Monetary / financial calculations inside your QRF → BOQ automation

For **monetary / financial calculations inside your QRF → BOQ automation**, you need two things:

1. **Exact decimal precision** (no binary-float rounding surprises).
2. **Convenience helpers** for currency formatting, FX, amortisation, etc.

---

1. Core precision → `decimal`

```
from decimal import Decimal, ROUND_HALF_UP


qty   = Decimal('12.50')     # kg
rate  = Decimal('78.35')     # $/kg
total = (qty * rate).quantize(Decimal('0.01'), ROUND_HALF_UP)
```

2. Money wrapper → `money` or `py-moneyed`

```
from money import Money


unit_price = Money('78.35', 'USD')
line_total = Money('12.50', 'USD') * unit_price
```

3. Excel / reporting → `openpyxl`, `xlsxwriter` (they both **preserve `Decimal` precision** when you write values).
4. Optional extras
   - `forex-python` – real-time FX rates for multi-currency bids.
   - `numpy-financial` – NPV, IRR, loan amortisation if you need financing tables.
   - `babel` – locale-aware currency formatting.

Quick recipe (fits your Python stack):

```
from decimal import Decimal
from openpyxl import Workbook
```

```
from money import Money

wb = Workbook()
ws = wb.active
ws.append(['Item', 'Qty (kg)', 'Rate ($)', 'Amount ($)'])

for item, qty, rate in [
        ('Column UC203', Decimal('253.4'), Decimal('1.08')),
        ('Beam UB305', Decimal('417.9'), Decimal('1.08'))]:
    amount = Money(qty * rate, 'USD')
    ws.append([item, float(qty), float(rate), str(amount)])

wb.save('BOQ_financial.xlsx')
```

All values stay exact (no rounding errors), and the worksheet shows standard $-formatting.

Revision #1
Created 2025-07-18 01:54:08 UTC by Admin
Updated 2025-07-18 01:55:50 UTC by Admin