Soluciones a los problemas: Clase 1 – Introducción y Fundamentos

Problemas: Clase 1 – Introducción y Fundamentos

Hola Algorithmic

```
# imprime el saludo en la consola
print("Hola, Algorithmic!")
```

¿Número Par o Impar?

```
# leemos un número y lo convertimos a entero
a = int(input())
# si el número es impar
if a % 2 == 1:
    print("Odd")  # mostramos "Odd"
else:
    print("Even")  # si no, es par y mostramos "Even"
```

Máximo de Dos Enteros

```
# leemos varios números separados por espacios, los convertimos a enteros y
mostramos el mayor
print(max(map(int, input().split())))
```

Contar Cuadrados Perfectos en una Lista

```
# leemos una lista de números enteros separados por espacios
arr = list(map(int, input().split()))
count = 0  # contador de números que son cuadrados perfectos
for num in arr:
    if num >= 0:
        # calculamos la raíz entera y comprobamos si es cuadrado perfecto
        raiz = int(num**(1/2))
        if raiz * raiz == num:  # Comprobamos si es un cuadrado perfecto
            count += 1
# mostramos la cantidad de cuadrados perfectos
print(count)
```

Otra solución utilizando compresión de listas

```
# leemos números separados por espacios y contamos cuántos son cuadrados perfectos
print(sum(
    1 for x in input().split()
    if int(int(x)**(1/2)) * int(int(x)**(1/2)) == int(x)  # comprobamos si x es
cuadrado perfecto
))
```

El mensaje secreto

```
def descifrar_mensaje(linea):
    resultado = []
    vocales = "aeiouAEIOU"
    for c in linea:
        if c in vocales:
            continue  # saltamos las vocales
        elif c.isalpha():
            resultado.append(c.upper())  # convertimos consonantes a mayúscula
        elif c == ' ':
            resultado.append('_')  # reemplazamos espacios por guiones bajos
        else:
            resultado.append(c)  # otros caracteres los dejamos igual
    return ''.join(resultado)
# leemos hasta que se ingrese '#'
while True:
    linea = input() # leemos la linea de entrada
    if linea == '#': # si es '#' terminamos el ciclo
        break
    print(descifrar_mensaje(linea))
```

Número Más Frecuente en una Lista

```
# leemos una lista de números separados por espacios
a = list(map(int, input().split()))
d = {}       # diccionario para contar cuántas veces aparece cada número
ma = -1      # mayor cantidad de repeticiones encontrada
sol = 0      # número con mayor frecuencia (y menor en caso de empate)
for x in a:
    if x in d:
        d[x] += 1
    else:
        d[x] = 1
    # actualizamos la solución si encontramos más repeticiones
    if ma < d[x]:
        ma = d[x]
        sol = x
```

```
    # si hay empate en repeticiones, elegimos el menor número
    elif ma == d[x] and x < sol:
        sol = x
print(sol)  # mostramos el número más frecuente
```

Suma de Todos los Elementos en una Matriz

```
# leemos la cantidad de filas y columnas
n = int(input())
m = int(input())
su = 0  # acumulador de la suma
for i in range(n):
    # leemos una fila de números y sumamos sus valores
    su += sum(map(int, input().split()))
print(su)  # mostramos la suma total
```

Batalla de Soldados de Juguete

Explicación de la solución

- Ordenamos tus soldados (`a`) y los del otro niño (`b`) de menor a mayor.
- Usamos un puntero `p` para recorrer los soldados del otro niño.
- Para cada uno de tus soldados (en orden creciente):
  - Si tu soldado es **más fuerte** que el soldado actual del otro niño (`x > b[p]`), ganamos la batalla.
  - Avanzamos `p` al siguiente soldado del otro niño para la próxima comparación.
- Esta estrategia asegura **maximizar el número de victorias** porque siempre usamos el soldado más débil posible que aún pueda ganar.
- Finalmente, `sol` contendrá el **número máximo de victorias** que puedes lograr.

```
# leemos la cantidad de soldados
n = int(input())
# leemos las fuerzas de tus soldados y del otro niño
a = list(map(int, input().split()))
b = list(map(int, input().split()))
# ordenamos ambos conjuntos de soldados de menor a mayor
a.sort()
b.sort()
p = 0       # puntero para recorrer los soldados del otro niño
sol = 0     # contador de victorias
# recorremos tus soldados
for x in a:
    if x > b[p]:  # si tu soldado es más fuerte que el actual del otro niño
        sol += 1  # ganamos la batalla
        p += 1    # pasamos al siguiente soldado del otro niño
print(sol)  # mostramos el número máximo de victorias
```

Columna Máxima por Fila

Idea de la solución

- Para **maximizar la suma**, en cada fila basta con elegir el **elemento más grande**.
- Recorremos cada fila, encontramos su máximo y lo sumamos al total.
- Al final, `sol` contendrá la **suma máxima posible** siguiendo la regla de elegir un elemento por fila.

```
# leemos la cantidad de filas (n) y columnas (m)
n, m = map(int, input().split())
# leemos la matriz fila por fila
a = []
for i in range(n):
    a.append(list(map(int, input().split())))
sol = 0  # Acumulador de la suma máxima
# recorremos cada fila
for i in range(n):
    ma = 0
    # buscamos el elemento más grande de la fila
    for j in range(m):
        ma = max(ma, a[i][j])
    sol += ma  # sumamos el máximo de la fila al total
print(sol)  # mostramos la suma máxima
```

Bingo Musical

Idea de la solución

- Usamos **conjuntos (`set`)** para manejar los números de cada jugador.
- `sol1` guarda los números que están en **todas las listas** mediante la **intersección** (`&=`).
- `sol2` guarda los números que están en **al menos una lista** mediante la **unión** (`|=`).
- Al final imprimimos ambos conjuntos convertidos a cadenas.

```
# leemos el número de jugadores
n = int(input())
# inicializamos los sets
sol1 = set([i for i in range(1, 1001)])  # para intersección: todos los números
posibles
sol2 = set()                             # para unión: empieza vacío
# procesamos cada jugador
for i in range(n):
    numeros = list(map(int, input().split()))[1:]  # ignoramos el primer número
(cantidad)
    set_numeros = set(numeros)                     # convertimos la lista en
conjunto
    sol1 &= set_numeros  # intersección: números que están en todas las listas
hasta ahora
    sol2 |= set_numeros  # unión: números que aparecen en al menos una lista
# mostramos el resultado
```

```
print(" ".join(map(str, sol1)))  # números que aparecen en todas las listas
print(" ".join(map(str, sol2)))  # números que aparecen en al menos una lista
```