PRÁCTICO 5.2 – Tuplas, Sets y Diccionarios

Metodología: Se recomienda que los primeros ejercicios se realicen en pseudocódigo para luego implementarse en Python. Para cada ejercicio, defina un archivo y pruebe si el método definido funciona.

Estructuras de datos de ejemplo:

```
tuple1 = (“Orange”, [1, 2, 3], 2022, “Septiembre”)
tuple2 = ((), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d'))
tuple3 = (('R', 'W', 'B'), ('G', 'P', 'L'), ('O', 'Y', 'L'))
set1a = {‘A’, ‘B’, ‘C’}, set1b = {‘A’}, set1c = {‘D’}
set2a = {10, 50, 20, 5}, set2b = {25, 33, 11, 3}
dict1a = {‘A’: 1, ‘B’: 2}, dict1b = {‘C’: 3, ‘D’: 4},
dict2 = {‘D’: 1, ‘B’: 5, ‘F’: 3, ‘A’: 7, ‘C’: 2}
dict3 = {‘s1’: {‘n1’:‘A’, ‘n2’:‘B’}, ‘s2’: {‘n1’:‘A’, ‘n2’:‘B’},
‘s3’: {‘n1’:‘C’, ‘n2’:‘D’}, ‘s4’: {‘n1’:‘E’, ‘n2’:‘F’}}
```

TUPLAS:

1. Escriba un programa de Python para eliminar un elemento de una tupla.

   ```
   INPUT: pop_tuple(tuple1, 2022)
   OUTPUT: (“Orange”, [1, 2, 3], “Septiembre”).
   Debe retornar tuple1 si no existe el elemento indicado.
   ```

2. Escriba un programa de Python para separar una tupla dada una posición.

   ```
   INPUT: split_tuple(tuple1, 2)
   OUTPUT: [(“Orange”, [1, 2, 3]), (2002, “Septiembre”)].
   Debe retornar tuple1 si no existe la posición indicada.
   ```

3. Escriba un programa de Python para encontrar el índice de un elemento de una tupla (similar a la función index).

   ```
   INPUT: index_tuple(tuple1, 2022)
   OUTPUT: 2
   Debe retornar -1 si no existe el elemento indicado.
   ```

4. Escriba un programa en Python para invertir una tupla (similar a función reverse).

   ```
   INPUT: reverse_tuple(tuple1)
   OUTPUT: (“Septiembre”, 2022, [1, 2, 3], “Orange”)
   Debe retornar tuple1 si no es posible realizar la inversión.
   ```

5. Escriba un programa en Python para eliminar tuplas vacías de una lista de tuplas.

   ```
   INPUT: remove_spaces_from_tuple(tuple2)
   OUTPUT: (('',), ('a', 'b'), ('a', 'b', 'c'), 'd')
   Debe retornar tuple2 si no existe ninguna tupla vacía.
   ```

6. Escriba un programa de Python que convierta un String dado en una tupla y elimine los espacios en blanco.

   ```
   INPUT: str_to_tuple(str1) con str1 = “Universidad Montevideo”
   OUTPUT: (‘U’, ‘n’, ‘i’, ‘v’, ‘e’, ‘r’, ‘s’, ‘i’, ‘d’, ‘a’, ‘d’,
   ‘M’, ‘o’, ‘n’, ‘t’, ‘e’, ‘v’, ‘i’, ‘d’, ‘e’, ‘o’)
   Debe retornar tupla vacía si el String indicado es vacío.
   ```

7. Escriba un programa de Python para verificar si un elemento específico se presenta en una tupla de tuplas.

   ```
   INPUT: is_element_in_tuple(tuple3, ‘G’)
   OUTPUT: True
   INPUT: is_element_in_tuple(tuple3, ‘F’)
   OUTPUT: False
   ```

SETS:

8. Escriba un programa en Python para verificar si un conjunto es un subconjunto de otro conjunto.

   ```
   INPUT: is_subset_conjuntos(set1a, set1b)
   OUTPUT: True
   INPUT: is_subset_conjuntos(set1a, set1c)
   OUTPUT: False
   ```

9. Escriba un programa de Python para eliminar todos los elementos de un conjunto dado.

   ```
   INPUT: clear_conjunto(set1a)
   OUTPUT: set()
   También debe retornar set() si el conjunto de entrada es vacío.
   ```

10. Escriba un programa en Python para encontrar el valor máximo y mínimo en un conjunto.

    ```
    INPUT: max_conjunto(set2a)
    OUTPUT: 50

    INPUT: min_conjunto(set2a)
    OUTPUT: 5
    ```

11. Escriba un programa en Python para verificar si dos conjuntos dados no tienen elementos en común.

    ```
    INPUT: is_disjoint_conjuntos(set1a, set1b)
    OUTPUT: False

    INPUT: is_disjoint_conjuntos(set1a, set1c)
    OUTPUT: True
    ```

12. E Escriba un programa en Python para encontrar los elementos en un conjunto dado que no están en otro conjunto.

```
INPUT: difference_conjuntos(set1a, set1b)
OUTPUT: {‘B’, ‘C’, ‘D’}
Debe retornar set() si todos los elementos se encuentran en ambos conjuntos.
```

