

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.

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()` ($\text{PI}r^2$) and `getPerimeter()` ($2\text{PI}r$) 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.

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.

Question 5

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
```

```
#  $3^2 + 4^2 = 25$ 
```

```
#  $5^2 = 25$ 
```

```
is_triplet(13, 5, 12) → True
```

```
#  $5^2 + 12^2 = 169$ 
```

```
#  $13^2 = 169$ 
```

```
is_triplet(1, 2, 3) → False
```

```
#  $1^2 + 2^2 = 5$ 
```

```
#  $3^2 = 9$ 
```

Notes

Numbers may not be given in a sorted order.

Solution

Question 1

To solve this problem, we can create a list comprehension that loops from 1 to the given number, and for each number, we check if it is divisible by 4. If it is divisible by 4, we multiply it by 10 and add it to the list. If it is not divisible by 4, we add it directly to the list. Finally, we return the resulting list.

python

```
def amplify(num):  
    return [(i*10 if i%4 == 0 else i) for i in range(1, num+1)]
```

Question 2

To solve this problem, we can count the occurrence of each number in the list and return the number that occurs only once.

```
python
def unique(lst):
    for num in lst:
        if lst.count(num) == 1:
            return num
```

Question 3

To solve this problem, we can create a `Circle` class with a constructor that takes a `radius` argument. We can then define two methods, `getArea()` and `getPerimeter()`, that calculate the area and perimeter of the circle using the formulae πr^2 and $2\pi r$, respectively. We can use the `math` library to access the value of π .

```
python
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def getArea(self):
        return math.ceil(math.pi * self.radius ** 2)

    def getPerimeter(self):
        return math.ceil(2 * math.pi * self.radius)
```

Question 4

To solve this problem, we can simply use the `sorted()` function with a key function that returns the length of each string.

```
python
def sort_by_length(lst):
    return sorted(lst, key=len)
```

Question 5

To solve this problem, we can first sort the three numbers in ascending order. We can then check if the sum of the squares of the two smallest numbers is equal to the square of the largest number.

python

```
def is_triplet(a, b, c):  
    nums = [a, b, c]  
    nums.sort()  
    return nums[0]**2 + nums[1]**2 == nums[2]**2
```