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's too difficult, just solve the challenge any way you can.

Ans.1

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def amplify(list\_range):

amplify\_number = 4

print([i if i%amplify\_number!=0 else i\*10 for i in range(1,list\_range+1)])

try:

amplify(4)

amplify(3)

amplify(25)

lg.info("""Class amplify() has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

Ans.2

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def unique(l):

unique1 = list(set(l))

min = int()

max = int()

for i in unique1:

numb = l.count(i)

if numb == 1:

min = i

if numb > 1:

max = i

print("unique number is --->", min)

try:

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

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

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

lg.info("""function unique() has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

class circle():

import math

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

self.radius = radius

def getArea(self):

print("Area is :--> ",math.pi\*(self.radius\*\*2) )

def getPerimeter(self):

print("Perimeter is :--> ",math.pi\*(self.radius\*2) )

try:

circy = circle(11)

circy.getArea()

circy = circle(4.44)

circy.getPerimeter()

lg.info("""Class circle() has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

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

sorted\_list = sorted(my\_list, key=len)

print(sorted\_list)

try:

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

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

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

lg.info("""Class circle() has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def is\_triplet(a1,a2,a3):

group = [a1,a2,a3]

maximum = max(group)

group.remove(maximum)

if group[0]\*\*2 + group[1]\*\*2 == maximum\*\*2:

print(True)

return True

else:

print(False)

return False

try:

is\_triplet(3, 4, 5)

is\_triplet(13, 5, 12)

is\_triplet(1, 2, 3)

lg.info("""Class circle() has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass