# Multiparadigm Programming with Python (PyMPP) Chapter Index

**H. Conrad Cunningham**

**20 April 2022**

## Contents




Professor of Comp uter and Information Science
University of Mississippi
214 Weir Hall
P.O. Box 1848
University, MS 38677
(662) 915-7396 (dept. office)

**Browser Advisory:** The HTML version of this textbook requires a browser

that supports the display of MathML. A good choice as of April 2022 is a recent version of Firefox from Mozilla.

# Multiparadigm Programming with Python

## Chapters 5-9 in One Document

- ***Multiparadigm Programming with Python (PyMPP)***:
  - as HTML
  - as PDF

## Chapter 1: Future Chapter

TODO

## Chapter 2: Programming Paradigms (ELIFP)

- Chapter 2, Programming Paradigms, from *Exploring Languages with Interpreters and Functional Programming* (ELIFP):
  - as HTML
  - as PDF
- Slides: (HTML) Programming Paradigms

## Chapter 3: Object-Based Paradigms (ELIFP)

- Chapter 3, Object-Based Paradigms, from *Exploring Languages with Interpreters and Functional Programming* (ELIFP):
  - as HTML
  - as PDF
- Slides: (HTML) Object-Based Paradigms

## Chapter 4: Future Chapter

## Chapter 5: Python Types

- Chapter:
  - as HTML
  - as PDF
- Slides:
  - (HTML) Type System Concepts (from ELIFP, Ch. 5)
  - No slides yet for Python-specific parts of chapter

## Chapter 6: Python Program Components

- Chapter:
  - as HTML

  - as PDF
- Slides: NONE yet

### Chapter 7: Python Object Orientation

- Chapter:
  - as HTML
  - as PDF
- Slides: NONE yet

### Chapter 8: Python Metaprogramming

- Chapter:
  - as HTML
  - as PDF
- Slides: NONE yet

### Chapter 9: Python Decorators and Metaclasses

- Chapter:
  - as HTML
  - as PDF
- Slides: NONE yet

### Chapter 10: Future Chapter

TODO

### Chapter 11: Software Testing Concepts (ELIFP)

- Chapter 11, Software Testing Concepts, from *Exploring Languages with Interpreters and Functional Programming* (ELIFP):
  - as HTML
  - as PDF
- Slides: NONE yet

### Chapter 12: Future Chapter on Python Testing

TODO

I plan for this chapter to use Python Rational Arithmetic modules smilar to those below. Testing will be both manual and using PyTest.

- Develop new chapter similar to Chapter 12, Testing Haskell Programs, from *Exploring Languages with Interpreters and Functional Programming* (ELIFP):
  - as HTML

  - as PDF
- HTML Slides: None yet
- Python source code for Rational Arithmetic case study:
  - Module rational_arith (Rat class)
  - Module ration_rep (RatRep) — abstract base class for rational number representation
  - Module rational_core (RatRep subclass RatCore) — relatively prime integer tuple (i.e., compute GCD on creation)
  - Module rational_defer (RatRep subclass RatDefer) — integer tuple not relatively prime (i.e., defer computing GCD until value required)
  - Module rational_test (manual black box test script) tests both representations
  - Module test_rational_core (PyTest black box test script for Rat/RatCore) uses parameterized function tests
  - Module test_rational_defer (PyTest black box test script for Rat/RatDefer) uses parameterized function tests
  - Module test_rational (PyTest black box test script for Rat with both RatRep) also uses parameterized fixture

## Examples from SICP [1]

TODO: Given most of these examples use a functional programming approach, not many have been developed yet in Python.

- Chapter:
  - HTML
  - PDF
- HTML Slides: None yet

This document also examines implementations in Scala, Elixir, Lua, Racket Scheme, and Elm.

## CookieJar Abstract Data Type

We discuss Python implementations of a CookieJar abstract data type (ADT) in the following possible future chapter.

- Chapter:
  - as HTML
  - as PDF
- HTML Slides: None yet

This document also examines implementations of this ADT in Scala and Ruby.

## Fowler's Lair Domain Specific Language

We discuss several Python implementations of Fowler's internal (and one external) Lair Domain-Specific Languages (DSL) in the following *Notes on Domain-Specific Languages* index page.

- Chapter:
  - as HTML
  - as PDF
- HTML Slides: None yet

This document also examines implementations of these DSLs in Lua—as well as Fowler's original Ruby code.


## Acknowledgements

For my Spring 2018, Python-based offering of CSci 658 (Software Language Engineering), I drafted the three-chapter booklet *Python 3 Metaprogramming* [4]. This booklet was inspired by David Beazley's Python 3 Metaprogramming tutorial slides from PyCon'2013 [2]. (Beazley's tutorial draws on material from his and Brian K. Jones' book *Python Cookbook* [3].) I adapted and extended the material from Beazley's terse slides to answer questions I had as a person relatively new to Python—or that I expected my students might have.

In Fall 2018, I continued to develop this material for my Python-based offering of CSci 556 (Multiparadigm Programming). I divided the previous booklet into chapters 5-9 of *Multiparadigm Programming in Python 3*. The Spring 2018 work used Python 3.6, but the Fall 2018 work used some Python 3.7 features.

This new document sought to be compatible with the concepts, terminology, and approach of my in-progress textbook *Exploring Languages with Interpreters and Functional Programming* [5], in particular of Chapters 2, 3, 5, 6, 7, and 11.

I retired from the full-time faculty in May 2019. As one of my post-retirement projects, I am continuing work on possible textbooks based on the course materials I had developed during my three decades as a faculty member. In January 2022, I began refining the existing content, integrating separately developed matreials together, reformatting the documents, constructing a unified bibliography (e.g., using citeproc), and improving my build workflow and use of Pandoc. I adapted this index page from a portions of my Fall 2018 CSci 556 and Spring 2019 CSci 555 course notes pages.

For the 2022 version of this work, I changed most references to "Python 3" to just be "Python". Official support for Python 2 has now been dropped.

I maintain this chapter as text in Pandoc's dialect of Markdown using embedded LaTeX markup for the mathematical formulas and then translate the document to HTML, PDF, and other forms as needed.

## References


[1] Harold Abelson and Gerald Jockay Sussman. 1996. *Structure and interpretation of computer programs (SICP)* (Second ed.). MIT Press, Cambridge, Massachusetts, USA. Retrieved from https://mitpress.mit.edu/sicp/

[2] David Beazley. 2013. Python 3 metaprogramming (tutorial). Retrieved from http://www.dabeaz.com/py3meta/

[3] David Beazley and Brian K. Jones. 2013. *Python cookbook* (Third ed.). O'Reilly Media, Sebastopol, California, USA.

[4] H. Conrad Cunningham. 2018. Multiparadigm programming with Python 3. University of Mississippi, Department of Computer and Information Science, University, Mississippi, USA. Retrieved from https://john.cs.olemiss.edu/~hcc/csci556/Py3MPP/Ch05/05_Python_Types.html

[5] H. Conrad Cunningham. 2022. *Exploring programming languages with interpreters and functional programming (ELIFP)*. University of Mississippi, Department of Computer and Information Science, University, Mississippi, USA. Retrieved from https://john.cs.olemiss.edu/~hcc/docs/ELIFP/ELIFP.pdf