Arquivo: brl.py

```
import locale
from PyQt6.QtWidgets import QLineEdit

# Define o locale para BRL
try:
    locale.setlocale(locale.LC_ALL, 'pt_BR.UTF-8')
except locale.Error:
    try:
        locale.setlocale(locale.LC_ALL, 'Portuguese_Brazil.1252')
    except locale.Error:
        print("Aviso: Locale 'pt_BR' não encontrado. A formatação pode estar incorreta.")

def formatar_para_brl(valor):
    """
    Converte um valor para o formato de moeda BRL (R$), tratando de forma inteligente
    números (floats) e strings em diferentes formatos.
    """
    if valor is None or str(valor).strip() == "":
        return "R$ 0,00"

    try:
        numeric_value = 0.0

        # --- INÍCIO DA NOVA LÓGICA ---

        # 1. Se o valor já é um número (como 62631.05 vindo da planilha),
        #    usa-o diretamente, sem fazer nenhuma limpeza de texto.
        if isinstance(valor, (int, float)):
            numeric_value = float(valor)

        # 2. Se for um texto (string), aplicamos uma lógica mais inteligente.
        elif isinstance(valor, str):
            cleaned_str = valor.replace('R$', '').strip()

            # Verifica se o formato é brasileiro (com vírgula decimal, ex: "62.631,05")
            if ',' in cleaned_str and '.' in cleaned_str:
                # Remove o ponto de milhar e troca a vírgula por ponto decimal
                numeric_value = float(cleaned_str.replace('.', '').replace(',', '.'))
            # Verifica se o formato já está como float (ex: "62631.05")
            # Este é o caso principal da sua planilha
            elif '.' in cleaned_str:
                # Remove vírgulas de milhar (caso existam) e converte
                numeric_value = float(cleaned_str.replace(',', ''))
            # Trata casos com apenas vírgula decimal (ex: "62631,05")
            elif ',' in cleaned_str:
                 numeric_value = float(cleaned_str.replace(',', '.'))
            # Se não houver separador, trata como número inteiro
            else:
                numeric_value = float(cleaned_str)
```

```
        # Se o tipo não for número nem string, tenta uma conversão direta
        else:
            numeric_value = float(valor)
        # Usa o locale para formatar o número final no padrão brasileiro
        return locale.currency(numeric_value, grouping=True, symbol=True)

    except (ValueError, TypeError):
        # Em caso de qualquer erro, retorna um valor seguro.
        return "R$ 0,00"


class CustomQLineEdit(QLineEdit):
    def __init__(self, valor_inicial, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.valor_inicial = valor_inicial  # Armazena o valor inicial como número
        self.setText(formatar_para_brl(valor_inicial))  # Exibe o valor inicial formatado
        self.textChanged.connect(self.validar_valor)  # Valida quando o texto é alterado

    def validar_valor(self):
        """Valida o valor digitado e converte para float."""
        texto_atual = self.text().replace('R$', '').replace('.', '').replace(',', '.').strip()
        try:
            float(texto_atual)  # Apenas valida se é um número
        except ValueError:
            self.setStyleSheet("border: 1px solid red;")  # Indica erro visualmente
        else:
            self.setStyleSheet("")  # Remove o erro visual

    def focusOutEvent(self, event):
            """Override do evento focusOut para aplicar formatação BRL apenas se o valor for
válido."""
        texto_atual = self.text().replace('R$', '').replace('.', '').replace(',', '.').strip()
        try:
            # Atualiza apenas se o valor for válido
            valor_numerico = float(texto_atual)
            self.setText(formatar_para_brl(valor_numerico))
        except ValueError:
            self.setText(formatar_para_brl(self.valor_inicial))  # Retorna ao valor inicial
        super().focusOutEvent(event)
```