

# Taller 1 Métodos Numéricos

*Nombre:* Luis Enrique Pérez Señalín

## Resolver los 3 ejercicios de csacademy

### 1. Ejercicio de xplora

<https://csacademy.com/ieeextreme-practice/task/xplora/>

```
import json

# Datos de entrada
# Paso 1: Leer el número de entradas y los datos JSON
N = int(input())
entries = []
for _ in range(N): entries.append(input())

# Procesar los datos
author_citations = {}

for entry in entries:
    data = json.loads(entry)
    citing_paper_count = data['citing_paper_count']
    for author in data['authors']['authors']:
        author_citations[author['full_name']] = []

for entry in entries:
    data = json.loads(entry)
    citing_paper_count = data['citing_paper_count']
    for author in data['authors']['authors']:
        author_name = author['full_name']
        author_citations[author_name].append(citing_paper_count)

# Calcular el h-index de cada autor
def calculate_h_index(citations):
```

```

citations.sort(reverse=True)
h_index = 0
for index, n_citations in enumerate(citations):
    # Compara que la cantidad de articulos mínimo tenga
    # la misma cantidad de citaciones
    if index + 1 <= n_citations:
        h_index = index + 1
    else:
        break
return h_index

author_h_index = {}
for author in author_citations: author_h_index[author] = calculate_h_index(author_citation

# Paso 4: Ordenar los autores por h-index y por nombre alfabético en caso de empate
sorted_authors = sorted(author_h_index.items(), key=lambda authors: (-authors[1],authors[0]
#Key, lambda authors, retorna un array, [author, value], primero es por el value, entonces
# ponemos authors[1], pero los pone de menor a mayor, por eso agregamos el - al inicio, y
# si tienen el mismo value, osea el h_index, usa el valor de authors[0], que son los nombres

# Paso 5: Imprimir los resultados
for author, h_index in sorted_authors:
    print(f"{author} {h_index}")

```

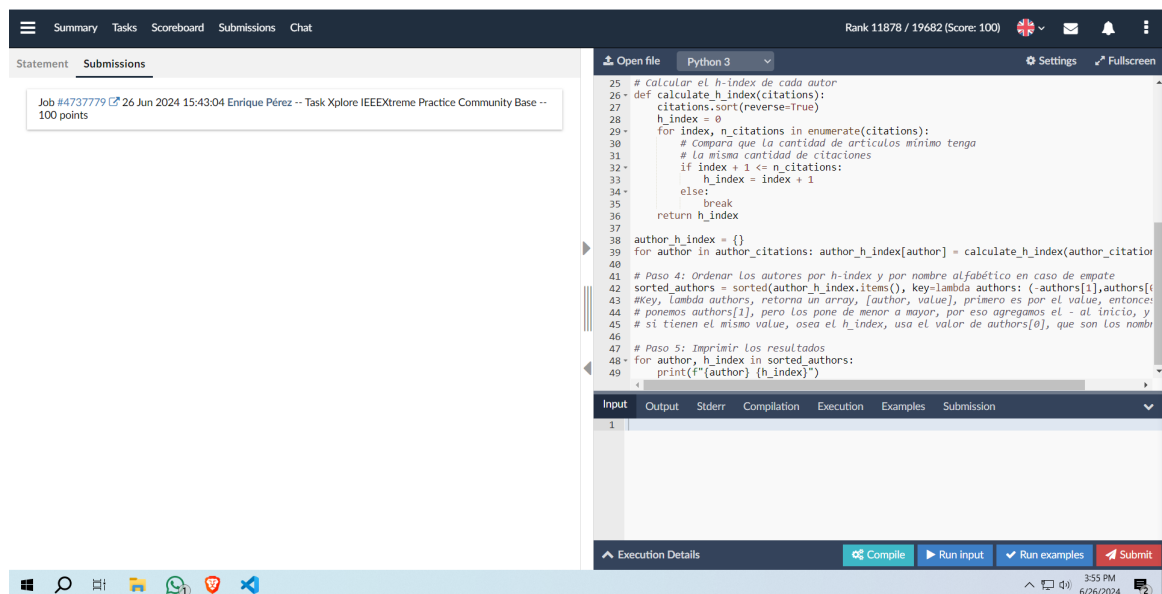


Figura 1: Captura Xplore, imagen\_muestra

### 3. Ejercicio de Make distinct

<https://csacademy.com/ieeextreme-practice/task/make-distinct/>

```
from collections import defaultdict

n = int(input())
numbers = list(map(int, input().split()))

list_numbers = sorted(numbers)
before_numbers = list(list_numbers)

def make_distinct():
    numero_alto = 0
    viewer = defaultdict()
    for i in range(len(list_numbers)-1):
        if abs(list_numbers[i] - list_numbers[i+1]) > 1:
            numero_alto = list_numbers[i] + 1
            break
    numero_bajo = list_numbers[0] - 1
    for index, number in enumerate(list_numbers):
        if (index+1 == len(list_numbers)): break
        if number != list_numbers[index + 1]:
            viewer[number] = 1
            continue
        else:
            if(numero_alto > abs(numero_bajo)):
                list_numbers[index] = numero_bajo
                if list_numbers[index] - list_numbers[index-1] > 1: numero_bajo = list_numbers[index-1]
                else: numero_bajo -= 1
            else:
                for i in range(index, len(list_numbers)-1):
                    if abs(list_numbers[i] - list_numbers[i+1]) > 1:
                        numero_alto = list_numbers[i] + 1
                else:
                    numero_alto = list_numbers[len(list_numbers)-1] + 1
                    list_numbers[index] = numero_alto
                    numero_alto += numero_alto
            viewer[number] = 1

make_distinct()
list_numbers = sorted(list_numbers)

def calculate_operations(arr_original, arr_distinct):
    operations = 0
    # Comparar y calcular las diferencias
```

```

for original, distinct in zip(arr_original, arr_distinct):
    operations += abs(distinct - original)

return operations

print(calculate_operations(before_numbers, list_numbers))

