Ejercitación_Clase1

August 18, 2021

1 Capacitación Semillero DISW: Curso Python

2 Clase 1

2.1 Ejercicios

2.1.1 Ejercicio 1

Crear un programa que pida al usuario que ingrese su nombre y edad, y luego muestre por pantalla un mensaje que diga " cumplirá 100 años en ."

[ ]:
```
name = input("¿Cuál es tu nombre?: ")
age = int(input("¿Cuátos años tenes?: "))
# TODO: impresión por pantalla.
```

2.1.2 Ejercicio 2

Dada una lista, pedir al usuario que ingrese un valor umbral y devolver todos los valores de la lista menores al elegido por el usuario.

[ ]:
```
umbral = # TODO
lista = [52,10,2,3,6,7,-2,5,8]
# TODO mostrar todos los valores de lisa por debajo de umbral.
```

2.1.3 Ejercicio 3

Dadas dos listas, por ejemplo:

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

escribir un programa que devuelva una lista que contenga únicamente los valores comunes entre ambdas listas.

2.1.4 Ejercicio 4

Concatenar lo siguientes 3 diccionarios para crear uno nuevo:

dict1 = {1:10,2:20,3:30}

dict2 = {3:30, 4:40}

dict3 = {5:50}

2.1.5 Ejercicio 5

A partir del diccionario creado en el ejercicio 4, crear un iterador que barra los pares (*key*, *value*). Imprimir por pantalla ambos valores.

2.1.6 Ejercicio 6

Escribir un script que dados ds diccionarios, los unifique en uno sólo, obteniendo la suma de los valores para los *keys* repetidos.

dict1 = {'a':1, 'b':5, 'c': 10}

dict2 = {'a':2, 'c': 1, 'd':3}

2.1.7 Ejercicio 7

Implemenar un juego de *piedra papel o tijera* para dos jugarodres. Pedir a los usuarios en que ingresen su jugada. Las reglas son: - Piedra le gana a tijera - Papel le gana a piedra - Tijera le gana a papel - El juego termina cuando gana alguno de los dos jugadores

2.1.8 Ejercicio 8

Implementar el juego "ahorcado". Para ello el jugador 1 ingresa una palabra. Luego el jugador 2 debe ir adivinando las letras que lo componen. Escribir una función que maneje el desarrollo del juego, que tome como entrada la palabra elegida por el jugador 1. El juego termina cuando el jugador 2 acierta la palabra o usa una cantidad de intentos mayor a 15.

2.1.9 Ejercicio 9

El objetivo es crear una *aplicación* que permita manejar el presupuesto y gastos de una persona. Para ello vamos a usar un enfoque de programación orientada a objeto.

Crear un clase `Categoría`. Debería permitir instanciar distintas categorías del presupuesto, como alimentos, vestimenta, entretenimiento. La clase debe incluir un atributo `contabilidad` que sea de tipo lista. Además la clase deberá contener los siguientes métodos: - `deposito`: recibe un monto y una descripción. Si no se da ninguna descripción por defecto debe completarse con un string vacío. Además debe agregar a la lista `contabilidad` un objeto de la forma `{'cantidad': cantidad, 'descripcion':descripcion}` - `extraccion`: similar a depósito. Si no hay fondos suficientes, no se debe ejecutar la operación, y no debe agregarse nada a `contabilidad`. En caso de haber fondos suficientes, el objeto a agregar a `contabilidad` debe contemplar el monto con signo negativo (egreso). El método debería devolver `True` si se ejecutó la operación `False` si no. - `obtener_balance`: devuelve el balance actual de la categoría basado en los depósitos y extracciones que se hayan producido. - `transferencia`: acepta un monto y el nombre de otra categoría del presupuesto como argumentos. Este método transfiere plata de una categoría a la otra. Se debe represetar con una `extracción` y un `depósito` en las catergorías que corresponda, con la descripción "Transf. a Categoría destinp" y "Trans. de Categoría destino". Nuevamente si no alcanzan los fondos no se ejecuta la operación y devuelve `False`, en caso contrario devuelve `True`. - `verificar_fondos`: acepta como argumento un mnonto. Devuelve `True` si se dispone de fondos suficientes (si el balance de la categoría es mayor al monto ingresado), `False` en caso contrario. Debería ser utilizado por los metodos `transferencia` y `extracción`.

Cuando se imprime el objeto `presupuesto` debería mostrar:

Un titulo de 30 caracteres con el nombre de la categoría centrado entre *.

Una lista de los items en la contabilidad. Cada línea debe mosrar los primeros 20 caracteres de la descripción y luego el monto, con dos decimales.

Una línea mostrando el total de la categoría

Por ejemplo:

```
*************Alimentos*************
deposito inicial        1000.00
alimentos               -10.15
restaurant y mas com -15.89
Transf. a Vestimenta    -50.00
Total: 923.96
```

Además de la clase Categoría, crear una función llamada `crear_tabla_gastos` que tome como argumento una lista de categoría. Debe devolver un *string* que sea un diagrama de barras.

La gráfica debe mostrar el porcentage gastado en cada categoría que se le pasa a la función. Dicho porcentaje se calcula únicamente sobre las extracciones. Un ejemplo sería

```
Porcentaje usado por categoría
100|
 90|
 80|
 70|
 60| o
 50| o
 40| o
 30| o
 20| o  o
 10| o  o  o
  0| o  o  o
    ----------
     A  V  A
     l  e  u
     i  s  t
     m  t  o
     e  i
     n  m
     t  e
     o  n
     s  t
        a
```

[ ]: