Question1

Create a function that takes an integer and returns a list from 1 to the given number, where:

* If the number **can be divided** evenly by 4, amplify it by 10 (i.e. return 10 times the number).
* If the number **cannot be divided** evenly by 4, simply return the number.

**Examples**

amplify(4) ➞ [1, 2, 3, 40]

amplify(3) ➞ [1, 2, 3]

amplify(25) ➞ [1, 2, 3, 40, 5, 6, 7, 80, 9, 10, 11, 120, 13, 14, 15, 160, 17, 18, 19, 200, 21, 22, 23, 240, 25]

**Notes**

* The given integer will always be equal to or greater than 1.
* Include the number (see example above).
* To perform this problem with its intended purpose, try doing it with list comprehensions. If that's too difficult, just solve the challenge any way you can.

def amplify(in\_num):

out\_list = []

for ele in range(1,in\_num+1):

if ele%4 == 0:

out\_list.append(ele\*10)

else:

out\_list.append(ele)

print(f'{in\_num} ➞ {out\_list}')

amplify(4)

amplify(3)

amplify(25)

Question2

Create a function that takes a list of numbers and return the number that's unique.

**Examples**

unique([3, 3, 3, 7, 3, 3]) ➞ 7

unique([0, 0, 0.77, 0, 0]) ➞ 0.77

unique([0, 1, 1, 1, 1, 1, 1, 1]) ➞ 0

**Notes**

Test cases will always have exactly one unique number while all others are the same.

def unique(in\_list):

out\_num = ''

for ele in set(in\_list):

if in\_list.count(ele) == 1:

out\_num = ele

print(f'{in\_list} ➞ {out\_num}')

unique([3, 3, 3, 7, 3, 3])

unique([0, 0, 0.77, 0, 0])

unique([0, 1, 1, 1, 1, 1, 1, 1])

Question3

Your task is to create a Circle constructor that creates a circle with a radius provided by an argument. The circles constructed must have two getters getArea() (PI*r^2) and getPerimeter() (2*PI\*r) which give both respective areas and perimeter (circumference).

For help with this class, I have provided you with a Rectangle constructor which you can use as a base example.

**Examples**

circy = Circle(11)

circy.getArea()

# Should return 380.132711084365

circy = Circle(4.44)

circy.getPerimeter()

# Should return 27.897342763877365

**Notes**

Round results up to the nearest integer.

import math

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def getArea(self):

print(f'Radius ➞ {round(math.pi\*self.radius\*self.radius)}')

def getPerimeter(self):

print(f'Perimeter ➞ {round(2\*math.pi\*self.radius)}')

circy = Circle(11)

circy.getArea()

circy = Circle(4.44)

circy.getPerimeter()

Question4

Create a function that takes a list of strings and return a list, sorted from shortest to longest.

**Examples**

sort\_by\_length(["Google", "Apple", "Microsoft"])

➞ ["Apple", "Google", "Microsoft"]

sort\_by\_length(["Leonardo", "Michelangelo", "Raphael", "Donatello"])

➞ ["Raphael", "Leonardo", "Donatello", "Michelangelo"]

sort\_by\_length(["Turing", "Einstein", "Jung"])

➞ ["Jung", "Turing", "Einstein"]

**Notes**

All test cases contain lists with strings of *different* lengths, so you won't have to deal with multiple strings of the same length.

def sort\_by\_length(in\_list):

print(sorted(in\_list,key=len))

sort\_by\_length(["Google", "Apple", "Microsoft"])

sort\_by\_length(["Leonardo", "Michelangelo", "Raphael", "Donatello"])

sort\_by\_length(["Turing", "Einstein", "Jung"])

Question5

Create a function that validates whether three given integers form a **Pythagorean triplet**. The sum of the squares of the *two smallest integers* must equal the square of the *largest number* to be validated.

**Examples**

is\_triplet(3, 4, 5) ➞ True

# 3² + 4² = 25

# 5² = 25

is\_triplet(13, 5, 12) ➞ True

# 5² + 12² = 169

# 13² = 169

is\_triplet(1, 2, 3) ➞ False

# 1² + 2² = 5

# 3² = 9

**Notes**

Numbers may not be given in a sorted order.

def is\_triplet(a,b,c):

if ((a\*\*2+b\*\*2) == (c\*\*2)):

print(f'{a,b,c} ➞ {True}')

else:

print(f'{a,b,c} ➞ {False}')

is\_triplet(3, 4, 5)

is\_triplet(3, 4, 5)

is\_triplet(1, 2, 3)