**NAME : SATHISH Vs**

**DATE : 26-03-2023**

**SUBJECT : ASSIGNMENT 1**

**FIND MAXIMUM VALUE IN ARRAY**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **4** | **5** | **6** | **7** | **8** | **2** | **3** | **5** | **2** |

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 i>ans.

Post-condition: after the loop iteration stored the max values in the ans variable.

Loop variant: after finding the max number of integer to check condition **if i>ans:**

Loop invariant: finally print the initializing variables print(ans) when condition is true, store the maximum all integer in ans variable .

Pseudo code:

array=[1,4,5,6,7,8,2,3,5,2]

ans=array[0]

for i in array:

if i>ans:

ans=i

print(ans)

**MOVE ZEROS TO THE END OF ARRAY**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 0 | 4 | 0 | 1 | 0 | 5 | 0 |

pre-condition: lst is a list of integers.

Post-condition: All the zeros in lst have been moved to the end of the list, and the order of the non-zero elements has been preserved

Loop variant :

At the start of each iteration of the outer loop, all elements from lst[0] to lst[i-1] have been processed and moved to their final positions in the list.

At the start of each iteration of the inner loop, lst[i] is a zero element that has not yet been processed.

At the end of each iteration of the inner loop, lst[i] has been swapped with the first non-zero element after it (if any), or lst[i] is already at the end of the list.

The order of the non-zero elements in the list has been preserved throughout the algorithm.

Loop invariant:

At the start of each iteration of the loop, all elements from the start of the list up to the current index have been processed and moved to their final positions in the list.

**Pseudo code:**

lst = [3, 0, 4, 0, 1, 0, 5, 0]

i = 0

while i < len(lst):

if lst[i] == 0:

j = i + 1s

while j < len(lst) and lst[j] == 0:

j += 1

if j == len(lst):

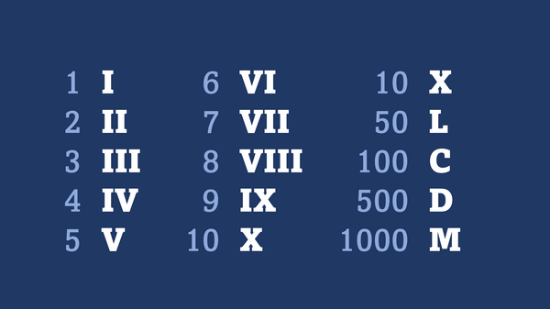
break

lst[i], lst[j] = lst[j], lst[i]

i +=1

print(lst)

**CONVERT ROMAN LETTERS TO INTEGERS**

****

Precondition: 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.

Postcondition: 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 Invariant: 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 Variant: 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.

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)