Fundamentos de Programación

Cuaderno de Trabajo 5

Ejercicios resueltos

1. Programa una función que evalúe si dos fechas, definidas en términos de día, mes y año, son iguales. Asume que las fechas se implementan como diccionarios con entradas dia, mes y año y que corresponden a fechas válidas.

SOLUCION PROPUESTA:

```
def fecha(dia, mes, anno):
    """ int, int, int -> dict
        OBJ: Crea un nuevo diccionario de tipo fecha.
        PRE: dia/mes/anno es una fecha válida.
    """
    return {'dia': dia, 'mes': mes, 'anno': anno}

def iguales(fecha1,fecha2):
    """ dict, dict -> bool
        OBJ: Calcula si una fecha es igual a otra.
        PRE: Los diccionarios fecha1 y fecha2 son de tipo fecha.
    """
    return fecha1 == fecha2

# probadores
f1 = fecha(1,12,2020)
f2 = fecha(1,12,2019)
print(iguales(f1,f1))
print(iguales(f1,f2))
```

2. Implementa ahora dos funciones, una que indiquen si una fecha es posterior a otra y una segunda que calcule si una fecha es anterior a otra. Todas las fechas se consideran válidas. Trata de reutilizar la función del problema 1.

SOLUCION PROPUESTA:

```
def anterior(fecha1,fecha2):
    """ dict, dict -> bool
        OBJ: Calcula si una fecha es anterior a otra.
        PRE: Los diccionarios fecha1 y fecha2 son de tipo fecha.
    """
    return  fecha1['anno'] < fecha2['anno']  or \
            fecha1['anno'] == fecha2['anno'] and\
            fecha1['mes'] < fecha2['mes'] or \
            fecha1['anno'] == fecha2['anno'] and \
            fecha1['mes'] == fecha2['mes'] and \
            fecha1['dia'] < fecha2['dia']

def posterior(fecha1,fecha2):
    """ dict, dict -> bool
        OBJ: Calcula si una fecha (fecha1) es posterior a otra (fecha2).
        PRE: Los diccionarios fecha1 y fecha2 son de tipo fecha.
    """
    return not iguales(fecha1,fecha2) and not anterior(fecha1,fecha2)
```

```
# probadores
f1 = fecha(1,12,2020)
f2 = fecha(1,12,2019)
print(anterior(f1,f2))
print(anterior(f2,f1))
print(posterior(f2,f1))
```

3. Hacer un programa que cree un diccionario con personas (nombre-edad) y permita que el usuario pregunte si una persona, elegida por él, está o no en el mismo. Al final se mostrará un informe con el contenido del diccionario.

SOLUCION PROPUESTA:

```
edades = {}
edades['Susana'] = 23
edades['Pedro'] = 19
edades['Andres'] = 78
edades['Angela'] = 45

nombre_buscado = input('Dime un nombre para buscarlo: ')
if nombre_buscado in edades:
    print (nombre_buscado, ' sí está en el diccionario. Su edad es: ',\
    edades[nombre_buscado], ' años')
else:
    print (nombre_buscado, " no está en el diccionario")
    print ("\nLas siguientes personas están en el diccionario:")
    for persona in edades.keys(): print(persona, end='\t')
    print ('\nSus edades son...')
    for edad in edades.values(): print(edad, end='\t')
    print ('\nEl diccionario tiene ', len(edades), ' entradas')
```

Ejercicios propuestos

1. Piensa en las estructuras de datos que permitan almacenar información sobre personas con objeto de hacer un estudio estadístico. Así, se deberá almacenar el nombre, sexo y edad de cada persona. Programa posteriormente una función para leer por teclado datos relativos a una persona y otra que muestra dichos datos por pantalla.
2. Programa un software que utilice lo programado en el ejercicio 1 para leer los datos de 10 personas y calcule la media de edad general, la media por sexo, la cantidad de mujeres que tienen entre 13 y 16 años y el número de hombres menores de 20 años.
3. Amplía el ejercicio anterior mostrando al final del proceso los datos completos de la mujer y el hombre más jóvenes de todos los introducidos.
4. Implementa una estructura que de soporte a los puntos en 2D, es decir, con dos coordenadas x e y. Programa después funciones para su suma y resta.
5. Programa una función distancia_2D que calcule la distancia entre dos puntos. La función retornará un número real según la siguiente fórmula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

6. Implementa una aplicación que solicite 10 valores y los almacene en un contenedor. A continuación, tu aplicación recorrerá el contenedor restándole a cada valor el valor que

se encuentre en la siguiente posición (ej.: [1, 3, 5], resultado [(1-3), (3-5), (5-1)] = -2, -2, 4). Modulariza tu solución.

