

1. 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:

- 1.The given integer will always be equal to or greater than 1.
- 2.Include the number (see example above).
- 3.To perform this problem with its intended purpose, try doing it with list

Ans:

```
In [31]: 1 def amplify(n):
          2     output = []
          3     for i in range(1,n+1):
          4         if i%4 == 0:
          5             output.append(i*10)
          6         else:
          7             output.append(i)
          8     print(f'amplify({n}) → {output}')
          9     amplify(4)
         10     amplify(3)
         11     amplify(25)
```

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]

2. 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:

```
In [32]: 1 def unique(num):
          2     output = ''
          3     for i in set(num):
          4         if num.count(i) == 1:
          5             output = i
          6     print(f'unique{num} → {output}')
          7     unique([3, 3, 3, 7, 3, 3])
          8     unique([0, 0, 0.77, 0, 0])
          9     unique([0, 1, 1, 1, 1, 1, 1, 1])
```

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

3. 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()` (πr^2) and `getPerimeter()` ($2\pi r$) 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.

Ans:

```
In [34]: 1 import math
          2 class Circle:
          3     def __init__(self, radius):
          4         self.radius = radius
          5     def getArea(self):
          6         print(f'{round(math.pi*self.radius*self.radius)}')
          7     def getPerimeter(self):
          8         print(f'{round(2*math.pi*self.radius)}')
          9 circy = Circle(11)
         10 circy.getArea()
         11 circy = Circle(4.44)
         12 circy.getPerimeter()

380
28
```

4. 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:

```
In [38]: 1 def sort_by_length(string):
2         print(f'sort_by_length({string})→{sorted(string,key=len)}')
3         sort_by_length(["Google", "Apple", "Microsoft"])
4         sort_by_length(["Leonardo", "Michelangelo", "Raphael", "Donatello"])
5         sort_by_length(["Turing", "Einstein", "Jung"])

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']
```

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.

Ans:

```
In [39]: 1 def is_triplet(a,b,c):
2         if ((pow(a,2)+pow(b,2)) == pow(c,2)):
3             print(f'is_triplet{a,b,c} → {True}')
4         else:
5             print(f'is_triplet{a,b,c} → {False}')
6         is_triplet(3, 4, 5)
7         is_triplet(3, 4, 5)
8         is_triplet(1, 2, 3)
```

`is_triplet(3, 4, 5) → True`

`is_triplet(3, 4, 5) → True`

`is_triplet(1, 2, 3) → False`