# Design Document: Callable and Non-Callable Bond Analytics Scripts

This document serves as a comprehensive design guide for the development of Python scripts to calculate key risk and return metrics for callable and non-callable fixed rate bonds using the QuantLib library. It merges architectural, workflow, data structure, extensibility, and diagrammatic details, providing a blueprint for robust and maintainable implementation.

## 1. Overview

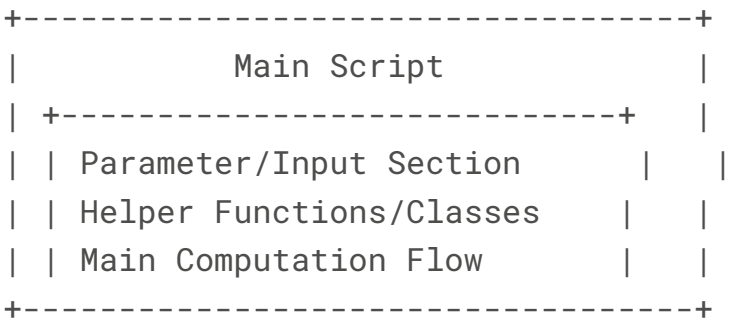
The codebase consists of two Python scripts—`callable_bond_metrics.py` and `non_callable_bond_metrics.py`—with a modular, object-oriented design to calculate Yield to Maturity (YTM), Yield to Worst (YTW), durations, and convexities for fixed-rate bonds. A shared utility module is recommended for common tasks to maximize code reuse and extensibility.

## 2. High-level Architecture

The system is organized in three primary layers for each script:

- Parameter Definition Section: User-specified bond and market parameters, currently hardcoded for clarity.
- Helper Functions/Classes: Encapsulate calculations for yield, durations, convexity, etc.
- Main Computation Flow: Orchestrates QuantLib object construction, metric calculations, and outputs results.
- Error Handling: Ensures safe failover and clarity in exceptional circumstances.

```
text
```



# 3. Module, Class, and File Design

## 3.1 Core Classes and Utilities

Class Diagram (UML Style):

```
text
```

```
+-------------------------------------------------------------+
|             BondMetricsCalculator (abstract)                |
+-------------------------------------------------------------+
| - evaluation_date: date                                     |
| - calendar: Calendar                                        |
| - business_convention: BusinessDayConvention                |
| - settlement_days: int                                      |
| - face_amount: float                                        |
| - coupon_rate: float                                        |
| - frequency: Frequency                                      |
| - issue_date: date                                          |
| - maturity_date: date                                       |
| - flat_market_rate: float                                   |
|-------------------------------------------------------------|
| + calculate_durations()                                     |
| + calculate_convexity()                                     |
| + calculate_effective_duration_convexity()                  |
+-------------------------------------------------------------+
           /                                   \
          /                                     \
         /                                       \
+--------------------------+
+----------------------------------+
| NonCallableBondCalculator |     | CallableBondCalculator
|
+--------------------------+
+----------------------------------+
| + calculate_ytm()         |     | - call_schedule: List[Tuple]
|
```

```
|                              |
+----------------------------------+
+----------------------------+     | + calculate_yields_to_call()
|
                                   | + get_ytw_and_metrics()
|

+----------------------------------+
```

## 3.2 Utility Functions Module

(Shared by both scripts for DRY code.)

```
text
```

```
+-------------------------------------------------+
|                BondMetricsUtils                 |
+-------------------------------------------------+
| + build_quantlib_schedule(...)                  |
| + build_discount_curve(...)                     |
| + attach_pricing_engine(...)                    |
| + calculate_metric_with_exception_safety(...)   |
+-------------------------------------------------+
```

## 3.3 File Structure

```
text
```

```
/project-root/
  callable_bond_metrics.py
  non_callable_bond_metrics.py
  bond_metrics_utils.py
  requirements.txt
  README.md
  docs/
    functional_requirements.md
```

`design_document.md`

# 4. Data Model & Core Components

## 4.1 Input Parameters

At the top of each script:

- Market: evaluation_date, calendar, business_convention, settlement_days, flat_market_rate
- Bond: face_amount, coupon_rate, frequency, issue_date, maturity_date
- Callable Only: call_schedule (dates and call prices)

Types are standard Python (dates, floats) and QuantLib enums/constants.

## 4.2 QuantLib Object Construction

- Schedule: QuantLib `Schedule` for cashflow timings.
- Bonds: `FixedRateBond` (non-callable) or `CallableFixedRateBond` (with `CallabilitySchedule`).
- Discount Curve: FlatForward built from specified rate.
- Pricing Engine: `DiscountingBondEngine` for present value calculations.

