PRO - UT1-AE2. Prueba práctica

Ejercicios

A1. Comprobar validez contraseña

Edita el archivo adjunto **a1.py** que se adjunta. Su contenido es el siguiente:

```
def valid_passwd(passwd):
    # Completar código de la función
    pass # eliminar esta línea

passwd1 = "As5rtG"
valid_passwd(passwd1)
passwd2 = "lej50*paRapa"
valid_passwd(passwd2)
passwd3 = "lej50paapa"
valid_passwd(passwd2)
```

A la función `valid_passwd()` le pasamos una cadena de texto con una contraseña y muestra si es o no válida. Además, en caso de que la contraseña no sea válida el programa debe mostrar un mensaje indicando todos los criterios que no cumple.

Se considera que una contraseña es válida si:

- Tiene longitud de 10 o más caracteres
- Contiene, al menos, un caracter en mayúscula
- Contiene, al menos, uno de los caracteres especiales de la siguiente cadena: "*#+-.,()[]{}"

El resultado de ejecutar el programa anterior debería ser:

```
La contraseña As5rtG no es válida porque:
- Su longitud es de menos de 10 caracteres
- No contine caracteres especiales

La contraseña lej50*paRapa es válida

La contraseña lej50paapa no es válida porque:
- No contiene caracteres en mayúscula
- No contiene caracteres especiales
```

A2. Tratamiento de un texto

Completar el código del archivo adjunto **a2.py** de forma que las funciones realicen lo que se especifica en su documentación. El contenido del archivo es el siguiente:

```
def count_phrases(text):
    """
    Función a la que se le pasa un texto y devuelve el número de frases que
contiene
    El caracter delimitador de frases es el punto '.'
    """
```

```
    # Completar código

def phrases_in_text(text):
    """
    Función a la que se le pasa un texto y devuelve una lista con las frases que
contiene
    El caracter delimitador de frases es el punto '.'
    """
    # Completar código


def ordered_words(text):
    """
    Función a la que le pasamos un texto y devuelve una lista con las palabras
que contiene
    La lista debe estar ordenada alfabéticamente
    Todas las palabras deben estar en minúscula
    """

    # Completar código

def n_words_end_in(letter, text):
    """
    Función a la que se le pasa una letra y un texto y devuelve el número
    de palabras de la lista que terminan por dicho carácter
    """
    # Completar código

def delete_word(text, word):
    """
    función a la que le pasamos un texto y una palabra y devuelve el mismo texto
    sin que aparezca la palabra a eliminar
    """
    # Completar código

paragraph = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
vitae posuere nunc. Pellentesque varius metus id risus dictum tempor. Duis ac dui
ac mauris commodo aliquet. Mauris vitae nisl non massa; fringilla ullamcorper
quis nec libero. Interdum et malesuada fames; ac ante ipsum primis in faucibus.
Duis fermentum erat eget ipsum cursus, in ullamcorper libero mattis. Nulla ac
feugiat nulla, non molestie lacus. Sed sit amet elementum lectus. Integer id dui
efficitur, mattis velit a, efficitur ligula. Nullam mattis enim nulla, eu
molestie sem eleifend sit amet. Mauris commodo posuere pretium. Pellentesque
justo metus, auctor ornare sem nec, ornare pharetra elit. "

num_phrases = count_phrases(pararagraph)
print(f"El texto tiene {num_phrases} frases\n")

list_phrases = phrases_in_text(pararagraph)
print(f"Las frases del texto son:\n{list_phrases}\n")

last_phrase = list_phrases[-1]
l_words_ordered = ordered_words(lastphrase)
print("La palabras de la ultima frase en orden alfabético
son:\n{l_words_ordered}\n")

letter = "r"
n_words_end = n_words_end_in(letter, last_phrase)
```

```
print("En la última frase hay {n_words_end} palabra(s) que terminan en
{letter}:\n{l_words_ordered}\n")

phrase4 = list_phrases[3]
text_without_word = delete_word(phrase4, "ac")
print("La cuarta frase sin la palabra 'ac' queda:\n{text_without_word}\n")
```

El programa debería devolver:

```
El texto tiene 12 frases

Las frases del texto son:
['Lorem ipsum dolor sit amet, consectetur adipiscing elit', 'Vivamus vitae
posuere nunc', ..., 'Pellentesque justo metus, auctor ornare sem nec, ornare
pharetra elit']

La palabras de la ultima frase en orden alfabético son:
['auctor', 'elit', 'justo', 'metus', 'nec', 'ornare', 'pellentesque', 'pharetra']

En la última frase hay 1 palabra(s) que terminan en r

La cuarta frase sin la palabra 'ac' queda:
Duis dui mauris commodo aliquet
```

A3. Tragaperras

Completar el código que se adjunta en el archivo **a3.py**. Su contenido es el siguiente

```
def puntuacion(a, b, c):
    # Completar el código de esta función
    pass # Eliminar esta línea
def partida(m, n):
    # Completar el código de esta función
    pass # Eliminar esta línea


# Programa principal
monedas = int(input("Con cuantas monedas quieres empezar: "))
n_tiradas = int(input("Cuantas tiradas quieres realizar: "))
p_final = partida(monedas, n_tiradas)
print(f"Has finalizado con {p_final} monedas")
```

Teniendo en cuenta que:

a) La función de nombre `puntuacion()` a la que le pasamos 3 valores enteros devuelve:

- 0 si todos los parámetros son distintos
- 2 si dos de los parámetros son iguales
- 5 si los tres valores son iguales.

b) La función de nombre `partida()` recibe como parámetros un número que representa el número de monedas iniciales y el número de tiradas y devuelve la puntuación final teniendo en cuenta que:

- Por cada tirada la función genera 3 números al azar del 1 al 5 y calcula la puntuación usando la función `puntuacion()`.

