1)**find max value in a array?**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **23** | **65** | **12** | **87** | **12** | **343** | **098** | **12** | **76** | **312** |

Pre-condition: we need to define a variables a loop counter and variables max stored the maximum of all integers. Before starting the loop to find max value from b<array. We initialize b=0.pre -condition is true enter the first iteration true.

Post-condition: after the loop iteration stored the max values in the b=0.

Loop variant: after finding the max number of integer to check condition **if b<i: c**heck b less than equal to integer.

Loop invariant: finally print the initializing variables print(b) when condition is true, store the maximum all integers.

Pseudo code:

array= [23,65,87,12,343,098,12,76,312]

b=0

for i in aray:

if b<i:

    b=i

print(b)

2) **move zeros to the end of array?**

**A =**  **[2, 7, 6, 0, 9, 0, 1, 0, 0, 8, 0]**

pre-condition: write one function to name the element of array. pass the arguments If the given array is **[2, 7, 6, 0, 9, 0, 1, 0, 0, 8, 0],** it should be transform to **[2, 7, 6. 9, 1, 8, 0, 0, 0,**  .  we run another for loop from **k to** the given list's length.

Post-condition: variable `k` manage the index of the next available position

To check all element in stored in ‘k’.

Loop variant: first iterate the given list, to find and check all element in array. and check whether a current element is non-zero , if not then store next free position in the list the loop should not terminate until end the condition**. Num list[k] = i** to the **k = k + 1 .**

Loop invariant: move all 0's to the end of the list /remaining indices. to check range and find the length of elements.  **for i in range len(num\_list)**. Stored list[i]=0,and this return the value from the condition.

And define a variable inside the function to create object and define function. Then call the function to check overall statement program and we get the output.

**Pseudo code:**

def move\_zeros(num\_list):

k = 0

for i in num\_list:

if i:

num\_list[k] = i

k = k + 1

for i in range(k, len(num\_list)):

num\_list[i] = 0

return num\_list

num\_list = ***[2, 7, 6, 0, 9, 0, 1, 0, 0, 8, 0]***

order\_list = move\_zeros(num\_list)

**CONVERT ROMAN LETTERS TO INTEGERS**

Pre-condition: Before calling the roman\_to\_int() function, you must ensure that the input s meets the following conditions:

s is a non-empty string.

s contains only valid Roman numeral characters: 'I', 'V', 'X', 'L', 'C', 'D', and 'M'.

The Roman numerals in s are ordered according to the rules of Roman numerals, with larger numerals preceding smaller ones.

Post-condition: After the roman\_to\_int() function is called with a valid input s, the function returns an integer value that represents the decimal equivalent of the input Roman numeral.

Loop-variant: During the loop, the variable result keeps track of the sum of the decimal values of the Roman numerals processed so far. At the start of each iteration of the loop, result contains the sum of the decimal values of the Roman numerals processed up to the previous iteration.

Loop invariant: The loop processes one Roman numeral character at a time until all characters in the input string s have been processed. The loop variant is the number of characters remaining in s to be processed. At the start of each iteration of the loop, the loop variant decreases by 1, until it reaches 0 and the loop terminates.

Pseudo-code:

def roman\_to\_int(s):

roman\_dict = { 'I': 1, 'V': 5,'X': 10,'L': 50, 'C': 100,'D': 500,'M': 1000 }

result = 0

prev\_value = 0

for c in s:

value = roman\_dict[c]

if value > prev\_value:

result += value - 2 \* prev\_value

else:

result += value

prev\_value = value

return result

numeral = 'XIV'

integer = roman\_to\_int(numeral)

print(integer)