PROGRAMACIÓN CON PYTHON

(CEFIRE CTEM)

POO en Python (parte 1)
Ejercicios voluntarios

Autora: María Paz Segura Valero (segura_marval@gva.es)

CONTENIDO

1. Introducción

En este documento puedes encontrar una serie de **ejercicios voluntarios** para que pongas en práctica lo aprendido durante esta unidad.

En el aula virtual dispondrás de las soluciones pero te recomiendo que intentes solucionarlos por ti mismo/a porque, aunque no sean obligatorios para superar el curso, sí pueden ayudarte a enfrentarte al *ejercicio obligatorio* del final.

Puedes utilizar el **foro** del curso para consultar tus dudas.

2. Ejercicios voluntarios

2.1. Ejercicio 1: Vuelo

Crea una clase **Vuelo** que posea los siguientes atributos de objeto: *origen*, *destino*, *dia* y *clase*. Además, deberás ofrecer los siguientes métodos de instancia:

- `__init__()` que se encargue de recibir por parámetro los valores necesarios para informar los atributos de objeto. El valor por defecto para el atributo `clase` es `'turista'`.
- `imprimir()` que mostrará por pantalla el valor de los atributos del objeto.

Recuerda incluir el parámetro especial **self** en los dos métodos anteriores.
No es necesario que contemples el encapsulamiento de objetos.

A continuación puedes ver un ejemplo de uso de la clase.

```
>>> v1 = Vuelo('Valencia', 'Menorca', '15-08', 'primera')
>>> v1.imprimir()
Datos del vuelo:
origen --> Valencia
destino --> Menorca
día --> 15-08
clase --> primera
>>> v2 = Vuelo('Valencia', 'París', '09-04')
>>> v2.imprimir()
Datos del vuelo:
origen --> Valencia
destino --> París
día --> 09-04
clase --> turista
```

Añade la siguiente estructura al final del fichero e incluye algunas instrucciones de prueba de la clase:

```
if __name__ == "__main__":
    #instrucciones de prueba
```

En el aula virtual dispones de la solución del ejercicio en el fichero **ud3_ejer1_vuelo.py**.

2.2. Ejercicio 2: Vuelo ampliado

Vamos a modificar la clase **Vuelo** del ejercicio anterior para reescribir algunos métodos especiales. Para ello, ten en cuenta las siguientes indicaciones:

- Reescribe el método `__str__()` para que devuelva una cadena de texto con los valores de los atributos de un vuelo. Por ejemplo:

  ```
  Origen: Valencia - Destino: Menorca - Día: 15-08 - Clase: primera
  ```

  Solo habrá que pasarle como parámetro de entrada el atributo especial **self**.

  Este método se ejecutará automáticamente cuando se utilice `print()` o `str()` con un objeto.

- Reescribe el método `__eq__()` para que compare dos vuelos. Para ello se pasarán como parámetros de entrada el atributo **self** y otro objeto **Vuelo**. El método devolverá **True** cuando los valores de los atributos de los dos vuelos sean iguales y **False** en caso contrario.

A continuación puedes ver algunos ejemplos de uso de la clase.

Ejemplo 1: Creamos un vuelo y usamos la función `print()` con él.

```
>>> v1 = Vuelo('Valencia', 'Menorca', '15-08', 'primera'); v1.imprimir()
Datos del vuelo:
origen --> Valencia
destino --> Menorca
día --> 15-08
clase --> primera
>>> print(v1)
Origen: Valencia - Destino: Menorca - Día: 15-08 - Clase: primera
```

Ejemplo 2: Creamos un vuelo y usamos la función de conversión a cadenas `str()` con él.

```
>>> v2 = Vuelo('Valencia', 'París', '09-04'); v2.imprimir()
 Datos del vuelo:
 origen --> Valencia
 destino --> París
 día --> 09-04
 clase --> turista
>>> str(v2)
'Origen: Valencia - Destino: París - Día: 09-04 - Clase: turista'
```

Ejemplo 3: Creamos varios vuelos y los comparamos entre sí.

```
>>> v1 = Vuelo('Valencia', 'Menorca', '15-08', 'primera')
>>> v2 = Vuelo('Valencia', 'París', '09-04')
>>> v3 = Vuelo('Valencia', 'Menorca', '15-08', 'primera')
>>> v1 == v2
False
>>> v1 == v3
True
```

No es necesario que contemples el encapsulamiento de objetos.

En el aula virtual dispones de la solución del ejercicio en el fichero **ud3_ejer2_vuelo_ampliado.py**.

2.3. Ejercicio 3: Contacto de agenda

Crea una clase llamada **Contacto** con los siguientes elementos:

- Atributos de clase: `idioma`, que almacenará el idioma en el que mostrar la información por pantalla. Valores posibles: `'cs'`, `'va'`, `'en'`.
- Atributos de objeto: `nombre`, `telefono`, `cuenta_correo`. Todos ellos de tipo cadena de texto.
- Métodos de instancia:
  - `__init__()` que se encargue de recibir por parámetro los valores necesarios para informar los atributos de objeto.
  - `imprimir()` que mostrará por pantalla el valor de los atributos del objeto.
  - Recuerda incluir el parámetro especial **self** en los dos métodos anteriores.
- Método de clase: `cambiarIdioma(cls, idioma)` que comprobará si el `idioma` pasado como parámetro es uno de los posibles. Si es así, modificará el atributo de

la clase. Si no lo es, pondrá el idioma 'cs' y generará un error (`raise ValueError`).

