Ejercicios de Colecciones no secuenciales

Programación — DAW

Ricardo Pérez López
IES Doñana

Curso 2025/2026

1. Dada una lista de correos electrónicos de usuarios, devolver:

   a. El número de correos distintos.

   b. La colección de correos sin duplicados.

```
emails = [
    "ana@mail.com", "juan@mail.com", "ana@mail.com",
    "luis@mail.com", "juan@mail.com"
]
```

2. Tres sistemas registran usuarios activos en un día. Obtener:

   a. Usuarios activos en todos los sistemas.

   b. Usuarios activos en al menos uno.

   c. Usuarios activos sólo en el sistema A.

```
a = {"ana", "juan", "luis"}
b = {"juan", "luis", "carmen"}
c = {"luis", "ana"}
```

3. Un sistema tiene usuarios permitidos y usuarios bloqueados. Calcular el conjunto de usuarios que pueden acceder realmente.

```
permitidos = {"ana", "juan", "luis", "carmen"}
bloqueados = {"juan", "carmen"}
```

4. Dadas dos frases, devolver el conjunto de palabras que aparecen en ambas.

```
t1 = "python es simple y potente"
t2 = "python es potente y elegante"
```

5. Dos alumnos entregan textos. Se considera sospechoso de plagio si comparten más del 70 % del total de palabras distintas usadas entre los dos textos. Comprobar si ha habido plagio. Ignorar mayúsculas y minúsculas.

```
a = "programar en python es divertido y educativo"
b = "programar en python es educativo y muy divertido"
```

6. Un profesor y un alumno indican los días que tienen disponibles cada uno:

```
profesor = {"lunes", "martes", "jueves"}
alumno   = {"martes", "miércoles", "jueves"}
```

   - Calcular:

     a. Días posibles para tutoría.

     b. Días en los que ninguno puede.

7. Comprobar si una contraseña contiene:

   - al menos una vocal, y
   - al menos un dígito.

```
password = "abc9x"
```

8. Una fila de Sudoku es válida si contiene exactamente los números del 1 al 9 sin repetición, da igual el orden. Comprobar si una lista de números enteros es una fila válida en el Sudoku. Por ejemplo, la siguiente es una fila válida:

```
fila = [9, 2, 7, 1, 5, 4, 3, 6, 8]
```

9. Un usuario tiene un conjunto de permisos, y una acción requiere ciertos permisos mínimos. Determinar si el usuario puede realizar la acción.

```
usuario = {"leer", "escribir", "exportar"}
requeridos = {"leer", "escribir"}
```

10. Dado un texto y un conjunto de palabras prohibidas, indica si aparece alguna palabra prohibida en el texto.

```
texto = "este mensaje es confidencial"
prohibidas = {"secreto", "confidencial", "privado"}
```

11. Dado un conjunto de alumnos matriculados y un conjunto de alumnos aprobados, obtener el conjunto de alumnos suspensos.

```
matriculados = {"ana", "juan", "luis", "carmen"}
aprobados = {"ana", "luis"}
```

12. Recorrer una lista y mostrar sólo los elementos que aparecen por primera vez. Por ejemplo, para esta entrada:

    ```
    datos = [3, 1, 3, 2, 1, 4, 2, 5]
    ```

    se espera obtener la siguiente salida:

    ```
    [3, 1, 2, 4, 5]
    ```

13. Se dispone de dos listas. Una muestra los nombres de las personas (`names`), mientras que el otro muestra sus profesiones (`jobs`). Crear un diccionario que asocie a cada persona con su respectiva profesión.

Persona	Profesión
Annie	Profesor
Steven	Ingeniero
Lisa	Doctor
Osman	Cajero

    - Ejemplos:

      ```
      names = ["Dennis", "Vera", "Mabel", "Annette", "Sussan"]
      jobs = ["Butcher", "Programmer", "Doctor", "Teacher", "Lecturer"]

      assign_person_to_job(names, jobs) = {
        "Dennis": "Butcher",
        "Vera": "Programmer",
        "Mabel": "Doctor",
        "Annette": "Teacher",
        "Sussan": "Lecturer"
      }
      ```

    - Notas:

      - Las dos listas tienen la misma longitud.
      - El índice de un nombre en la lista `names` es el mismo que el índice de la profesión de la persona en la lista `jobs`, como se muestra en la tabla anterior.

    - Fuente: `https://edabit.com/challenge/XPnTdP9eDkwqAQeWq`

14. ¡Acabas de regresar a casa y descubres que han robado tu mansión! Dado un diccionario de los artículos robados, devolver el monto total del robo (número). Si no se robó nada, devolver la cadena `"Lucky you!"`.

    - Ejemplos:

      ```
      calculate_losses({
        "tv" : 30,
        "skate" : 20,
        "stereo" : 50,
      }) == 100
      calculate_losses({
        "painting" : 20000,
      }) == 20000
      calculate_losses({}) == "Lucky you!"
      ```

    - Nota: El valor del artículo siempre es positivo.

