i in range(len(values)):
                 if values[i] != "_":
                     values[i] = int(values[i])
```

```
    begin = 0
    last = len(values) - 1

    new_begin = 0
    while begin < last:
        if values[begin] == "_":
            i = begin
            while i < last and values[i] == "_":
                i += 1
            count = i - begin + 1
            number = values[i] // count
            new_begin = i
            while i >= begin:
                values[i] = number
                i -= 1
            begin = new_begin

        else: # number string
            i = begin

            i += 1
            while i < last and values[i] == "_":
                i += 1
            count = i - begin + 1

            if i <= last:
                if values[i] == "_":
                    number = values[begin] // count
                else:
                    number = (values[begin] + values[i]) // count

            else:
                number = values[begin] // (count - 1)
                i -= 1

            new_begin = i
            while i >= begin:
                values[i] = number
                i -= 1
            begin = new_begin

    return values

input_data = ["_,_,_,24",
# 6,6,6,6
"40,_,_,_,60",
# 20,20,20,20,20
"80,_,_,_,_",
```

```
        # 16,16,16,16,16
        "_,_,30,_,_,_,50,_,_"
        # 10,10,12,12,12,12,4,4,4
        ]

        for inp in input_data:
            smoothed_values= curve_smoothing(inp)
            print(smoothed_values)
```

```
[6, 6, 6, 6]
[20, 20, 20, 20, 20]
[16, 16, 16, 16, 16]
[10, 10, 12, 12, 12, 12, 4, 4, 4]
```

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:

```
In [16]: # 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 stri



         # you can free to change all these codes/structure
         def compute_conditional_probabilites(A):
             # your code
             # print the output as per the instructions
             m = 5
             n = 3
             count = []
             for i in range(m):
                 count.append([])
                 for _ in range(n):
                     count[i].append(0)
             s_count = [0, 0, 0]
             for i in range(len(A)):
                 p = int(A[i][0].replace("F", ""))
                 q = int(A[i][1].replace("S", ""))

                 count[p-1][q-1] += 1
                 s_count[q-1] += 1


             for i in range(m):
```

```
                for j in range(n):

                    print(f"P(F=F{i+1}|S==S{j+1})={count[i][j]}/{s_count[j]}, ", end="")
                print()

        A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S

        compute_conditional_probabilites(A)
```

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

Q9: Given two sentances S1, S2
You will be given two sentances S1, S2 your task is to find
Ex:

```
In [17]: # 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 stri

         # you can free to change all these codes/structure
         def string_features(S1, S2):
             # your code
             s1 = set(S1.split())
             s2 = set(S2.split())
             a = len(s1.intersection(s2))
             b = s1.difference(s2)
             c = s2.difference(s1)
             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)
         print(list(b))
         print(list(c))
```

```
7
['F', '5', 'first']
['S', 'second', '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
$\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + ... + (1 \cdot log_{10}(0.8) + 0 \cdot log_{10}(0.2)))$

In [18]:
```
# 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 stri


# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code
    n = len(A)
    loss = (-1 * 1 /n *
            sum([y * math.log10(y_score) + (1 - y) * math.log10(1 - y_score)
            for y, y_score in 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)
```

```
0.42430993457031635
```