Pruebas técnicas Python

Problemas

Problema 1

Escribir una función que, dado un entero positivo devuelva `True` si el número es un **primo circular** y `False` en otro caso.

Nota: Un número es **primo circular** si es un número primo y todas las rotaciones de sus dígitos son números primos también. Por ejemplo: **197** lo es ya que **197**, **971** y **719** son números primos.

Problema 2

Escribir una función que, dado un entero positivo `True` si todos los dígitos de número son diferentes `False` en otro caso.

Ejemplo:

```
>> func(123)
True
>> func(222)
False
```

Problema 3

Escribir una función que, dado un entero positivo **mayor que 100** genere una cadena de texto siguiendo los siguientes pasos:

1. Sumar todos los dígitos de número pasado como parámetro.
2. Repetir el cálculo descrito en el apartado 1 hasta que la suma caiga en un número entre 0 y 9.

El código final sera el resultado de añadir a la izquierda el número calculado al número original.

Ejemplo:

```
>> func(69810)
669810
>> func(3201)
63201
```

Problema 4

Escribir una clase llamada **CustomString** que será una clase string que solo permitirá incluir caracteres alfabéticos. De la **a** la **z** (mayúsculas y minúsculas, la letra ñ quedaría excluida) y espacios.

Ejemplo:

```
>> string1 = CustomString("Ho,la")
La instancia creada solo puede contener caracteres de la "a" a la "z"
El caracter "," se elminará
>> string1 + ","
No puedes añadir el caracter "," a la cadena
>> string1 + " mundo"
>>Hola mundo
```

Problema 5

Escribir una función que, dada una lista de enteros devuelva una tupla con los siguientes elementos: una subsecuencia que cumpla que la suma de sus elementos es maximal y la suma de estos.

Ejemplo:

```
>> func([-8, -4, 6, 8, -6, 10,-4,-4])
([6,8,-6,10], 18)
```

Problema 6

Escribir una función que, dada una lista de **unos** y **ceros**. Devuelva una lista de tal manera que todos los **ceros** estén al principio y todos los **unos** estén al final.

Ejemplo:

```
>> func([0,1,1,1,0])
[0,0,1,1,1]
```

Problema 7

En una tienda de componentes informáticos disponen de dos tipos de ordernadores: portátiles y de sobremesa. Ambos tipos de ordenadores tienen atributos comunes como: marca, modelo. Los orderadores de sobremesa tienen una caracterísitca única que es el volumen de su CPU mientras que los portátiles tienen como característica única la duración de la batería. Se pide modelar un sistema que permita ver las características de cada equipo con un simple `print`.

Ejemplo:

```
>> s = Sobremesa("Dell", "Power", 10)
>> p = Portatil("Dell", "Mini Power", 2)
```

```
>> print(s)
Marca: Dell, Modelo: Power, Volumen: 10 L
>> print(p)
Marca: Dell, Modelo: Mini Power, Batería: 2 h
```

Problema 8

Queremos construir una base de datos de usuarios. Los datos que necesitamos de cada usuario son: nombre y contraseña. Es importante conocer en número de usuarios creados por que es necesario implementar un mecanismo que nos permita determinar ese número en tiempo de ejecución. Se pide crear un sistema que nos permita crear usuarios y, en un momento dado, conocer el número de usuarios creados.

Ejemplo:

```
>> u1 = Usuario("Usuario", "Password")
>> u2 = Usuario("Usuario2", "Password2")
>> print(FUNCIONALIDAD A IMPLEMENTAR)
El número de usuarios creados es 2.
```

Problema 9

Necesitamos gestionar nuestros archivos y para ello vamos crear una clase que, dado un directorio **que únicamente contiene archivos** nos permita:

- Listar los archivos contenidos en el directorio
- Crear un nuevo archivo vacio
- Eliminar un archivo
- Optener la extensión de cualquier archivo **independientemente de si esta en el directorio seleccionado**.

La clase creada anteriormente es muy util pero peligrosa. Para limitar su poder a **solo archivos de audio** será necesario crear otra clase qué nos permita las mismas funcionalidades pero solo con arhivos que tengan las extensiones: .wav y .mp3.

Ejemplo:

```
>> a = Archivos("directorio/con/archivos")
>> a.listar()
foo.txt
boo.mp3
>> a.crear("new.txt")
>> a.listar("new.txt")
foo.txt
boo.mp3
new.txt
>> a.eliminar("new.txt")
>> a.listar("new.txt")
foo.txt
```

```
boo.mp3
>> m = ArchivosAudio("directorio/con/archivos")
>> m.listar()
boo.mp3
>> FUNCIONALIDAD A IMPLEMENTAR("archivo.ext")
ext
```

Problema 10

Se desea crear un sistema de compras que, dado un conjunto de productos (con stock limitado), nos permita añadirlos a un carrito de la compra. Las funcionalidades que debemos incorporar son las siguientes:

- Crear un carrito de compra
- Añadir productos al carrito
- Controlar el stock de los productos
- Eliminar productos de carrito (total o parcialmente)
- Mostrar información de carrito: productos, coste por producto, coste total, etc.

Ojo: El stock disponible de los productos es limitado por lo que debemos incorporar a nuestra implementación un sistema que nos pemita limitar la compra en función del stock.

Se deberá entregar una propuesta con una explicación detallada de su funcionamiento y su correspondiente implementación.

Nota: El uso del sistema de compras será programático **no es necesario crear ni bases de datos externas ni interfaces gráficas**.

Problema 11

Construir un script que haga **1000** peticiones a una dirección web cualquiera (parámetro de entrada). Es importante que el script se ejecute de la manera más rápida posible.

```
>> python script.py https://httpbin.org/get
Hecho! Tempo total X segundos
```