**ADVANCED PROGRAMMING ASSIGNMENT\_7 -SUBMITTED BY SAMUEL DEVDAS**

1. Write a function that counts how many concentric layers a rug.

**Examples**

count\_layers([

"AAAA",

"ABBA",

"AAAA"

]) ➞ 2

count\_layers([

"AAAAAAAAA",

"ABBBBBBBA",

"ABBAAABBA",

"ABBBBBBBA",

"AAAAAAAAA"

]) ➞ 3

count\_layers([

"AAAAAAAAAAA",

"AABBBBBBBAA",

"AABCCCCCBAA",

"AABCAAACBAA",

"AABCADACBAA",

"AABCAAACBAA",

"AABCCCCCBAA",

"AABBB BBBBAA",

"AAAAAAAAAAA"

]) ➞ 5

Ans.

def count\_layers(inp):

mid=len(inp)//2

return(mid+1)

count\_layers([

"AAAAAAAAAAA",

"AABBBBBBBAA",

"AABCCCCCBAA",

"AABCAAACBAA",

"AABCADACBAA",

"AABCAAACBAA",

"AABCCCCCBAA",

"AABBB BBBBAA",

"AAAAAAAAAAA"

])

2. There are many different styles of music and many albums exhibit multiple styles. Create a function that takes a list of musical styles from albums and returns how many styles are unique.

**Examples**

unique\_styles([

"Dub,Dancehall",

"Industrial,Heavy Metal",

"Techno,Dubstep",

"Synth-pop,Euro-Disco",

"Industrial,Techno,Minimal"

]) ➞ 9

unique\_styles([

"Soul",

"House,Folk",

"Trance,Downtempo,Big Beat,House",

"Deep House",

"Soul"

]) ➞ 7

Ans.

def unique\_styles(inp):

style\_str=''

for i in inp:

style\_str+=','+i

stripcomma=style\_str.lstrip(',')

split=stripcomma.split(',')

return len(set(split))

unique\_styles([

"Soul",

"House,Folk",

"Trance,Downtempo,Big Beat,House",

"Deep House",

"Soul"

])

3. Create a function that finds a target number in a list of prime numbers. Implement a binary search algorithm in your function. The target number will be from 2 through 97. If the target is prime then return "yes" else return "no".

**Examples**

primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

is\_prime(primes, 3) ➞ "yes"

is\_prime(primes, 4) ➞ "no"

is\_prime(primes, 67) ➞ "yes"

is\_prime(primes, 36) ➞ "no"

4. Create a function that takes in n, a, b and returns the number of positive values raised to the nth power that lie in the range [a, b], inclusive.

**Examples**

power\_ranger(2, 49, 65) ➞ 2

# 2 squares (n^2) lie between 49 and 65, 49 (7^2) and 64 (8^2)

power\_ranger(3, 1, 27) ➞ 3

# 3 cubes (n^3) lie between 1 and 27, 1 (1^3), 8 (2^3) and 27 (3^3)

power\_ranger(10, 1, 5) ➞ 1

# 1 value raised to the 10th power lies between 1 and 5, 1 (1^10)

power\_ranger(5, 31, 33) ➞ 1

power\_ranger(4, 250, 1300) ➞ 3

Ans.

import numpy as np

def power\_ranger(exp,lower,upper):

lower\_pow=round(np.power(lower,1/exp))

upper\_pow=round(np.power(upper,1/exp))

return len(range(lower\_pow,upper\_pow+1))

power\_ranger(4, 250, 1300)

5. Given a number, return the difference between the maximum and minimum numbers that can be formed when the digits are rearranged.

**Examples**

rearranged\_difference(972882) ➞ 760833

# 988722 - 227889 = 760833

rearranged\_difference(3320707) ➞ 7709823

# 7733200 - 23377 = 7709823

rearranged\_difference(90010) ➞ 90981

# 91000 - 19 = 90981

Ans.

def rearranged\_difference(inp):

str\_inp=[i for i in str(inp)]

str\_inp.sort()

minimum=''

for i in str\_inp:

minimum+=i

str\_inp.sort(reverse=True)

maximum=''

for i in str\_inp:

maximum+=i

return int(maximum)-int(minimum)

rearranged\_difference(3320707)