Question1

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

1. If the number can be divided evenly by 4, amplify it by 10 (i.e. return 10 times the

number).

2. 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&#39;s too difficult, just solve the challenge any way you can.

Ans1

def amplify(num):

num\_list = list(range(1, num+1))

amplified\_list = [n\*10 if n % 4 == 0 else n for n in num\_list]

return amplified\_list

print(amplify(4)) # [1, 2, 3, 40]

print(amplify(3)) # [1, 2, 3]

print(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]

Question2

Create a function that takes a list of numbers and return the number that&#39;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.

Ans2

def unique(lst):

freq = {}

for num in lst:

freq[num] = freq.get(num, 0) + 1

for num in lst:

if freq[num] == 1:

return num

print(unique([3, 3, 3, 7, 3, 3])) # Output: 7

print(unique([0, 0, 0.77, 0, 0])) # Output: 0.77

print(unique([0, 1, 1, 1, 1, 1, 1, 1])) # Output: 0

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() (PIr^2) and

getPerimeter() (2PI\*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.

Ans3

import math

class Circle:

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

self.radius = radius

def getArea(self):

return round(math.pi \* self.radius\*\*2)

def getPerimeter(self):

return round(2 \* math.pi \* self.radius)

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&#39;t have to deal with

multiple strings of the same length.

Ans4

def sort\_by\_length(lst):

return sorted(lst, key=len)

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

# Output: ["Apple", "Google", "Microsoft"]

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

# Output: ["Raphael", "Leonardo", "Donatello", "Michelangelo"]

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

# Output: ["Jung", "Turing", "Einstein"]

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.

Ans5

def is\_triplet(n1, n2, n3):

squares = [n1\*\*2, n2\*\*2, n3\*\*2]

squares.sort()

# Check if the sum of the squares of the first two numbers equals the square of the third number

if squares[0] + squares[1] == squares[2]:

return True

else:

return False

print(is\_triplet(3, 4, 5)) # Output: True

print(is\_triplet(13, 5, 12)) # Output: True

print(is\_triplet(1, 2, 3)) # Output: False