Desafíos de funciones

Instrucciones

A continuación de detallan variados desafíos a desarrollar. Para su correcta corrección, los programas deben ser almacenados en un comprimido `.zip` de la siguiente manera:

Desafio-funciones.zip

├── concatena_letras.py

├── letra_o.py

├── letra_i.py

├── letra_x.py

├── menu_banco.py

NOTA:

- Los ejercicios que piden funciones se evalúan llamando a la función directamente y comparando el resultado.
- Para tener la evaluación correcta del ejercicio, se considera el nombre de la función y el resultado.
- No es necesario que el programa tenga una salida o muestre algo en pantalla por sí solo.

Desafío - Concatenando letras

Crear una función llamada `gen` que reciba el número de letras a generar, y devuelva un string con todas las letras generadas concatendas.

Ejemplo:

gen 4

> "abcd"

gen 10

> "abcdefghij"

Desafío - Programa de letras

1. La función `letra_o(n)`, la cual al ser llamada, retornará una cadena de texto que al ser imprimida con *print*, dibujará una letra "o" según el ejemplo

> print(letra_o(n))

```
*****
*   *
*   *
*   *
*****
```

2. La función `letra_i(n)`, la cual al ser llamada, retornará una cadena de texto que al ser imprimida con *print*, dibujará una letra "i" según el ejemplo

> print(letra_i(5))

```
*****
  *
  *
  *
*****
```

3. La función `letra_x(n)`, la cual al ser llamada, retornará una cadena de texto que al ser imprimida con *print*, dibujará una letra "x" según el ejemplo

> print(letra_x(5))

```
*   *
 * *
  *
 * *
*   *
```

Desafío - Menú de banco (Opcional)

Crear un programa que al ser ejecutado llame a la función `mostrar_menu(saldo = x)` la cual debe mostrar un menú con las siguientes opciones:

> Bienvenido al portal del Banco Amigo. Escoja una opción:
>
> 1. Consultar saldo
> 2. Hacer depósito
> 3. Realizar giro
> 4. Salir

El parámetro `saldo` es *opcional*, y corresponde al saldo inicial con el que inicia el programa al ejecutarse. Por defecto debe ser `0`.

El programa debe contar además con las siguientes funciones:

Estas funciones ***no deben ser llamadas al ejecutar el programa***. El llamado hacia cada una se explica después en la sección de requerimientos.

`depositar(saldo, cantidad)`

Función que recibe los parámetro `saldo` (int) y `cantidad` (int). Debe retornar el nuevo `saldo`, correspondiente al `saldo` ingresado más la `cantidad`.

`girar(saldo, cantidad)`

Función que recibe los parámetros `saldo` (int) y `cantidad` (int). Debe validar que `cantidad` no exceda a `saldo`. Si es así, debe retornar `False`. En caso contrario, debe restar esta `cantidad` a `saldo`, y retornar el resultado.

Requerimientos

1. Una vez que el usuario escoge una opción, se debe validar que las únicas opciones que se pueda ingresar son `1`, `2`, `3` o `4`. Si se introduce otra opción, se debe mostrar el mensaje `"Opción inválida. Por favor ingrese 1, 2, 3 ó 4."`, y volver a solicitar una opción.
2. Las funciones `depositar()` y `girar()` ***no deben contener*** `print()`. Las salidas deben manejarse desde `mostrar_menu()`.
3. Al escoger la opción `1`, se debe mostrar el saldo actual en pantalla. Luego se vuelve a mostrar el menú.
4. Al escoger la opción `2`, se debe solicitar la cantidad a depositar, y con ella llamar a la función `depositar(saldo, cantidad)`. Luego se debe mostrar el nuevo saldo en pantalla, y volver a mostrar el menú.
5. Al escoger la opción `3`, primero se debe validar que exista saldo (debe ser mayor a 0). Si no existe saldo, debe mostrar mensaje `"No puede realizar giros. Su saldo es 0"`. Si existe saldo, se debe solicitar la cantidad a girar, y con ella llamar a la función `girar(saldo, cantidad)`.

- Si la función retorna `False`, se debe mostrar el mensaje `"No se puede girar esta cantidad. Su saldo es de "` (concatenar el valor de `saldo`). Se debe solicitar nuevamente la cantidad a girar y volver a llamar a la función `girar(cantidad)`. **Nota**: El operador de comparación ***is*** es más estricto que `==`. En Python, de manera general se considera el valor `0` como `False`. Si se desea comprobar que un objeto tiene ***exactamente*** el valor `False` (y no `0`), se debe usar **is**.
- Si la función retorna un nuevo saldo (*o sea, "no retorna `False`"*), se debe mostrar el nuevo saldo en pantalla, y volver a mostrar el menú.

5. Solo se detiene la ejecución del programa al escoger la opción `4`.
6. **El saldo debe conservarse**, de acuerdo a las operaciones que se realicen, durante una misma ejecución del programa.

Ejemplo de flujo esperado

```
¡Bienvenido al Banco Amigo!. Escoja una opción:
1. Consultar saldo
2. Hacer depósito
3. Realizar giro
4. Salir

1

Su saldo es de 0

¡Bienvenido al Banco Amigo!. Escoja una opción:
1. Consultar saldo
2. Hacer depósito
3. Realizar giro
4. Salir

2

1000

Su nuevo saldo es de 1000

¡Bienvenido al Banco Amigo!. Escoja una opción:
1. Consultar saldo
2. Hacer depósito
3. Realizar giro
4. Salir

1

Su saldo es de 1000
```