## 4.3 Calculation Helpers

Encapsulate metric calculation logic as standalone functions or methods, e.g.:

- `calculate_ytm(bond, price)`
- `calculate_duration(bond, yield, type)`
- `calculate_convexity(bond, yield)`
- `calculate_effective_duration_convexity(bond, yield, shift)`
- (Callable) `calculate_yields_to_call(bond, call_schedule, price)`, `get_ytw_and_metrics(...)`

# 5. Main Workflow Logic

1. Parameter Setup: Set QuantLib evaluation date and define market and bond parameters.
2. Schedule Construction: Create QuantLib `Schedule` for coupons.
3. Bond Construction: Instantiate appropriate bond object.

4. Discount Curve Creation: Use `FlatForward` for simplicity.
5. Engine Attachment: Assign pricing engine.
6. Metric Calculations:
    - Non-callable: Compute YTM, modified/Macaulay duration, convexity, effective duration/convexity.
    - Callable: For each call date, compute yield/duration/convexity, select YTW (lowest yield) and corresponding metrics.
7. Output: Print all metrics in a human-readable format, rounding decimals.

# 6. Error Handling & Edge Cases

- Yield Convergence: If yield calculations fail (e.g., price too high), output “N/A” for that scenario.
- Sanity Checks: Validate input coherence (e.g., issue date < maturity, call dates in range).
- Graceful Degradation: Incomplete metrics reported as “N/A”; computation continues without crash.
- Output Clarity: Clear labeling on any non-computable or exceptional result.

# 7. Output Format

Console output, e.g.:
`text`

```
Yield to Maturity:      0.049876
Modified Duration:      7.773221
Macaulay Duration:      8.136912
Convexity:              89.124902
Effective Duration:     7.763029
Effective Convexity:    89.056277
```

For callable bonds, corresponding “To Worst” labels are included.

# 8. Sequence Diagrams

## 8.1 Non-Callable Bond Sequence

`text`

```
User/Script
   |
   | 1. Provide input parameters
   v
NonCallableBondCalculator
   |
   | 2. Initialize QuantLib objects, attach pricing engine
   | 3. Calculate YTM, durations, convexities
   v
Print/Output
```

Expanded for YTM:

text

```
Main Script         NonCallableBondCalculator    QuantLib
    |                        |                      |
    |--input parameters----->|                      |
    |                        |--init bond---------->|
    |                        |   (FixedRateBond)    |
    |                        |<---------------------|
    |                        |--bondYield()-------->|
    |                        |<---------------------|
    |<---- YTM value --------|                      |
    | print YTM to user      |                      |
```

## 8.2 Callable Bond Sequence

text

```
User/Script
   |
   | 1. Provide input (incl. call schedule)
   v
CallableBondCalculator
   |
```

```
   | 2. Initialize CallableFixedRateBond, engine
   | 3. For each call date:
   |     - calculate yield, duration, convexity
   | 4. Select minimum yield (YTW) and associated metrics
   v
Print/Output
```

## 9. Extensibility & Maintainability

- Parameterization: All user inputs are centralized for adjustment.
- Modularity: Each computation is encapsulated for easy extension/addition and robust unit testing.
- Code Reuse: Shared logic across bond types centralized in a utility module.
- Future Expansion: CLI parameter input, additional bond types (floaters, step-ups), REST/GUI components as separate layers.

## 10. External Dependencies

- QuantLib-Python library
- Python 3.x

Specify in `requirements.txt` and document in `README.md`.

## 11. Potential Enhancements

- Config/CLI-driven parameter input
- Logging for debug and error handling
- Internationalization (i18n) of outputs
- Unit and integration tests
- Packaging as a library

## 12. Appendix: Example Function Signatures

python

```python
def calculate_ytm(bond: QuantLib.Bond, price: float) -> float:
    ...
```

```python
def calculate_duration(bond: QuantLib.Bond, ytm: float,
duration_type: QuantLib.Duration.Type) -> float:
    ...

def calculate_effective_duration_convexity(bond: QuantLib.Bond,
ytm: float, shift: float) -> Tuple[float, float]:
    ...
```

Callable Bond:
python

```python
def calculate_yields_to_call(bond:
QuantLib.CallableFixedRateBond, call_schedule: List[Tuple[date,
price]], clean_price: float) -> List[Tuple[date, float]]:
    ...
```