# Assignment 16 Solutions

#### 1.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.

In [1]:

**def** stutterWord():

in\_string **=** input('Enter the Word :')

out\_string **=** in\_string**.**replace(in\_string[0:2],((in\_string[0:2]**+**'... ')**\***2)**+** in\_string[0:2]) **+**'?'

print(f'{in\_string} ➞ {out\_string}')

**for** i **in** range(3):

stutterWord()

Enter the Word :incredible

incredible ➞ in... in... incredible?

Enter the Word :enthusiastic

enthusiastic ➞ en... en... enthusiastic?

Enter the Word :outstanding

outstanding ➞ ou... ou... outstanding?

#### 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

In [2]:

**import** math

**def** radianToDegree():

in\_num **=** int(input('Enter the angle in Radians: '))

out\_num **=** (180**/**math**.**pi)**\***in\_num

print(f'{in\_num} radian(s) ➞ {out\_num:.1f} degrees')

**for** x **in** range(3):

radianToDegree()

Enter the angle in Radians: 1

1 radian(s) ➞ 57.3 degrees

Enter the angle in Radians: 20

20 radian(s) ➞ 1145.9 degrees

Enter the angle in Radians: 50

50 radian(s) ➞ 2864.8 degrees

#### 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`

In [3]:

**def** checkCurzon():

in\_num **=** int(input("Enter a number: "))

**if** (pow(2,in\_num)**+**1)**%**((2**\***in\_num)**+**1) **==** 0:

print(f'{in\_num} is a Curzon Number')

**else**:

print(f'{in\_num} is Not a Curzon Number')

**for** x **in** range(4):

checkCurzon()

Enter a number: 5

5 is a Curzon Number

Enter a number: 10

10 is Not a Curzon Number

Enter a number: 14

14 is a Curzon Number

Enter a number: 12

12 is Not a Curzon Number

#### 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

In [4]:

**import** math

**def** areaOfHexagon():

in\_num **=** int(input('Enter the side length of a Hexagon: '))

out\_num **=** ((3**\***math**.**sqrt(3))**/**2)**\***(pow(in\_num,2))

print(f'Area for Hexagon of sidelength {in\_num} ➞ {out\_num:.1f}')

**for** x **in** range(3):

areaOfHexagon()

Enter the side length of a Hexagon: 1

Area for Hexagon of sidelength 1 ➞ 2.6

Enter the side length of a Hexagon: 2

Area for Hexagon of sidelength 2 ➞ 10.4

Enter the side length of a Hexagon: 3

Area for Hexagon of sidelength 3 ➞ 23.4

#### 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`

In [5]:

**def** getBinary():

in\_num **=** int(input("Enter a Number: "))

out\_num **=** bin(in\_num)**.**replace('0b','')

print(f'Binary of {in\_num} ➞ {out\_num}')

**for** x **in** range(3):

getBinary()

Enter a Number: 1

Binary of 1 ➞ 1

Enter a Number: 5

Binary of 5 ➞ 101

Enter a Number: 10

Binary of 10 ➞ 1010