13. Escriba un programa de Python para eliminar los elementos pares de un conjunto y agregarlos a un segundo conjunto.

```
INPUT: trasladar_pares(set2a, {33, 17, 24})
OUTPUT: {10, 17, 24, 20, 50, 33}
Debe retornar set() si no existe ningún elemento par en conjunto.
```

14. Escriba un programa de Python para verificar que todos los elementos de una lista se encuentran en un conjunto.

```
INPUT: todos_en_conjunto(set2a, [10, 50])
OUTPUT: True
INPUT: todos_en_conjunto(set2a, [13, 50])
OUTPUT: False
Debe retornar False si el conjunto o la lista son vacíos.
```

DICCIONARIOS:

15. Escriba un programa Python para concatenar diccionarios para crear uno nuevo.

```
INPUT: concatenar_diccionarios(dict1a, dict1b)
OUTPUT: {‘A’: 1, ‘B’: 2, ‘C’: 3, ‘D’: 4}
Debe retornar {} si ambos diccionarios son vacíos, o el diccionario con contenido si el otro diccionario es vacío.
```

16. Escriba un script de Python para verificar si una llave (key) determinada ya existe en un diccionario.

```
INPUT: existe_key(dict1a, ‘A’)
OUTPUT: True
INPUT: existe_key(dict1a, ‘D’)
OUTPUT: False
Debe retornar False si el diccionario es vacío.
```

17. Escriba un script de Python para verificar si un valor (value) determinado ya existe en un diccionario.

```
INPUT: existe_value(dict1a, 1)
OUTPUT: True
INPUT: existe_value(dict1a, 4)
OUTPUT: False
Debe retornar False si el diccionario es vacío.
```

18. Escriba un script de Python para imprimir un diccionario donde las llaves son números entre a y b (incluidos) y los valores son cuadrados de las llaves.

    ```
    INPUT: generar_cuadrados_diccionario(5, 10)
    OUTPUT: {5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
    Debe retornar {} si a > b o si a < 0 o b > 20.
    ```

19. Escriba un programa de Python para mapear dos listas en un diccionario.

    ```
    INPUT: lists_to_dict([‘A’, ‘B’, ‘C’], [1, 2, 3])
    OUTPUT: {‘A’: 1, ‘B’: 2, ‘C’: 3}
    Debe retornar {} si el largo de las listas es diferente o si son vacías (no tienen ningún elemento).
    ```

20. Escriba un programa de Python para ordenar un diccionario dado por llave.

    ```
    INPUT: ordenar_por_llave(dict2)
    OUTPUT: {‘A’: 7, ‘B’: 5, ‘C’: 2, ‘D’: 1, ‘F’: 3}
    Debe retornar {} si el diccionario de entrada es vacío.
    ```

21. Escriba un programa de Python para ordenar un diccionario dado por valor.

    ```
    INPUT: ordenar_por_valor(dict2)
    OUTPUT: {‘D’: 1, ‘C’: 2, ‘F’: 3, ‘B’: 5, ‘A’: 7}
    Debe retornar {} si el diccionario de entrada es vacío.
    ```

22. Escriba un programa de Python para eliminar duplicados de un diccionario.

    ```
    INPUT: eliminar_duplicados_dict(dict3)
    OUTPUT: {‘s1’: {‘n1’:‘A’, ‘n2’:‘B’}, ‘s3’: {‘n1’:‘C’, ‘n2’:‘D’}, ‘s4’: {‘n1’:‘E’, ‘n2’:‘F’}}
    Debe retornar el diccionario original en caso de no tener elementos duplicados, o {} si el diccionario es vacío.
    ```

23. Escriba un programa en Python para convertir una lista en un diccionario anidado de llaves (keys).

    ```
    INPUT: list_to_nested_dict([1, 2, 3, 4]})
    OUTPUT: {1: {2: {3: {4: {}}}}}
    Debe retornar {} si la lista de entrada es vacía.
    ```

24. Escriba un programa Python para dividir un diccionario dado de listas en una lista de diccionarios.

    ```
    INPUT: 'Math': [88, 89, 62, 95], 'Physics': [77, 78, 84, 80]}
    OUTPUT: [{'Math': 88, 'Physics': 77}, {'Math': 89, 'Physics': 78}, {'Math': 62, 'Physics': 84}, {'Math': 95, 'Physics': 80}]
    Debe retornar {} si el diccionario de entrada es vacío.
    ```