Guía de Ejercicios 8: Tuplas, conjuntos y diccionarios

Objetivos:

- Ejercitar el uso de los tipos de datos tupla, conjunto y diccionario y sus operaciones básicas.
- Repasar el concepto de mutabilidad, y entender cómo aplica a estos tipos de datos.

Tuplas

Ejercicio 1. ¿Qué valores se imprimen al ejecutar el siguiente programa?

```
t:tuple[int, str]
u:tuple[int, str]
t = (10, 'diez')
u = t
print(u[0], u[1])
t = (20, 'veinte')
print(t, u)
t[0] = 30       # ¿Es válido hacer esta asignación?
```

Ejercicio 2. Especificar, programar y verificar con testing funciones para resolver los siguientes problemas, usando tuplas:

(a) Dado un punto en el plano (una tupla de dos floats), devolver su distancia al origen (`(0.0, 0.0)`). Ejemplo: para el punto `(4.0, 3.0)`, debe devolver `5.0`.

(b) Dados dos puntos en el plano, devolver su suma, componente a componente. Ejemplo: para los puntos `(-1.0, 2.0)` y `(1.5, 3.3)`, debe devolver `(0.5, 5.3)`.

(c) Dado un punto en el plano y un float `f`, devolver el punto resultante de sumar `f` a cada componente del punto. Ejemplo: para `(-1.0, 2.0)` y `3.3`, debe devolver `(2.3, 5.3)`.

(d) Dado un punto en el plano, devolver el punto entero más cercano. Ejemplo: para el punto `(-1.9, 2.2)`, debe devolver `(-2, 2)`. Sugerencia: usar `round`.

Ejercicio 3. Implementar la siguiente función:

```
def buscar(elem:int, lista:list[int]) -> tuple[bool, int]:
    ''' Requiere: nada.
        Devuelve: (True, p) si elem aparece en la lista por primera vez
        en la posición p; o bien (False, None) si no aparece nunca.
    '''
```

Ejemplos: Siendo `xs` una lista de enteros con valor `[3, 4, 5, 5]`, `buscar(1, xs)` devuelve la tupla `(False, None)`, y `buscar(5, xs)` devuelve la tupla `(True, 2)`. (`None` es un valor especial de Python, que significa literalmente *ningún valor*.[1])

[1] A quienes quieran profundizar en el tema de `None`, recomendamos leer la página `https://recursospython.com/guias-y-manuales/el-tipo-de-dato-none/`.

Conjuntos

Ejercicio 4. Evaluación de expresiones.

(a) Para cada una de las siguientes expresiones, determinar cuál es su tipo y evaluarla a mano, realizando las conversiones de tipos que sean necesarias. Suponer que `xs` es una lista de enteros con valor `[1, 4, 2, 1, 5]`.

(I) `{}`
(II) `{'a', 'b', 'a'}`
(III) `set()`
(IV) `set(xs)`
(V) `list(set(xs))`
(VI) `len(xs) - len(set(xs))`
(VII) `5 in set(xs)`
(VIII) `set(xs) & {1, 10}`
(IX) `set(xs) | {1, 10}`
(X) `set(xs) - {1, 10}`

(b) Revisar las respuestas del punto anterior, esta vez usando una consola `ipython`.

Ejercicio 5. Dada una lista de strings (posiblemente con repetidos), queremos averiguar cuántos strings únicos hay. Escribir una función en Python que resuelva esto de manera sencilla usando conjuntos. Por ejemplo, en `['abc', 'd', 'ef', 'abc', 'ef', 'ef']` hay 3 strings únicos.

Ejercicio 6. Un **pangrama** es un texto que usa todas las letras posibles del alfabeto de un idioma. Se pide escribir una función en Python que determine si un texto en español es un pangrama. Por simplicidad, suponer que el texto de entrada está compuesto solamente por letras en minúscula, sin tildes ni diéresis, pero sí puede tener eñes. Por ejemplo, `es_pangrama('extraño pan de col y kiwi se quemo bajo fugaz vaho')` debe devolver `True`.[2]

Ejercicio 7. Determinar a mano qué se imprime por pantalla en cada `print` al ejecutar el siguiente programa. Revisar luego los resultados en Python.

```
def agregar1(x:str, c:set[str]):
    c.add(x)

def agregar2(x:str, c:set[str]) -> set[str]:
    r:set[str] = c | {x}
    return r

conjunto:set[str] = set()
agregar1('a', conjunto)
print(conjunto)

otro:set[str] = conjunto
agregar1('b', otro)
print(conjunto, otro)

otro = agregar2('c', otro)
print(conjunto, otro)

agregar1('d', conjunto)
print(conjunto, otro)
```

[2] Fuente: `https://es.wikipedia.org/wiki/Pangrama`

Ejercicio 8.

