Python Basic Programming Assignment 16

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 stutter(word):
    return word[:2] + "..." + " " + word[:2] + "..." + " " + word + "?"
print(stutter('incredible'))
print(stutter('enthusiastic'))
print(stutter('outstanding'))
```

```
in... in... incredible?
en... en... enthusiastic?
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.

In [2]:
```
def radians_to_degrees(radians):
    return round(radians * 180 / 3.14, 1)
print(radians_to_degrees(0))
print(radians_to_degrees(3.14))
print(radians_to_degrees(3.14/2))
print(radians_to_degrees(3.14/4))
print(radians_to_degrees(3.14*2))
```

```
0.0
180.0
90.0
45.0
360.0
```

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.

In [4]:
```
def is_curzon(num):
    if (1 + 2**num) % (1 + 2*num) == 0:
        return True
    return False
print(is_curzon(14))
print(is_curzon(19))
print(is_curzon(20))
```

```
True
False
False
```

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

In [5]:
```
import math
def find_hexagon_area(x):
    return (3 * math.sqrt(3) / 2) * x**2
# Test the function
print(find_hexagon_area(4))
```

```
41.569219381653056
```

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.

In [6]:
```
def binary(num):
    # Convert num to integer
    num = int(num)
    # Initialize empty list to store binary digits
    binary_digits = []
    # Divide num by 2 until it becomes 0
    while num > 0:
        # Append the remainder of num divided by 2 to the list
        binary_digits.append(num % 2)
        # Divide num by 2 and store the result
        num = num // 2
    # Reverse the list of binary digits
    binary_digits = binary_digits[::-1]
    # Convert list of binary digits to a string and return it
    return ''.join(map(str, binary_digits))
print(binary('128')) # Output: 10000000
print(binary('64')) # Output: 1000000
print(binary('32')) # Output: 100000
print(binary('16')) # Output: 10000
print(binary('8')) # Output: 1000
print(binary('4')) # Output: 100
print(binary('2')) # Output: 10
print(binary('1')) # Output: 1
```

```
10000000
1000000
100000
10000
1000
100
10
1
```

```
In [ ]:
```