Fundamentos de Programación

Cuaderno de Trabajo 5

Grado en Sistemas de la Información, turno de mañana

Grado en Ingeniería Informática, turnos de mañana y tarde

Ejercicios resueltos

1. Programa una estructura de datos que permita almacenar fechas como la agrupación de tres datos, dd, mm y aaaa y una función que calcule si una fecha es anterior a otra.

SOLUCION PROPUESTA:

```
class Fecha:
    def __init__(self,dd,mm,aa):
        self.dia = dd
        self.mes = mm
        self.anno = aa

def es_anterior(fecha1,fecha2):
""" Fecha, Fecha -> bool
 OBJ: Calcula si una fecha es anterior a otra
 PRE: fecha1 y fecha2 son fechas válidas """
    if (fecha2.anno > fecha1.anno):
        respuesta = True
    elif (fecha1.anno > fecha2.anno):
        respuesta = False
    else: # si los dos años son iguales
        if (fecha2.mes > fecha1.mes):
            respuesta = True
        elif (fecha1.mes > fecha2.mes):
            respuesta = False
        else: # el mes y el año son iguales
            respuesta = (fecha2.dia > fecha1.dia)
    return respuesta
```

2. Se pide ahora programar una función que evalúe si dos fechas son iguales y otra que indique si una fecha es posterior a otra. Trate de reutilizar la función del problema 1.

SOLUCION PROPUESTA:

```
def iguales (fecha1,fecha2):
"""Fecha, Fecha -> bool
   OBJ: Calcula si una fecha es igual a otra
   PRE: fecha1 y fecha2 son fechas válidas """
   return  fecha1.dia == fecha2.dia and \
           fecha1.mes == fecha2.mes and \
           fecha1.anno == fecha2.anno
```

Una alternativa a la anterior, reutilizando la función del apartado 1 (si bien un tanto rebuscada) sería:

```
def iguales2(fecha1,fecha2):
"""Fecha, Fecha -> bool
   OBJ: Calcula si una fecha es igual a otra
   PRE: fecha1 y fecha2 son fechas válidas """
   return not es_anterior(fecha1,fecha2) and \
          not es_anterior(fecha2,fecha1)

def es_posterior(fecha1,fecha2):
"""Fecha, Fecha -> bool
   OBJ: Calcula si una fecha es igual a otra
   PRE: fecha1 y fecha2 son fechas válidas """
      return not es_anterior(fecha1,fecha2) and \
             not iguales(fecha1,fecha2)


>>> f2=Fecha(1,1,2070)
>>> f1=Fecha(1,2,2070)
>>> es_anterior(f1,f2)
False
>>> es_posterior(f1,f2)
True
>>> iguales(f1,f2)
False
```

3. Programa una función para leer datos de tipo Fecha por teclado

SOLUCION PROPUESTA:

```
def leer_fecha():
""" nada -> Fecha
  OBJ: Lee valores para los atributos de una fecha,
       no controla si es una fecha válida """
    dia = int(input("Introduce dia: "))
    mes = int(input("Introduce mes: "))
    anno = int(input("Introduce año: "))
    f = Fecha (dia, mes, anno)
    return f
```

4. Programa una función para mostrar datos de tipo Fecha por pantalla.

SOLUCION PROPUESTA:

```
def mostrar_fecha(f):
    """ Fecha -> nada
    OBJ: Muestra los atributos de una fecha por pantalla
    PRE: f es una fecha válida """
    print(f.dia,'/',f.mes,'/',f.anno)
```

