**Python Advance Programming Assignment-07**

**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", "AABBBBBBBAA", "AAAAAAAAAAA" ]) ➞ 5

**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

**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"

In [3]:

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]

**def** is\_prime(lop,num):

count**=**0

**for** i **in** lop:

**if** num**>**1:

**for** j **in** (2,i):

**if** num**%j**==0:

count**+=**1

**if** count**==**1:

**return** 'yes'

**return** 'No'

In [4]:

is\_prime(primes, 3)

Out[4]:

'yes'

In [5]:

is\_prime(primes, 4)

Out[5]:

'No'

In [6]:

is\_prime(primes, 67)

Out[6]:

'yes'

In [7]:

is\_prime(primes, 36)

Out[7]:

'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) ➞ 4

In [38]:

**def** power\_ranger(n,a,b):

value**=**n**\*\***n

**if** value **in** range(a**+**1,b):

**return** n

**return** 1

In [39]:

power\_ranger(5,31,33)

Out[39]:

1

In [40]:

power\_ranger(4,250,1300)

Out[40]:

4

**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

In [16]:

**def** rearranged\_difference(digits):

string**=**str(digits)

string**=**[i **for** i **in** string]

minimum**=**int(''**.**join(sorted(string)))

maximum**=**int(''**.**join(sorted(string,reverse**=True**)))

**return** maximum**-**minimum

In [17]:

rearranged\_difference(972882)

Out[17]:

760833

In [18]:

rearranged\_difference(3320707)

Out[18]:

7709823

In [19]:

rearranged\_difference(90010)

Out[19]:

90981