(a) El **índice Jaccard** es una medida de similitud entre conjuntos. Se define como

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

donde $|C|$ se refiere a la cantidad de elementos del conjunto C.

Jaccard(A, B) se mueve entre 0 (cuando A y B son totalmente distintos) y 1 (cuando son iguales). Escribir una función en Python que compute el índice Jaccard. Por ejemplo, `jaccard({1,2,3}, {3,4})` debe devolver 0.25.

(b) Se denomina **k-shingles** a las subsecuencias de caracteres de longitud `k` en un string. Escribir una función en Python que, dados un string `s` y un entero `k`, devuelva el conjunto de `k`-shingles de `s`. Por ejemplo, `k_shingles('hola mundo', 5)` debe devolver el conjunto `{'hola ', 'ola m', 'la mu', 'a mun', ' mund', 'mundo'}`.

(c) Escribir una función en Python que, dados dos strings, extraiga sus `k`-shingles y compute la similitud Jaccard entre ambos conjuntos. Esta técnica se usa frecuentemente en la detección automática de plagio en textos.[3] Por simplicidad, suponer que los strings están compuestos solamente por letras en minúscula, sin tildes ni diéresis, pero sí pueden tener eñes.

Diccionarios

Ejercicio 9. Escribir una función que, dado un string `s`, devuelva un diccionario con la cantidad de apariciones de cada carácter en `s`. Por ejemplo, para `'agua'` tiene que devolver el diccionario `{'a': 2, 'g': 1, 'u': 1}`.

Ejercicio 10. Escribir una función que deletree palabras. Por ejemplo, dado `'aljibe'` debe devolver `'a, ele, jota, i, be, e.'` Usar un diccionario para representar las reglas.

Ejercicio 11. Determinar qué hace la siguiente función y ejecutarla a mano para algunas entradas. Revisar luego los resultados en Python.

```
def construir_tabla(n:int) -> dict[int, int]:
    ''' ...completar... '''
    r:dict[int, int] = dict()
    i:int = 1
    while i <= n:
        r[i] = i * i
        i = i + 1
    return r
```

Ejercicio 12. Escribir una función que encuentre el máximo valor entero de un diccionario de strings a enteros (es decir, `dict[str,int]`) y devuelva el conjunto de claves que tienen dicho valor. Por ejemplo, para el diccionario `{'a':10, 'b':14, 'c':9, 'd':14, 'e':7}` debe devolver `{'b', 'd'}` (porque tanto `'b'` como `'d'` tienen valor 14, que es el valor máximo del diccionario).

[3] Lectura opcional para quienes les interese el tema: *Mining of Massive Datasets* (`http://www.mmds.org`), sección 3.1.

Ejercicio 13.

(a) Determinar a mano qué se imprime por pantalla en cada `print` al ejecutar el siguiente programa. Suponer definida la función `construir_tabla` del Ejercicio 11. Revisar luego los resultados en Python.

```
def filtrar(tabla:dict[int, int], digito:int):
    ''' ...completar... '''
    claves_a_eliminar:list[int] = []
    for k in tabla:
        valor:int = tabla[k]
        ultimo_digito:int = int(str(valor)[-1])
        if ultimo_digito != digito:
            claves_a_eliminar.append(k)
    for k in claves_a_eliminar:
        tabla.pop(k)

tabla:dict[int, int] = construir_tabla(10)
otra:dict[int, int] = tabla
print(otra)

filtrar(tabla, 9)
print(tabla)
print(otra)
```

(b) Completar el *docstring* de la función `filtrar` estableciendo las cláusulas Requiere y Devuelve y, si corresponde, también la cláusula Modifica. Es decir, completarlo describiendo las restricciones sobre los parámetros de la función, el valor de retorno de la función y, si es que los tiene, los efectos colaterales de la función.

(c) Reescribir la función `filtrar`, de modo que devuelva un nuevo diccionario y no modifique el diccionario pasado como argumento.