Práctica 1. Conceptos de programación de TADs. Tipos básicos

Ejercicio 1: El código siguiente utiliza una clase para codificar las propiedades de un usuario de un sistema. Incluye los siguientes aspectos de relevancia:

- Un constructor, que recibe como parámetros el nombre, apellidos y edad del usuario. En él se inicializa el valor de los distintos campos que almacenan los objetos instanciados a partir de la clase.
- Los atributos o campos *nombre*, *apellidos* y *edad* son declarados como privados, usando los dos subrayados que preceden al nombre de cada uno.
- Un método que imprime por pantalla los distintos campos del usuario. Notar que las cadenas se imprimen de dos formas distintas, una concatenándolas con el operado "+", y otra mediante *cadenas f*.
- Otro que imprime un saludo usando el nombre almacenado.
- Un método final que incrementa el valor de un campo de la clase, concretamente el de `edad`.
- Notar el uso del campo `self` como identificación del propio objeto. Si no se indica, se crearían variables de ámbito local dentro de cada método, que no quedarían almacenadas en el objeto instanciado de la clase.

```
class Usuario:

    def __init__(self, nombre, apellidos, edad):
        self.__nombre = nombre
        self.__apellidos = apellidos
        self.__edad = edad

    def describir_usuario(self):

        print("***Usuario***")
        print("\tNombre: " + self.__nombre)
        print("\tApellidos: " + self.__apellidos)
        print(f"\tEdad: {self.__edad}")

    def saludar_usuario(self):
        print(f"Hola {self.__nombre}")

    def incrementar_edad(self):
        self.__edad += 1
```

Apartado a: Crea un proyecto en Visual Studio Code añadiendo o abriendo una carpeta de trabajo donde crear un nuevo fichero de código *Usuario.py*. Incrusta el código anterior y añade el siguiente código, que hace uso de la clase `Usuario` creando un objeto de la clase llamado `jose`.

```
if __name__ == '__main__':
    jose = Usuario('Jose', 'Santa Lozano', 40)
    jose.describir_usuario()
    jose.saludar_usuario()
    jose.incrementar_edad()
```

```
    jose.describir_usuario()
```

Prueba el funcionamiento del programa ejecutándolo y visualizando la salida por el terminal. Es importante observar cómo este código es ejecutable cuando el fichero *Usuario.py* se manda a interpretar directamente a Python, ya que se comprueba si este fichero (módulo) es el "principal" en la invocación al intérprete.

Apartado b: Modifica el anterior programa para que la clase guarde también la dirección del usuario, y la imprima dentro del método `describir_usuario`. Modifica el código de prueba de la clase para comprobar el correcto funcionamiento de estos cambios.

Apartado c: Añade un método adicional `clasifica_usuario` a la clase `Usuario` para que, en función de la edad del usuario, imprima un mensaje indicando si se trata de un bebé, un niño, un adolescente, un adulto o un anciano. Añade el código necesario para probar esta nueva funcionalidad.

Ejercicio 2: Crea un nuevo proyecto en una nueva carpeta donde se utilice una clase para guardar los datos de una empresa:

- Nombre.
- CIF.
- Facturación, en euros.
- Año de creación.
- Número de empleados.

Añade los métodos necesarios, además del constructor, para:

- Mostrar toda la información de la empresa.
- Modificar el valor de la facturación.
- Incrementar/decrementar el número de empleados.

Añade el código necesario para comprobar el correcto funcionamiento de la clase. Posteriormente, realiza un bucle de ejecución donde en cada iteración se sume una unidad al número de empleados y se imprima la información de la empresa. Puedes elegir el número de iteraciones.

Ejercicio 3: Ejecutar el siguiente código y analizar y explicar su resultado

```
import sys
data = []
for k in range(27):
    a = len(data)
    b = sys.getsizeof(data)
    print('Longitud ', a, ' Tamaño en bytes ', b)
    data.append(None)
```

Ejercicio 4: Desarrollar el TAD "Fraccion", que permita tratar valores formados por dos elementos, numerador y denominador, con el denominador distinto de 0, y considerando que las fracciones negativas tienen el numerador negativo. Queremos que el TAD nos permita tratar las fracciones como cualquier otro tipo de dato numérico. Debemos de poder sumar, restar, multiplicar y dividir fracciones.

- Realizar la especificación del TAD Fracción que permita realizar las operaciones básicas con fracciones.
- Implementar el constructor de la clase y la operación suma.
- Implementar el método mágico `__str__(self)` para que se visualice correctamente una fracción en pantalla.
- Implementar los operadores aritméticos `__sub__`, `__mul__` y `__truediv__`
- Implementar los operadores relacionales `__eq__`, `__gt__`, `__ge__`,`__lt__`,`__le__` y `__ne__`
- Realizar las siguientes operaciones:
    - 4/5+2/6
    - (1/2*1/2)-5/4
    - Comprueba el correcto comportamiento de los operadores relacionales

Ejercicio 5: Desarrollar el TAD "Fecha" para la representación de fechas válidas según el calendario Gregoriano, considerando las siguientes precondiciones para los números naturales que vayan a denotar el día, mes y año de una fecha en concreto: día∈[1,31], mes∈[1,12] y año≠0, además de tener en cuenta que hay ciertos meses con un número de días menor de 31; abril, junio, septiembre y noviembre tienen 30 días, mientras que febrero 28 (no considerar años bisiestos).

- Realizar la especificación informal del TAD Fecha para el almacenamiento y recuperación de cada elemento individual de una fecha.
- Implementar el constructor de la clase y el método mágico `__str__(self)` con el que obtener la representación de la fecha en formato día/mes/año para su posterior impresión en pantalla.
- Implementar dos funciones para devolver la fecha anterior y posterior a una fecha dada.
- Realizar distintas pruebas para comprobar el correcto funcionamiento del TAD Fecha.