    - Fuente: https://edabit.com/challenge/kSmkcYLcWRnEEwXwX

15. Dadas dos cadenas, crear una función que devuelva el número total de caracteres únicos de la cadena combinada.

    - Ejemplos:

      ```
      count_unique("apple", "play") == 5
      # "appleplay" tiene 5 caracteres únicos:
      # "a", "e", "l", "p", "y"
      count_unique("sore", "zebra") == 7
      count_unique("a", "soup") == 5
      ```

    - Notas:

      - Cada palabra contendrá al menos una letra.
      - Todas las palabras estarán en minúsculas.

    - Fuente: https://edabit.com/challenge/oTJaJ895ubqqpRPMh

16. Python tiene tres tipos principales de estructuras de datos formadas por objetos más pequeños:

    - Listas, escritas con corchetes `[]`, como `[1, 2, 4, 8]`.
    - Tuplas, escritas entre paréntesis `()`, como `(7, 8, 9)`.
    - Conjuntos, escritos con llaves `{}`, como `{2, 3, 5, 7, 11, 13}`.

Cada uno de estos tipos tiene sus propias propiedades y peculiaridades especiales que vale la pena conocer, pero este ejercicio solo consiste en transformar estos tipos de datos entre sí.

Esto se puede hacer de la siguiente manera:

- `tuple([1, 2, 4, 8])` devuelve `(1, 2, 4, 8)`
- `list({2, 3, 5, 7, 11})` devuelve `[2, 3, 5, 7, 11, 13]`
- `set((1, 2, 4))` devuelve `{1, 2, 4}`

Dadas dos estructuras de datos, `data1` y `data2`, devolver `data2` convertido al tipo de `data1`.

- Ejemplos:

```
convert([1, 2, 4, 8], (7, 8, 9)) == [7, 8, 9]
convert((7, 8, 9), [1, 2, 4, 8]) == (1, 2, 4, 8)
convert([1, 2, 4, 8], {2, 3, 5, 7, 11, 13}) == [2, 3, 5, 7, 11, 13]
convert({2, 3, 5, 7, 11, 13}, [1, 2, 4, 8]) == {8, 1, 2, 4}
```

- Notas:
  - Es posible que haya notado que el último ejemplo da `{8, 1, 2, 4}` en lugar de `{1, 2, 4, 8}`. Esto tiene que ver con el hecho de que en los conjuntos el orden no importa, por lo que Python considera que `{8, 1, 2, 4}` y `{1, 2, 4, 8}` es el mismo conjunto.
  - En los casos de prueba que no tendrá que preocuparse acerca de las órdenes: las respuestas siempre tendrán la orden dada por la aplicación de las funciones `list()`, `tuple()`, `set()`.
- Fuente: `https://edabit.com/challenge/GPRZauknkEbyRz5qM`

17. Dados tres argumentos (un diccionario `obj`, una clave `name` y un valor `value`) devolver un diccionario con ese nombre y valor (como pares clave-valor).

    - Ejemplos:

```
add_name({}, "Brutus", 300) == {"Brutus": 300}
add_name({"piano": 500}, "Brutus", 400) == {"piano": 500, "Brutus": 400}
add_name({"piano": 500, "stereo": 300}, "Caligula", 440)
    == {"piano": 500, "stereo": 300, "Caligula": 440}
```

    - Nota: El argumento `value` será un número.
    - Fuente: `https://edabit.com/challenge/owmKG47zYRJw72aiD`

18. Dada una lista de números, escribir una función que devuelva una lista que…

1. Elimine todos los elementos duplicados.
2. Esté ordenado de menor a mayor valor.

- Ejemplos:

```
unique_sort([1, 2, 4, 3]) == [1, 2, 3, 4]
unique_sort([1, 4, 4, 4, 4, 4, 3, 2, 1, 2]) == [1, 2, 3, 4]
unique_sort([6, 7, 3, 2, 1]) == [1, 2, 3, 6, 7]
```

- Fuente: https://edabit.com/challenge/EZMCpHaNFg2Yfsnxx

19. Crear una función que devuelva una lista de valores *booleanos*, a partir de un número dado. Repitiendo el dígito del número uno a la vez, agregar `True` si el dígito es `1` y `False` si es `0`.

- Ejemplos:

```
integer_boolean("100101") == [True, False, False, True, False, True]
integer_boolean("10") == [True, False]
integer_boolean("001") == [False, False, True]
```

- Nota: Los números sólo pueden ser `0` y `1`.
- Fuente: https://edabit.com/challenge/58DYAThA2dxnAsMpL

20. Crear una función que tome un diccionario de nombres de estudiantes y devuelva una lista de nombres de estudiantes en orden alfabético.

- Ejemplos:

```
get_student_names({
  "Student 1" : "Steve",
  "Student 2" : "Becky",
  "Student 3" : "John"
}) == ["Becky", "John", "Steve"]
```

- Fuente: https://edabit.com/challenge/HvkPdhijquecKASdF