**1.Create a function that takes a number n (integer greater than zero) as an argument, and returns 2 if n is odd and 8 if n is even.**

You can only use the following arithmetic operators:  
addition of numbers +, subtraction of numbers -, multiplication of number \*, division of number /, and exponentiation \*\*.  
You are not allowed to use any other methods in this challenge (i.e. no if statements, comparison operators, etc).  
**Examples:**  
f(1) ➞ 2  
f(2) ➞ 8  
f(3) ➞ 2

In [1]:

**def** f(in\_num):

output **=** [8,2]

print(f'f({in\_num})➞ {output[in\_num**&**1]}')

f(1)

f(2)

f(3)

f(1)➞ 2

f(2)➞ 8

f(3)➞ 2

**2.Create a function that returns the majority vote in a list. A majority vote is an element that occurs > N/2 times in a list (where N is the length of the list).**

**Examples:**  
majority\_vote(["A", "A", "B"]) ➞ "A"  
majority\_vote(["A", "A", "A", "B", "C", "A"]) ➞ "A"  
majority\_vote(["A", "B", "B", "A", "C", "C"]) ➞ None

In [2]:

**def** majority\_vote(in\_list):

out\_list **=** **None**

**for** ele **in** set(in\_list):

**if** in\_list**.**count(ele) **>** (len(in\_list)**/**2):

out\_list **=** ele

print(f'majority\_vote({in\_list}) ➞ {out\_list}')

majority\_vote(["A", "A", "B"])

majority\_vote(["A", "A", "A", "B", "C", "A"])

majority\_vote(["A", "B", "B", "A", "C", "C"])

majority\_vote(['A', 'A', 'B']) ➞ A

majority\_vote(['A', 'A', 'A', 'B', 'C', 'A']) ➞ A

majority\_vote(['A', 'B', 'B', 'A', 'C', 'C']) ➞ None

**3.Create a function that takes a string txt and censors any word from a given list lst. The text removed must be replaced by the given character char.**

**Examples:**  
censor\_string("Today is a Wednesday!", ["Today", "a"], "-") ➞ "----- is - Wednesday!"  
censor\_string("The cow jumped over the moon.", ["cow", "over"], "\*"), "The \*\*\* jumped \*\*\*\* the moon.")  
censor\_string("Why did the chicken cross the road ?", ["Did", "chicken", "road"], "\*") ➞ "Why \*\*\* the \*\*\*\*\*\*\* cross the \*\*\*\*?"

In [3]:

**def** censor\_string(in\_string\_1, in\_string\_2, rep\_char):

out\_string **=** []

**for** ele **in** in\_string\_1**.**split():

**if** ele**.**title() **in** in\_string\_2 **or** ele**.**lower() **in** in\_string\_2 **or** ele**.**upper() **in** in\_string\_2:

out\_string**.**append(rep\_char**\***len(ele))

**else**:

out\_string**.**append(ele)

print(f'censor\_string({in\_string\_1}) ➞ {" "**.**join(out\_string)}')

censor\_string("Today is a Wednesday!", ["Today", "a"], "-")

censor\_string("The cow jumped over the moon.", ["cow", "over"], "\*")

censor\_string("Why did the chicken cross the road ?", ["Did", "chicken", "road"], "\*")

censor\_string(Today is a Wednesday!) ➞ ----- is - Wednesday!

censor\_string(The cow jumped over the moon.) ➞ The \*\*\* jumped \*\*\*\* the moon.

censor\_string(Why did the chicken cross the road ?) ➞ Why \*\*\* the \*\*\*\*\*\*\* cross the \*\*\*\* ?

**4.In mathematics a Polydivisible Number (or magic number) is a number in a given number base with digits abcde... that has the following properties:**

* Its first digit a is not 0.
* The number formed by its first two digits ab is a multiple of 2.
* The number formed by its first three digits abc is a multiple of 3.
* The number formed by its first four digits abcd is a multiple of 4. Create a function which takes an integer n and returns True if the given number is a Polydivisible Number and False otherwise.

**Examples:**  
`is\_polydivisible(1232) ➞ True

# 1 / 1 = 1

# 12 / 2 = 6

# 123 / 3 = 41

# 1232 / 4 = 308

is\_polydivisible(123220 ) ➞ False

# 1 / 1 = 1

# 12 / 2 = 6

# 123 / 3 = 41

# 1232 / 4 = 308

# 12322 / 5 = 2464.4 # Not a Whole Number

# 123220 /6 = 220536.333... # Not a Whole Number`

In [4]:

**def** is\_polydivisible(in\_num):

output **=** **False**

in\_num\_temp **=** str(in\_num)

**for** ele **in** range(len(in\_num\_temp)):

**if**(int(in\_num\_temp[:ele**+**1])**%**(ele**+**1) **==** 0):

output **=** **True**

**else**:

output **=** **False**

**break**

print(f'is\_polydivisible({in\_num}) ➞ {output}')

is\_polydivisible(1232)

is\_polydivisible(123220)

is\_polydivisible(1232) ➞ True

is\_polydivisible(123220) ➞ False

**5.. Create a function that takes a list of numbers and returns the sum of all prime numbers in the list.**

**Examples:**  
sum\_primes([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) ➞ 17  
sum\_primes([2, 3, 4, 11, 20, 50, 71]) ➞ 87  
sum\_primes([]) ➞ None

In [5]:

**def** sum\_primes(in\_list):

out\_string **=** []

**for** ele **in** in\_list:

**if** ele **in** [2,3]:

out\_string**.**append(ele)

**elif** ele **in** [6**\***n**-**1 **for** n **in** range(0,ele)] **or** ele **in** [6**\***n**+**1 **for** n **in** range(0,ele)]:

out\_string**.**append(ele)

**if** 1 **in** out\_string:

out\_string**.**remove(1)

print(f'sum\_prices({in\_list}) ➞ {sum(out\_string) **if** len(out\_string) **>** 0 **else** **None**}')

sum\_primes([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

sum\_primes([2, 3, 4, 11, 20, 50, 71])

sum\_primes([])

sum\_prices([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) ➞ 17

sum\_prices([2, 3, 4, 11, 20, 50, 71]) ➞ 87

sum\_prices([]) ➞ None