**Question1. Write a function that stutters a word as if someone is struggling to read it. The first two letters are repeated twice with an ellipsis ... and space after each, and then the word is pronounced with a question mark ?.**

**Examples**

**stutter("incredible") ➞ "in... in... incredible?"**

**stutter("enthusiastic") ➞ "en... en... enthusiastic?"**

**stutter("outstanding") ➞ "ou... ou... outstanding?"**

**Hint :-Assume all input is in lower case and at least two characters long.**

**ANS:-**

**def stutter(word):**

**return (2\*(word[:2]+'... '))+word+'?'**

**word = input('Enter word : ')**

**print(stutter(word))**

**Enter word : outstanding**

**ou... ou... outstanding?**

**Question 2.Create a function that takes an angle in radians and returns the corresponding angle in degrees rounded to one decimal place.**

### Examples

**radians\_to\_degrees(1) ➞ 57.3**

**radians\_to\_degrees(20) ➞ 1145.9**

**radians\_to\_degrees(50) ➞ 2864.8**

**ANS:-**

***# Function for convertion***

**def radians\_to\_degrees(radian):**

**pi = 3.14159**

***#formula***

**degree = radian \* (180/pi)**

**return degree**

**radian = float(input('Enter the Radian : '))**

**print("degree =",(radians\_to\_degrees(radian)))**

**Enter the Radian : 20**

**degree = 1145.9165581759555**

**Question 3. In this challenge, establish if a given integer num is a Curzon number. If 1 plus 2 elevated to num is exactly divisible by 1 plus 2 multiplied by num, then num is a Curzon number.**

**Given a non-negative integer num, implement a function that returns True if num is a Curzon number, or False otherwise.**

### Examples

**is\_curzon(5) ➞ True**

**# 2 \*\* 5 + 1 = 33**

**# 2 \* 5 + 1 = 11**

**# 33 is a multiple of 11**

**is\_curzon(10) ➞ False**

**# 2 \*\* 10 + 1 = 1025**

**# 2 \* 10 + 1 = 21**

**# 1025 is not a multiple of 21**

**is\_curzon(14) ➞ True**

**# 2 \*\* 14 + 1 = 16385**

**# 2 \* 14 + 1 = 29**

**# 16385 is a multiple of 29**

**ANS:-**

**def checkIfCurzonNumber(n):**

**power, product = 0, 0**

***# Find 2\*\*n + 1***

**power = pow(2, n) + 1**

***# Find 2\*n + 1***

**product = 2 \* n + 1**

***# Check for divisibility***

**if (power % product == 0):**

**print(n, "is Curzon Number")**

**else:**

**print(n, "is not a Curzon Number")**

**n = int(input('Enter a number : '))**

**checkIfCurzonNumber(n)**

**Enter a number : 33**

**33 is Curzon Number**

**Question 4.Given the side length x find the area of a hexagon.**

****

### Examples

**area\_of\_hexagon(1) ➞ 2.6**

**area\_of\_hexagon(2) ➞ 10.4**

**area\_of\_hexagon(3) ➞ 23.4**

**ANS:-**

***# area of a Hexagon***

***# Area = (3 √3(n\*n) ) / 2***

**import math**

**def area\_of\_hexagon(s):**

**return ((3 \* math.sqrt(3) \* (sideLength \* sideLength)) / 2);**

***#length of a side.***

**sideLength = float(input('Enter the length : '))**

**print("Area:","{0:.4f}".format(area\_of\_hexagon(sideLength)))**

**Enter the length : 3**

**Area: 23.3827**

**Question 5.Create a function that returns a base-2 (binary) representation of a base-10 (decimal) string number. To convert is simple: ((2) means base-2 and (10) means base-10) 010101001(2) = 1 + 8 + 32 + 128.**

**Going from right to left, the value of the most right bit is 1, now from that every bit to the left will be x2 the value, value of an 8 bit binary numbers are (256, 128, 64, 32, 16, 8, 4, 2, 1).**

### Examples

**binary(1) ➞ "1"**

**# 1\*1 = 1**

**binary(5) ➞ "101"**

**# 1\*1 + 1\*4 = 5**

**binary(10) ➞ "1010"**

**# 1\*2 + 1\*8 = 10**

**ANS:-**

***# Function to convert Decimal number***

***# to Binary number***

**def decimalToBinary(n):**

**return bin(n).replace("0b", "")**

**for i in range(0,50):**

**print(decimalToBinary(i))**

**0**

**1**

**10**

**11**

**100**

**101**

**110**

**111**

**1000**

**1001**

**1010**

**1011**

**1100**

**1101**

**1110**

**1111**

**10000**

**10001**

**10010**

**10011**

**10100**

**10101**

**10110**

**10111**

**11000**

**11001**

**11010**

**11011**

**11100**

**11101**

**11110**

**11111**

**100000**

**100001**

**100010**

**100011**

**100100**

**100101**

**100110**

**100111**

**101000**

**101001**

**101010**

**101011**

**101100**

**101101**

**101110**

**101111**

**110000**

**110001**