7. Implementa una aplicación para ayudar en la gestión de cobros de una gasolinera. Mediante 3 listas deberá calcular el gasto de un total de 10 clientes. La primera lista será utilizada para almacenar el gasto de cada cliente en gasolina, la segunda para almacenar el gasto en productos de la tienda de la gasolinera, y la tercera lista almacenará la suma de las dos anteriores. La aplicación solicitará por teclado los gastos de gasolina y de la tienda para cada uno de los 10 clientes.
8. Realiza una aplicación que permita sumar dos matrices de tamaño 3x4. Las matrices contienen datos sobre Puntos2D. Debe hacer las siguientes tareas:

    a. Solicitar datos para la matriz A.
    b. Solicitar datos para la matriz B.
    c. Presentar el resultado de matriz A + matriz B.

9. El código QR (abreviatura de *Quick Response Code*) fue creado en 1994 por una filial japonesa de Toyota que fabrica componentes de automóviles. Estos códigos permiten almacenar información sobre un producto codificándola en un cuadrado de NxN con píxeles que pueden ser blancos o negros. En su formato más pequeño, los códigos tienen 21x21 píxeles (versión 1), y en la más grande 177x177 (versión 40). Programe una aplicación que genere aleatoriamente códigos QR versión 1 y los muestre por pantalla utilizando, por ejemplo, asteriscos. Nota: utilice la biblioteca `random` y la función `randint` para generar los números aleatorios.

ACABAR

10. Realiza una aplicación que permita comprobar si un tablero de Conecta4 (https://es.wikipedia.org/wiki/Conecta_4) ha completado las 4 en raya. La aplicación deberá:

    a. Solicitar datos para un tablero de 7x6
    b. Solicitar introducir valores para representar las fichas del tablero en un momento concreto del juego
    c. Mostrar por pantalla el tablero con los valores introducidos
    d. Indicar si alguna fila, columna o diagonal incluye 4 en raya (4 casillas consecutivas con una ficha de igual color).

11. Implementa un procedimiento que lea palabras de la entrada estándar hasta que se introduzca “fin” y retorne un diccionario de frecuencias de las longitudes de las palabras (palabras de longitud 4: 5, palabras de longitud 6: 3, etc.). Ten en cuenta que no hay límite a la longitud (en caracteres) de las palabras.

12. Programa una función que reciba un diccionario y una lista y que como salida genere dos listas: una con todos los valores de aquellos elementos de la lista que están en el diccionario, y otra con los que NO están en el diccionario.
13. Si tenemos un diccionario que contiene como claves el nombre de una persona y como valor una lista con sus “preferencias personales”, es posible programar una función `agregaPreferencia(diccionario, persona, preferencia)` tal que:

a. Si la persona no existe, la agrega al diccionario con una lista que contiene un solo elemento que es la nueva preferencia.
b. Si la persona existe y la preferencia actual existe en su lista, no tiene ningún efecto, pero si dicha preferencia no existe en su lista, la agrega a la lista.

14. Supón que tenemos 2 diccionarios, uno con la plantilla del FC Barcelona (22 jugadores, parejas "dorsal" y "nombre que aparece en la camiseta") y otro con las alineaciones de las distintas jornadas de la Liga 2020-2021 (número de jornada y lista de jugadores que jugaron esa jornada). Programa funciones para:
    a. Introducir los datos de una jornada a partir de su número y una lista de jugadores y dorsales. Obviamente no puede repetirse ningún jugador en la alineación ni participar jugadores que no pertenecen a la plantilla del Barcelona FC.
    b. Determinar si un jugador participó o no en una cierta jornada a partir de su dorsal o de su nombre (como se prefiera).
    c. Listar todas las jornadas en que jugó un determinado jugador cuyo dorsal se indica.
    d. Mostrar el nombre del jugador que más partidos jugó en una temporada, indicando el número de partidos que jugó.
15. En Python es muy sencillo tratar archivos de texto. Investiga cómo abrir y leer línea a línea un archivo de texto en Python y adapta el ejercicio 12 para que la entrada de las palabras sea ahora el texto leído de un archivo "entrada.txt".