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() (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.

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.

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.

Solutions:

Question 1: Here's the implementation of the `amplify` function:

```python

def amplify(num):

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

```

Explanation:

- The function takes an integer `num` as input.

- It uses a list comprehension to iterate over the range from 1 to `num+1`.

- If the current number `i` is divisible evenly by 4 (`i % 4 == 0`), it multiplies `i` by 10 (`i \* 10`).

- Otherwise, it keeps the number as is (`i`).

- The resulting list is returned.

Question 2: Here's the implementation of the `unique` function:

```python

def unique(lst):

for num in lst:

if lst.count(num) == 1:

return num

```

Explanation:

- The function takes a list `lst` as input.

- It iterates over each number `num` in the list.

- It checks if the count of `num` in the list (`lst.count(num)`) is equal to 1.

- If a number is found with a count of 1, it means it is unique, so that number is returned.

Question 3: Here's the implementation of the `Circle` class:

```python

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)

```

Explanation:

- The `Circle` class is defined with a constructor that takes a `radius` argument.

- The `radius` is stored as an instance variable.

- The class has two methods: `getArea()` and `getPerimeter()`.

- The `getArea()` method calculates the area of the circle using the formula `PI \* radius^2` and rounds the result to the nearest integer.

- The `getPerimeter()` method calculates the perimeter (circumference) of the circle using the formula `2 \* PI \* radius` and rounds the result to the nearest integer.

Question 4: Here's the implementation of the `sort\_by\_length` function:

```python

def sort\_by\_length(lst):

return sorted(lst, key=len)

```

Explanation:

- The function takes a list of strings `lst` as input.

- It uses the `sorted()` function to sort the list based on the length of the strings.

- The `key=len` parameter specifies that the sorting should be done based on the length of the strings.

- The sorted list is returned.

Question 5: Here's the implementation of the `is\_triplet` function:

```python

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

nums = [a, b, c]

nums.sort()

return nums[0] \*\* 2 + nums[1] \*\* 2 == nums[2] \*\* 2

```

Explanation:

- The function takes three integers `a`, `b`, and `c` as input.

- The function creates a list `nums` containing the three numbers.

- The list is sorted in ascending order using the `sort()` method.

- The function checks if the sum of the squares of the two smallest numbers (`nums[0] \*\* 2 + nums[1] \*\* 2`) is equal to the square of the largest number (`nums[2] \*\* 2`).

- If the condition is satisfied, it returns `True`; otherwise, it returns `False`.