# 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

+-----------------------------------+

| Main Script |

| +-----------------------------+ |

| | Parameter/Input Section | |

| | Helper Functions/Classes | |

| | Main Computation Flow | |

+-----------------------------------+

## 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

def calculate\_ytm(bond: QuantLib.Bond, price: float) -> float:

...

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

def calculate\_yields\_to\_call(bond: QuantLib.CallableFixedRateBond, call\_schedule: List[Tuple[date, price]], clean\_price: float) -> List[Tuple[date, float]]:

...