- **Después** de cada tirada se resta una moneda
- Después de restar la moneda de la tirada se muestra la información de la tirada (valores que se generaron y monedas que quedan.)
- Si después de una tirada la puntuación se queda a 0 no se hacen más tiradas y la función devuelve un 0.

Ejemplos de ejecución:

```
Con cuantas monedas quieres empezar: 5
Cuantas tiradas quieres realizar: 4
tirada 1: 4 3 1 -> 4 monedas
tirada 2: 2 1 2 -> 5 monedas
tirada 3: 5 3 4 -> 4 monedas
tirada 4: 5 1 2 -> 3 monedas
Has finalizado con 3 monedas
```

```
Con cuantas monedas quieres empezar: 3
Cuantas tiradas quieres realizar: 5
tirada 1: 2 3 1 -> 2 monedas
tirada 2: 2 1 5 -> 1 moneda
tirada 3: 5 3 4 -> 0 monedas
Has finalizado con 0 monedas
```

A4. Carrito compra tienda online

Completar el código que se adjunta en el archivo **a4.py**. Su contenido es el siguiente.

```python
def obtener_precio(catalogo, producto):
    # completar función
    pass #Eliminar esta línea

def total_compra(catalogo, carrito):
    # completar función
    pass # Eliminar esta línea

def  mostrar_compra(catalogo, carrito):
    # completar funcion
    pass #Eliminar esta línea

# Programa principal

catalogo = [["camiseta roja", 10.50], ["camiseta azul", 10.99], ["camiseta
blanca", 9.90],["pantalon vaquero", 21.50], ["pantalon corto", 13.50],
["chaleco", 17.20]]
carrito = [["camiseta roja", 1],["pantalon corto", 2], ["chaleco", 3]]

p_chaleco = obtener_precio(catalogo, "chaleco")
print(f"El precio del chaleco es de {p_chaleco}€.\n")

total = total_compra(catalogo, carrito)
print(f"El total de la compra es de: {total}€.\n")

print("El extracto de la compra en formato HTML es:\n")

mostrar_compra(catalogo, carrito)
```

Teniendo en cuenta que:

- La lista `catalogo` representa el catálogo de una tienda de ropa online con el nombre de los productos y su precio
- La lista `carrito` almacena los productos seleccionados y su cantidad por un usuario que ha accedido a la tienda online.
- Todos los productos del carrito existen en el catálogo.

Se deben completar las siguientes funciones:

a) La función `obtener_precio()` a la que le pasamos el catálogo y el nombre de un producto y devuelve el precio del mismo

b) La función `total_compra()` a la que le pasamos la lista con el catálogo y la lista con el carrito de la compra y devuelve el valor total de los productos seleccionados.

c) La función `mostrar_carrito()` a la que le pasamos el catálogo y muestra los detalles de la misma en una tabla html de la forma:

```
<table>
<tr>
    <th>Producto</th><th>cantidad</th><th>P U</th><th>P Total</th>
</tr>
<tr>
    <td>Camiseta roja</td><td>1</td><td>10.50</td><td>10.50</td>
</tr>
<tr>
    <td>Pantalon corto</td><td>2</td><td>13.50</td><td>27.0</td>
</tr>
<tr>
    <td>Chaleco</td><td>3</td><td>17.2</td><td>51.6</td>
</tr>
<tr>
    <td colspan='4'>Total: 89,1€</td>
</tr>
</table>
```

Teniendo en cuenta que:

- El nombre de los productos debe aparecer con el primer caracter en mayúscula.
- El precio unitario de cada producto lo obtenemos usando la función `obtener_precio()`
- El total del carrito lo calcule llamando a la función `total_compra()`

> **Nota**: si te resulta complicado mostrar la compra en formato HTML muéstrala por lo menos en formato texto

Después de completar las funciones, el resultado del programa debería ser de la forma:

```
El precio del chaleco es de 17.2€

El total de la compra es de: 89.1€

El extracto de la compra en formato HTML es:

<table>
```

```
<tr>
    <th>Producto</th><th>cantidad</th><th>P U</th><th>P Total</th>
</tr>
<tr>
    <td>Camiseta roja</td><td>1</td><td>10.50</td><td>10.50</td>
</tr>
<tr>
    <td>Pantalon corto</td><td>2</td><td>13.50</td><td>27.0</td>
</tr>
<tr>
    <td>Chaleco</td><td>3</td><td>17.2</td><td>51.6</td>
</tr>
<tr>
    <td colspan='4'>Total: 89,1€</td>
</tr>
</table>
```

Producto

Cuando termines comprime los archivos `.py` en un fichero de nombre `examen.zip` y entrégalos en el aula virtual