- Método estático: `validarCuentaCorreo(cc)` que utilizará la función `find()` y devolverá su resultado. Es decir, si contiene el carácter entonces devolverá la posición de la cadena en la que está y si no, devolverá un -1. Ejemplo de uso: ***cadena*.find(*caracter*)**

A continuación puedes ver algunos ejemplos de uso de la clase.

Ejemplo 1: Creamos el contacto `c1` y mostramos su información.

```
>>> c1 = Contacto("Ana", "606773344", "ana@dominio.org")
>>> c1.imprimir()
 Contacto:
   nombre --> Ana
   teléfono --> 606773344
   cuenta de correo --> ana@dominio.org
```

Ejemplo 2: Cambiamos el idioma por uno correcto.

```
>>> Contacto.cambiarIdioma("va"); Contacto.idioma
'va'
```

Ejemplo 3: Cambiamos el idioma por uno incorrecto.

```
>>> Contacto.cambiarIdioma("kk")
 Traceback (most recent call last):
   File "<pyshell>", line 1, in <module>
   File "E:\20-21 IES José Rodrigo Botet\MPAZ PROFE\CEFIRE\PYTHON
 voluntarios\ud3_ejer2_contacto.py", line 41, in cambiarIdioma
     raise ValueError("Idioma incorrecto")
 ValueError: Idioma incorrecto
>>> Contacto.idioma
'cs'
```

Ejemplo 4: Validamos varias cuentas de correo.

```
>>> c1 = Contacto("Ana", "606773344", "ana@dominio.org")
>>> c1.validarCuentaCorreo(c1.cuenta_correo)
3
>>> Contacto.validarCuentaCorreo("ana")
-1
```

No es necesario que contemples el encapsulamiento de objetos.
En el aula virtual dispones de la solución del ejercicio en el fichero **ud3_ejer3_contacto.py**.

2.4. Ejercicio 4: Contacto con encapsulamiento

Vamos a realizar los siguientes cambios en la clase **Contacto** del ejercicio anterior para contemplar el encapsulamiento de los datos:

- Añadir un método estático llamado **validarTelefono** que reciba como parámetro de entrada una cadena de texto y compruebe si se trata de un número de teléfono válido (podemos convertirlo a entero y ver que tiene una longitud de 9 dígitos). Si el número de teléfono es válido entonces devolverá un 0 y si no, un -1.
- Añadir métodos **getter** a los tres atributos de objeto de la clase.
- Añadir métodos **setter** a los tres atributos de objeto de la clase, teniendo en cuenta los siguientes puntos:
  - Para el atributo **nombre** no hay que hacer ninguna validación de formato. Directamente cambiaremos el valor del atributo oculto **self.__nombre** por el valor pasado por parámetro.
  - Para el atributo **telefono** utilizaremos la función **validarTelefono()** que comprobará si el formato del número de teléfono es correcto. Si lo es, cambiaremos el valor del atributo oculto **self.__telefono** por el valor pasado por parámetro. Si no lo es, cambiaremos el valor del atributo por una cadena vacía y lanzaremos una excepción **ValueError** con un mensaje de aviso.
  - Para el atributo **cuenta_correo** utilizaremos la función **validarCuentaCorreo()** que comprobará si el formato de la cuenta de correo es correcto. Si lo es, cambiaremos el valor del atributo oculto **self.__cuenta_correo** por el valor pasado por parámetro. Si no lo es, cambiaremos el valor del atributo por una cadena vacía y lanzaremos una excepción **ValueError** con un mensaje de aviso.

A continuación puedes ver algunos ejemplos de uso de la clase.

Ejemplo 1: Creamos un contacto con todos los datos correctos.

```
>>> c1 = Contacto("Ana", "606773344", "ana@dominio.org"); c1.imprimir()
Contacto:
  nombre --> Ana
  teléfono --> 606773344
  cuenta de correo --> ana@dominio.org
```

Ejemplo 2: Intentamos crear un contacto con el formato de teléfono incorrecto.

```
>>> c2 = Contacto("Ana", "kk", "ana@dominio.org")
 Traceback (most recent call last):
   File "<pyshell>", line 1, in <module>
   File "E:\20-21 IES José Rodrigo Botet\MPAZ PROFE\CEFIRE\PYTHON 30h [AVA
 voluntarios\ud3_ejer3_contacto_encapsulamiento.py", line 13, in __init__
     self.telefono = telefono
   File "E:\20-21 IES José Rodrigo Botet\MPAZ PROFE\CEFIRE\PYTHON 30h [AVA
 voluntarios\ud3_ejer3_contacto_encapsulamiento.py", line 40, in telefono
     raise ValueError("El formato del número de teléfono es incorrecto.")
 ValueError: El formato del número de teléfono es incorrecto.
```

Ejemplo 3: Intentamos crear un contacto con el formato de la cuenta de correo incorrecto.

```
>>> c3 = Contacto("Ana", "606773344", "kk")
 Traceback (most recent call last):
   File "<pyshell>", line 1, in <module>
   File "E:\20-21 IES José Rodrigo Botet\MPAZ PROFE\CEFIRE\PYTHON 30h [AV
 voluntarios\ud3_ejer3_contacto_encapsulamiento.py", line 14, in __init__
     self.cuenta_correo = cuenta_correo
   File "E:\20-21 IES José Rodrigo Botet\MPAZ PROFE\CEFIRE\PYTHON 30h [AV
 voluntarios\ud3_ejer3_contacto_encapsulamiento.py", line 48, in cuenta_c
     raise ValueError("El formato de la cuenta de correo es incorrecto.")
 ValueError: El formato de la cuenta de correo es incorrecto.
```

En el aula virtual dispones de la solución del ejercicio en el fichero **ud3_ejer4_contacto_encapsulamiento.py**.