5. Hacer un programa que cree un diccionario con personas (nombre-edad) y permita buscar si una persona 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. Implementa las estructuras de datos que permitan almacenar información anónima sobre personas con objeto de hacer un estudio estadístico. Así, se deberá almacenar el número de secuencia, sexo y edad de cada persona. Programe además un par de funciones para (a) leer por teclado datos relativos a una persona y (b) para mostrar dichos datos por pantalla.
2. Sin utilizar listas, programa un software que utilice lo programado en el ejercicio 1 para leer la edad de 10 personas y calcular 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. Amplíe 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 Punto2D 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 estructura "Cuadrado" que incluya información sobre sus 4 vértices y sobre su punto central, todos los cuales seran del tipo Punto2D.
7. Haciendo uso de lo programado en ejercicios anteriores, programe una función que dilucide si un cuadrado es mayor que otro.
8. Haciendo uso de la estructura Fecha de los ejercicios resueltos, implementa una estructura Cronologia que permita llevar la cuenta de 3 fechas importantes en la vida de toda persona: día de su nacimiento, fecha en que celebró su matrimonio y fecha de deceso.
9. Reutilizando lo programado en el ejercicio 8, implementa funciones para leer las fechas importantes de una persona y comprobar si son o no correctas (el orden ha de ser nacimiento, matrimonio, deceso, obligatoriamente). Si no lo fueran debería indicarse y solicitarse de nuevo.
10. Investiga (utilizando para ello Internet y/o la bibliografía de la asignatura) cómo preparar una clase para utilizar con ella el método "print" que empleamos para mostrar la información de otros tipos de datos. Se trata de permitir que una sentencia como `print(p)`, donde `p` es una referencia de tipo `Punto2D` de coordenadas `10,7` produzca una salida como la siguiente:

```
>>> print(p)
Punto2D -> x=10, y=7
```

11. Implemente una aplicación que solicite 10 valores y los almacene en un array. A continuación, recorra el array 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). Modularice su solución.

12. Implemente una aplicación para ayudar en la gestión de cobros de una gasolinera. Mediante 3 arrays deberá calcular el gasto de un total de 10 clientes. El primer array será utilizado para almacenar el gasto de cada cliente en gasolina, el segundo array será utilizado para almacenar el gasto en productos de la tienda de la gasolinera, el tercer array almacenará la suma de los dos arrays anteriores. La aplicación:
    a. Solicitará por teclado los gastos de gasolina y de la tienda para cada uno de los 10 clientes.
    b. Mostrará por pantalla la suma de los gastos para cada cliente.

13. Realice una aplicación que permita sumar dos matrices de tamaño 3x4. Las matrices contienen datos sobre Puntos2D y que realice las siguientes tareas:
    a. Solicitar datos para la matriz A.
    b. Solicitar datos para la matriz B.
    c. Presente el resultado de matriz A + matriz B.

14. 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.

15. Realice 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).

    Nota: Para modelar las casillas del tablero utilice un registro con dos campos, uno que guardará si hay ficha o no (valor booleano) y otro de qué color es la ficha (amarilla o roja). Si en una casilla no hay ficha, el valor del color no tendrá validez y por tanto se ignorará.

16. Realizar un programa que lea palabras hasta que se introduzca "*fin*", mostrando una estadística de las longitudes de las palabras, es decir, el número total de palabras de longitud 1 que se hayan introducido, el total de longitud 2, etc. No hay máxima longitud de las palabras. Una posible salida de este programa sería:

```
Palabras longitud 1: ninguna
Palabras longitud 2: 10
Palabras longitud 3: 21
Palabras longitud 4: 54
etc.
```

17. Crear un diccionario "agenda telefónica" donde la clave sea el nombre de una persona y el valor sea el teléfono. Programar una función para llenarlo y otra para mostrarlo. Para llenarlo, es necesario ir pidiendo contactos hasta el usuario diga que no quiere insertar más. Obviamente, no es válido introducir nombres repetidos.
18. Programar una función que reciba un diccionario y una lista y que como salida genere dos listas:
    a. una lista con todos los valores de aquellos elementos de la lista que están en el diccionario,
    b. otra lista con los valores de los elementos que NO están en el diccionario.
19. 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:
    - Si la persona no existe la agrega al diccionario con una lista que contiene un solo elemento.
    - 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.

20. Suponga 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 2017-2018 (número de jornada y lista de jugadores que jugaron esa jornada). Programe funciones para:
    - Introducir los datos de una jornada a partir de su número y una lista de jugadores y dorsales. Obviamente no pueden repetirse ningún jugador en la alineación ni participar jugadores que no pertenecen a la plantilla del Barcelona FC.
    - Determinar si un jugador participó o no en una cierta jornada a partir de su dorsal o de su nombre (como se prefiera).
    - Listar todas las jornadas en que jugó un determinado jugador cuyo dorsal se indica.
    - Mostrar el nombre del jugador que más partidos jugó en una temporada, indicando el número de partidos que jugó.