**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.**

def amplify(n):  
 a=[]  
 for i in range(1,n+1):  
 if(i%4==0) :a.append(i\*10)  
 else: a.append(i)  
 return a  
n=int(input())  
l=amplify(n)  
print(l)

**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.**

**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.**

class circle():  
 def \_\_init\_\_(self,r):  
 self.radius=r  
 def area(self):  
 return self.radius\*\*2\*3.14  
 def perimeter(self):  
 return 2\*self.radius\*3.14  
circy=circle(float(input()))  
print(circy.area())  
print(circy.perimeter())

**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"]

### Note 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(l):  
 l=sorted(l,key=len)  
 return l  
l=list(input().split())  
s=sort\_by\_length(l)  
print(s)

**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): m=max(a,b,c) if((a!=m)and(b!=m)): if((a\*\*2+b\*\*2)==(m\*\*2)): return True else: return False elif((a!=m and c!=m)): if((a\*\*2+c\*\*2)==(m\*\*2)): return True else: return False else: if((b\*\*2+c\*\*2)==(m\*\*2)): return True else: return False a=int(input()) b=int(input()) c=int(input()) print(is\_triplet(a,b,c))