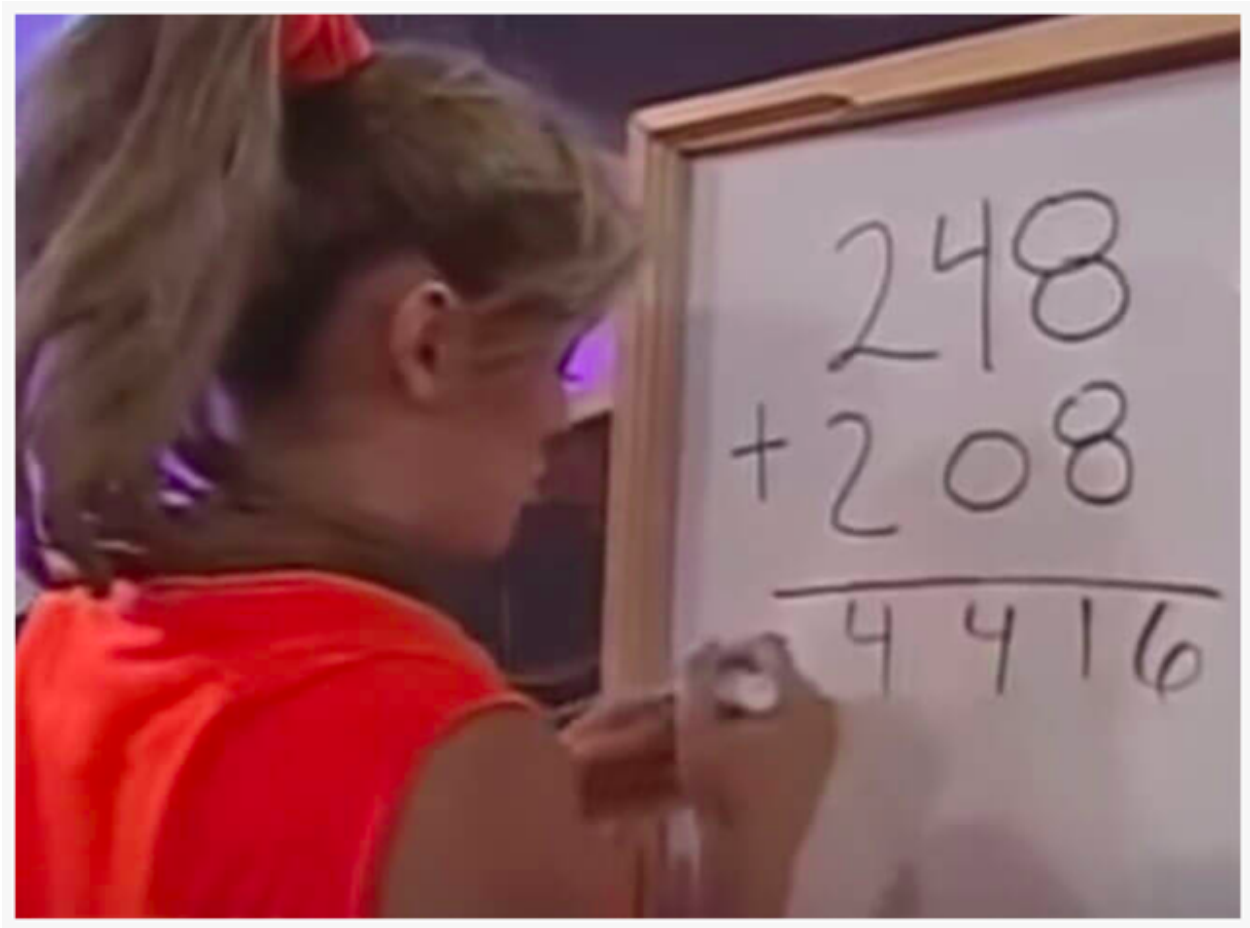
1. For this challenge, forget how to add two numbers together. The best explanation on what to do for this function is this meme:



Examples:
meme_sum(26, 39) ⟶ 515
2+3 = 5, 6+9 = 15
26 + 39 = 515
meme_sum(122, 81) ⟶ 1103
1+0 = 1, 2+8 = 10, 2+1 = 3
122 + 81 = 1103
meme_sum(1222, 30277) ⟶ 31499

Ans:

In [24]:
```
def meme_sum(num1,num2):
    a,b = str(num1),str(num2)
    add = ''
    while len(a) != len(b):
        if len(a) < len(b):
            a ='0'+a
        else:
            b='0'+b
    for i in range(len(a)):
        add += str(int(a[i])+int(b[i]))
    print(f'meme_sum{num1,num2} → {add}')
meme_sum(26, 39)
meme_sum(122, 81)
meme_sum(1222, 30277)
```

```
meme_sum(26, 39) → 515
meme_sum(122, 81) → 1103
meme_sum(1222, 30277) → 31499
```

