📝 Apuntes de Programación: Validación y Manipulación de Cadenas (Strings) en Python

Las Cadenas de Caracteres (`str`) son secuencias inmutables de caracteres y son esenciales para manejar texto. Python ofrece varios métodos de validación para verificar el contenido de una cadena antes de usarla, lo cual es fundamental para el manejo de errores.

1. Métodos de Validación de Contenido

Estos métodos devuelven `True` si **todos** los caracteres de la cadena cumplen la condición, y `False` en caso contrario (o si la cadena está vacía).

A. Validación Numérica

Método	Descripción	Ejemplo
`str.isdigit()`	Verifica si todos los caracteres son **dígitos** (`0-9`).	`"33".isdigit()` → `True`
`str.isnumeric()`	Similar a `isdigit()` , pero también incluye caracteres numéricos específicos de Unicode (ej. fracciones).	`"²".isnumeric()` → `True`
`str.isdecimal()`	Similar a `isdigit()` , pero solo para decimales.	

Uso Clave (Tu Código): Se usa para saber si una cadena es segura para ser convertida a un entero.

```
texto = "33"
if texto.isdigit():
    print("Puedo convertirlo a entero") # Ejecuta si es seguro convertir
    entero = int(texto)
```

B. Validación Alfabética y Alfanumérica

Método	Descripción	Ejemplo
`str.isalpha()`	Verifica si todos los caracteres son **letras** (`a-z` , `A-Z`). Ignora espacios, números o signos de puntuación.	`"Hola".isalpha()` → `True`
`str.isalnum()`	Verifica si todos los caracteres son **letras o números** (Alfanuméricos). No permite espacios ni símbolos.	`"a33b".isalnum()` → `True`

Uso Clave (Tu Código): Se usa para verificar si una cadena contiene elementos no deseados, como números o símbolos.

```
texto = "33"
if texto.isalpha():
    print("No tiene numero")
else:
```

```
print("Si es numero o símbolo") # El "33" no es solo alfabético
```

C. Otras Validaciones Útiles

Método	Descripción
`str.isspace()`	¿Contiene solo caracteres de espacio en blanco? (Espacios, tabuladores, saltos de línea).
`str.islower()` / `str.isupper()`	¿Está la cadena en minúsculas / mayúsculas?
`str.istitle()`	¿Tiene formato de título? (Primera letra de cada palabra en mayúscula, el resto en minúscula).

2. Consejos para Mejorar la Programación con Strings

🎯 Principio 1: Usar F-Strings para Formatear

- La forma más moderna, legible y rápida de formatear texto es usando **F-Strings** (cadenas literales con formato). Esto es un estándar en el código Python actual.

```
nombre = "María"
edad = 25
# En lugar de "Mi nombre es {} y tengo {} años".format(nombre, edad)
documento = f"Mi nombre es {nombre} y tengo {edad} años."
print(documento)
```

🎯 Principio 2: Inmutabilidad

- **Regla de oro:** Las cadenas de Python son **inmutables**. Cuando aplicas un método (ej. `upper()`, `replace()`), no modificas la cadena original. En su lugar, Python **crea una nueva cadena** con los cambios.

```
nombre = "ana"
nombre.upper() # Crea "ANA", pero la ignora
print(nombre)  # Imprime "ana" (la original)

nombre_mayus = nombre.upper()
print(nombre_mayus) # Imprime "ANA" (la nueva)
```

🎯 Principio 3: Métodos de Limpieza y Transformación

- `str.strip()` **:** Elimina espacios en blanco (o caracteres específicos) al inicio y al final de la cadena.
- `str.lower()` **/** `str.upper()` **:** Convierte la cadena completa a minúsculas / mayúsculas.
- `str.replace(old, new)` **:** Reemplaza todas las ocurrencias de una subcadena por otra.

🎯 Principio 4: Evitar el Uso de `eval()`

- **Seguridad en Exámenes:** Aunque `eval()` puede ejecutar código Python contenido en una cadena, **NUNCA** lo uses con datos de entrada que no puedas controlar (ej. entrada de usuario).