```

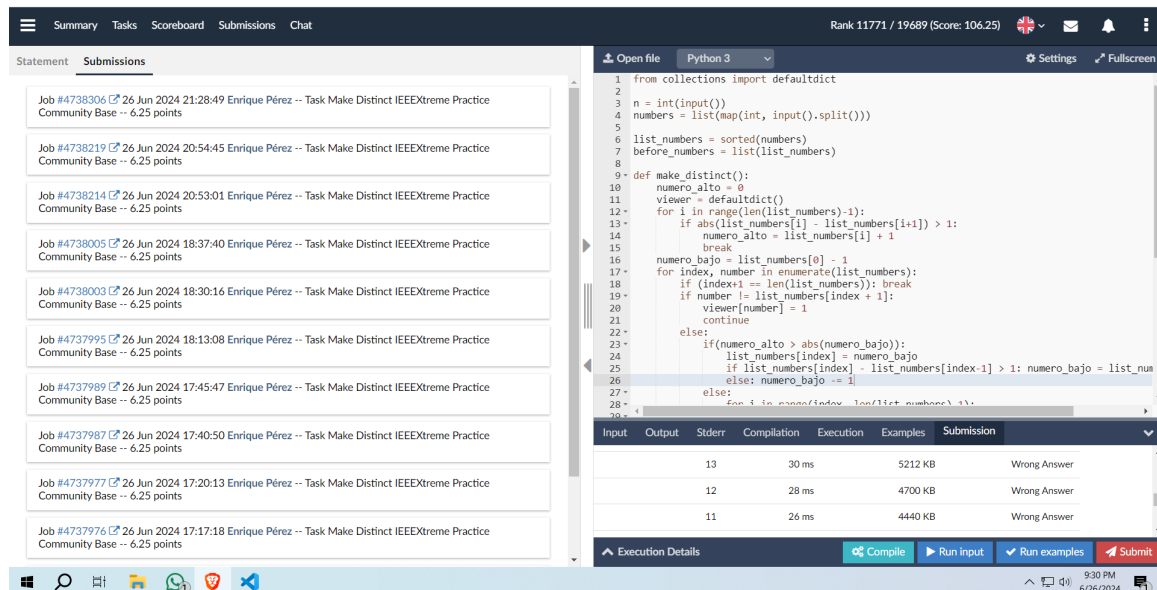


Figura 2: Captura make distinct, imagen\_muestra

### 3. Ejercicio de Troll Coder

<https://csacademy.com/ieeextreme-practice/task/troll-coder/>

```

import sys

n = int(input())

solucion_actual = [0] * n
print("Q", *solucion_actual)
sys.stdout.flush()
respuesta_actual = int(input())

for i in range(n):
    solucion_actual[i] = 1

```

```

print("Q", *solucion_actual)
sys.stdout.flush()
nueva_respuesta = int(input())

if nueva_respuesta <= respuesta_actual:
    solucion_actual[i] = 0
else:
    respuesta_actual = nueva_respuesta

print("A", *solucion_actual)
sys.stdout.flush()

```

The screenshot shows a web-based code editor interface. The top navigation bar includes 'Summary', 'Tasks', 'Scoreboard', 'Submissions', and 'Chat'. The main area is divided into three sections:

- Submissions Sidebar:** Lists three jobs:
  - Job #4738368: 26 Jun 2024 21:51:09 Enrique Pérez -- Task Troll Coder IEEEExtreme Practice Community Base -- Running
  - Job #4738367: 26 Jun 2024 21:50:08 Enrique Pérez -- Task Troll Coder IEEEExtreme Practice Community Base -- 100 points
  - Job #4738351: 26 Jun 2024 21:43:53 Enrique Pérez -- Task Troll Coder IEEEExtreme Practice Community Base -- 100 points
- Code Editor:** Contains Python code for a task, with line numbers 1 through 22. The code implements a logic for updating a solution based on input queries.
- Execution Details Panel:** Shows a table of test results:
 

| Test Number | CPU Usage | Memory Usage | Result                     |
|-------------|-----------|--------------|----------------------------|
| 45          | 24 ms     | 3940 KB      | Ok! Used 51 queries. (n+1) |
| 44          | 24 ms     | 3944 KB      | Ok! Used 46 queries. (n+1) |
| 43          | 26 ms     | 3936 KB      | Ok! Used 44 queries. (n+1) |
| 42          | 24 ms     | 3934 KB      | Ok! Used 47 queries. (n+1) |

The bottom status bar shows '9:51 PM 6/26/2024' and various system icons.

Figura 3: Captura Troller code, imagen\_muestra