Graded Questions: (Hypothesis Testing)

1. Concept of Hypothesis Testing – I

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Concept of Hypothesis Testing – II

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Coding Practice Graded:

**Valid email ID**

**Description**

Consider that email IDs are supposed to be for the following format:  
username@website.extension.  
Here, there are three conditions to keep in mind:  
1. The username can only contain characters 0-9, a-z and A-Z.  
2. The website name can contain only characters a-z  
3. The extension can have 2 or 3 alphabets(a-z).  
  
Given an email ID, you have to determine if it is valid or not.  
  
Sample Input:  
prerna@upgrad.com  
  
Sample Output:  
valid

Solution:

import re

def checkmail(email):

#complete the function

#the function should return the strings "invalid" or "valid" based on the email ID entered

regex = '(^[a-zA-Z0-9]+@[a-z]+\.[a-z]{2,3}$)'

if(re.search(regex,email)):

return "valid"

else:

return "invalid"

email=input()

print(checkmail(email))

**Flatten a list**

**Description**

Given a nested list, write python code to flatten the list.  
Note: The input list will strictly have 2 levels, i.e. the list will be of the form [[1,2],[3,4]]. Inputs like [1,[2,3]] and [[1,[2,3],4],5] are not applicable.  
  
**For example:**If the input list is :  
[[1,2,3],[4,5],[6,7,8,9]]  
Then the output should be:  
[1,2,3,4,5,6,7,8,9]

Solution:

import ast,sys

input\_str = sys.stdin.read()

input\_list = ast.literal\_eval(input\_str)

flat\_list = [item for sublist in input\_list for item in sublist]

print(flat\_list)

**Squares**

**Description**

Given a list of positive integers, you have to find numbers divisible by 3 and replace them with their squares.   
For example, consider the list below:  
Input: [1,2,3,4,5,6]  
The output for the above list would be: [1,2,9,4,5,36]. Because 3 and 6 were divisible by 3, these numbers were replaced with their squares.

Solution:

import ast,sys

input\_str = sys.stdin.read()

input\_list = ast.literal\_eval(input\_str)

for index, item in enumerate(input\_list):

if item%3 == 0:

input\_list[index] = item\*\*2

print(input\_list)

**A weird sum**

**Description**

Write a program that computes the value of n+nn+nnn+nnnn with a given digit as the value of n.  
For example, if n=9 , then you have to find the value of 9+99+999+9999.

Solution:

n=input()

n1=int("%s" %n)

n2=int("%s%s" %(n,n))

n3=int("%s%s%s" %(n,n,n))

n4=int("%s%s%s%s" %(n,n,n,n))

print(n1+n2+n3+n4)

**Frequent Letters**

**Description**

Given a string, you have to find the first n most frequent characters in it.

You have to print these n letters in alphabetically sorted order.

The input will contain two lines, the first line will contain a string and the second line will contain the letter n.

The output should be a list of n most frequent letters in alphabetically sorted order.

**Note:**If there are two letters with the same frequency, then the alphabetically preceding alphabet should be picked first. (For example, in "aabbccc", if n=2, then output list would have c and a.)

Sample Input:

ddddaacccb

3

Sample Output:

['a', 'c', 'd']

In the above example, the order of frequencies is : d>c>a>b

Here, d,c and a are three most frequent characters which are displayed in alphabetically sorted order.

Solution:

string=input()

n=int(input())

#write your code here

from collections import Counter

def mostFrequentLetter(string, n):

ctr = Counter(c for c in string.lower() if c.isalpha())

return ''.join(sorted(x[0] for x in ctr.most\_common(n)))

print(list(mostFrequentLetter(string,n)))

**2D array**

**Description**

Write Python code which takes 2 numbers x and y as input and generates a 2-dimensional numpy array where value in the i-th row and j-th column of the array should be (i+j)/2.  
**Note:**i=0,1,...x-1 and j=0,1....,y-1  
The input will have two lines with x and y respectively.  
The output should be a 2D numpy array.  
Sample Input:  
3  
4  
Sample Output:

[[0. 0.5 1. 1.5]

[0.5 1. 1.5 2. ]

[1. 1.5 2. 2.5]]

Solution:

x=int(input())

y=int(input())

from numpy import zeros

a = zeros([x,y])

for row in range(x):

for col in range(y):

a[row][col]= (row+col)/2

print(a)

1. Industry Demonstration of Hypothesis Testing

# Summary

So what did you learn in this session?

1. **T-distribution**:
   * A T-distribution is used whenever the standard deviation of the population is unknown
   * The degrees of freedom of a T-distribution is equal to sample size n - 1
   * For sample size ≥ 30, the T-distribution becomes the same as the normal distribution
   * The output values and results of both t-test and z-test are same for sample size ≥ 30
2. **Two-sample mean test - paired**:
   * It is used when your sample observations are from the same individual or object
   * During this test, you are testing the same subject twice
3. **Two-sample mean test - unpaired**:
   * During this test, you are not testing the same subject twice
   * It is used when your sample observations are independent

1. **Two-sample proportion test**:
   * It is used when your sample observations are categorical, with two categories
   * It could be True/False, 1/0, Yes/No, Male/Female, Success/Failure, etc.
2. **A/B Testing:**
   * A/B testing is a direct industry application of the two-sample proportion test
   * It is a widely used process in digital companies in the ecommerce, manufacturing and advertising domains
   * It provides a way to test two different versions of the same element and see which one performs better