Programming_Assingment24

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.

In [1]:

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

In [2]:

```
amplify(4)
```

Out[2]:

```
[1, 2, 3, 40]
```

In [3]:

```
amplify(3)
```

Out[3]:

```
[1, 2, 3]
```

In [4]:

```
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'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.

In [5]:

```
def unique(lst):
    s = list(set(lst))# give us unique value
    for i in s:
        if lst.count(i) == 1:
            return i
```

In [6]:

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

Out[6]:

```
7
```

In [7]:

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

Out[7]:

```
0.77
```

In [8]:

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

Out[8]:

```
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.

In [9]:

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

    def getArea(self):
        return round(self.radius**2*3.14)

    def getPerimeter(self):
        return round(2*self.radius*3.14)
```

In [10]:

```
circy = Circle(11)
circy.getArea()
```

Out[10]:

```
380
```

In [11]:

```
circy = Circle(4.44)
circy.getPerimeter()
```

Out[11]:

```
28
```

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.

In [12]:
```
def sort_by_length(lst):
    return sorted(lst, key = len)
```

In [13]:
```
sort_by_length(['Google', 'Apple', 'Microsoft'])
```

Out[13]:
```
['Apple', 'Google', 'Microsoft']
```

In [14]:
```
sort_by_length(['Leonardo', 'Michelangelo', 'Raphael', 'Donatello'])
```

Out[14]:
```
['Raphael', 'Leonardo', 'Donatello', 'Michelangelo']
```

In [15]:
```
sort_by_length(['Turing', 'Einstein', 'Jung'])
```

Out[15]:
```
['Jung', 'Turing', 'Einstein']
```

In [ ]:

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

$3^2 + 4^2 = 25$

$5^2 = 25$

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.

In [16]:

```
def is_triplet(a,b,c):
    lst = []
    lst.extend((a,b,c))
    lst = sorted(lst)

    if lst[0]**2 + lst[1]**2 == lst[2]**2:
        print('Triplets')
        return True
    else:
        print("not triplet")
        return False
```

In [17]:

```
is_triplet(3, 4, 5)
```

```
Triplets
```

Out[17]:

```
True
```

In [18]:

```
is_triplet(13, 5, 12)
Triplets
```

Out[18]:

```
True
```

In [19]:

```
is_triplet(1, 2, 3)
not triplet
```

Out[19]:

```
False
```