2. Given an integer, create a function that returns the next prime. If the number is prime, return the number itself.

Examples:
next_prime(12) → 13
next_prime(24) → 29
next_prime(11) → 11
11 is a prime, so we return the number itself
Ans:

In [6]:
```
def next_prime(num):
    output = 0
    list1 =[]
    if num == 2 or num == 3:
        list1.append(num)
    for i in range(num,1000):
        if (i+1)%6==0 or (i-1)%6==0:
            list1.append(i)
    for i in list1:
        for n in range(2,i):
            if i%n == 0:
                list1.remove(i)
                break
    output = min(list1)
    print(f'next_prime({num}) → {output}')
next_prime(12)
next_prime(24)
next_prime(11)
```

```
next_prime(12) → 13
next_prime(24) → 29
next_prime(11) → 11
```

3. If a person traveled up a hill for 18mins at 20mph and then traveled back down the same path at 60mph then their average speed traveled was 30mph.

Write a function that returns the average speed traveled given an uphill time, uphill rate and a downhill rate. Uphill time is given in minutes. Return the rate as an integer (mph). No rounding is necessary.

Examples:
ave_spd(18, 20, 60) → 30

ave_spd(30, 10, 30) → 15
ave_spd(30, 8, 24) → 12
Ans:

In [7]:
```
def ave_spd(utime,urate,drate):
    distance = urate*(utime/60)
    dtime = distance/drate
    output = (2*distance)/((utime/60)+dtime)
    print(f'ave_spd{utime,urate,drate} → {int(output)}')
ave_spd(18, 20, 60)
ave_spd(30, 10, 30)
ave_spd(30, 8, 24)
```

```
ave_spd(18, 20, 60) → 30
ave_spd(30, 10, 30) → 15
ave_spd(30, 8, 24) → 12
```

4. The Kempner Function, applied to a composite number, permits to find the smallest integer greater than zero whose factorial is exactly divided by the number.

kempner(6) → 3
1! = 1 % 6 > 0
2! = 2 % 6 > 0
3! = 6 % 6 === 0

kempner(10) → 5
1! = 1 % 10 > 0
2! = 2 % 10 > 0
3! = 6 % 10 > 0
4! = 24 % 10 > 0
5! = 120 % 10 === 0

A Kempner Function applied to a prime will always return the prime itself.
kempner(2) → 2
kempner(5) → 5

Given an integer n, implement a Kempner Function.

Examples:
kempner(6) → 3
kempner(10) → 5
kempner(2) → 2

Ans:

In [10]:

```python
def kempner(num):
    def factorial(num):
        if num == 1:
            return 1
        else:
            return num * factorial(num-1)
    for i in range(1,num+1):
        if factorial(i)%num == 0:
            output = i
            break
    print(f'kempner({num}) ⟶ {output}')

kempner(6)
kempner(10)
kempner(5)
kempner(2)
```

```
kempner(6) ⟶ 3
kempner(10) ⟶ 5
kempner(5) ⟶ 5
kempner(2) ⟶ 2
```

5. You work in a factory, and your job is to take items from a conveyor belt and pack them into boxes. Each box can hold a maximum of 10 kgs. Given a list containing the weight (in kg) of each item, how many boxes would you need to pack all of the items?

Examples:
boxes([2, 1, 2, 5, 4, 3, 6, 1, 1, 9, 3, 2]) ⟶ 5
Box 1 = [2, 1, 2, 5] (10kg)
Box 2 = [4, 3] (7kg)
Box 3 = [6, 1, 1] (8kg)
Box 4 = [9] (9kg)
Box 5 = [3, 2] (5kg)

Ans:

In [11]:

```python
def boxes(in_list):
    in_list_copy = in_list.copy()
    output = []
    temp = []
    while True:
        if len(in_list) != 0:
            if sum(temp) <= 10:
                temp.append(in_list.pop(0))
            else:
                in_list.insert(0,temp.pop())
                output.append(temp)
                temp = []
        else:
            output.append(temp)
            temp = []
            break
    print(f'boxes({in_list_copy}) ⟶ {output} ⟶ {len(output)}')

boxes([2, 1, 2, 5, 4, 3, 6, 1, 1, 9, 3, 2])
boxes([5, 5, 5, 5, 5, 5, 2, 3, 4, 5, 6])
```

```
boxes([2, 1, 2, 5, 4, 3, 6, 1, 1, 9, 3, 2]) ⟶ [[2, 1, 2, 5], [4, 3], [6, 1, 1], [9], [3, 2]] ⟶ 5
boxes([5, 5, 5, 5, 5, 5, 2, 3, 4, 5, 6]) ⟶ [[5, 5], [5, 5], [5, 5], [2, 3, 4], [5, 6]] ⟶ 5
```