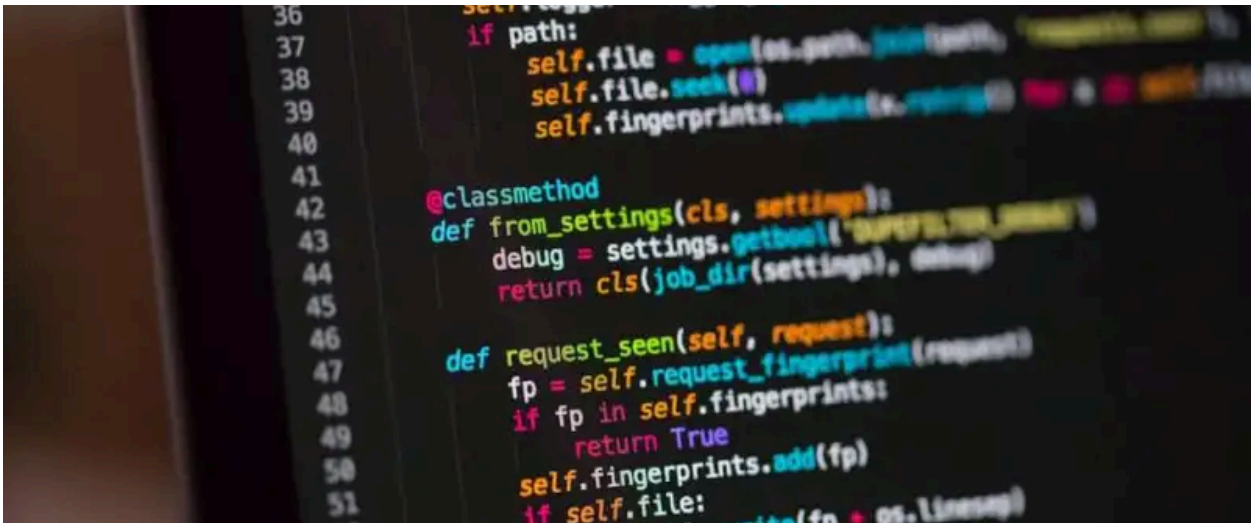

Separación de elementos usando Pila y Cola

Estructura

```
class Cola:
    def __init__(self):
        self.items = []
```

Se crea una lista llamada `items` para almacenar todos los valores que ingrese el usuario en la cola.

```
def encolar(self, elemento):
        self.items.append(elemento)
```

La función
`encolar` sirve para agregar un elemento al final de la cola., y `.append()`, agrega el nuevo dato al final de la lista.

```
def desencolar(self):
        if not self.esta_vacia():
            return self.items.pop(0)
```

Se verifica si la cola no esta vacía, y se usa `.pop(0)` para sacar el elemento **en la posición 0**, o sea, el primero que entró.

```
def esta_vacia(self):
        return len(self.items) == 0
```

Devuelve True si la cola esta vacía.

```
    def ver_elementos(self):
        return self.items.copy()
```

Devuelve una copia de la cola.

```
def tamaño(self):
    return len(self.items)
```

Cantidad de elementos ingresados.

```
class Pila:
    def __init__(self):
        self.items = []
```

Igual que en la cola, se crea una lista vacía items que guarda los elementos.
Pero ahora se comporta como una pila de platos: el último que entra es el primero que sale.

```
def apilar(self, elemento):
        self.items.append(elemento)
```

`apilar` agrega un nuevo elemento encima de la pila. Se agrega al final de la lista con `.append()`

```
def desapilar(self):
        if not self.esta_vacia():
            return self.items.pop()
```

Elimina un elemento.

```
def esta_vacia(self):
        return len(self.items) == 0
```

Longitud de la Pila.

```
def ver_elementos(self):
        return self.items.copy()
```

Métodos

```
import Estructura
```

Se importa Estructura para utilizar las funciones definidas dentro de este.

```
cola = Estructura.Cola()
pila = Estructura.Pila()
```

Se crea una instacia de cola y pila.

```
def verificar_par(cola, pila):

    tamaño_original = cola.tamaño()
    for i in range(tamaño_original):
        elemento = cola.desencolar()
```

Hacemos un bucle for que va desde 0 hasta el tamaño de la cola.
En cada vuelta del bucle, sacamos un elemento de la cola con desencolar().
Este elemento es lo que el usuario ingresó, y ahora lo vamos a clasificar.

```
if i % 2 == 0:
            cola.encolar(elemento)
        else:
            pila.apilar(elemento)
```

Si el número de vuelta (i) es impar (1, 3, 5...), entonces ese elemento se manda a la pila.
O sea, las posiciones impares van a la pila, y como la pila es LIFO, se van guardando al revés.

Main

```
from Estructura import Cola, Pila
from Metodos import procesar_cola
```

Se importa Estructura y métodos para utilizar la Cola, Pila y procesar_cola, previamente definidas.

```
def main():
    cola = Cola()
    pila = Pila()

    cadena = input("Ingrese una cadena de texto: ")
```

```
    for caracter in cadena:
        cola.encolar(caracter)
```

Se agrega el carácter.

```
    print("\nCola original:", cola.ver_elementos())
```

Se muestra la cola original, con todos los caracteres ingresados por el usuario, en el mismo orden en que se escribieron.

```
    procesar_cola(cola, pila)
```

Esta función separa los elementos en **pares e impares** según su posición:

- Si la posición es **par**, el elemento **regresa a la cola**.
- Si la posición es **impar**, el elemento **se guarda en la pila**.

```


    print("Cola final (posiciones pares):", cola.ver_elementos())
    print("Pila final (posiciones impares - orden LIFO):", pila.ver_elementos())
```

Muestra los elementos pares e impares.

```
if __name__ == "__main__":
    main()
```

Se asegura que main se ejecute.