

8. Data Types

The modules described in this chapter provide a variety of specialized data types such as dates and times, fixed-type arrays, heap queues, synchronized queues, and sets.

Python also provides some built-in data types, in particular, `dict`, `list`, `set` and `frozenset`, and `tuple`. The `str` class is used to hold Unicode strings, and the `bytes` class is used to hold binary data.

The following modules are documented in this chapter:

- 8.1. `datetime` — Basic date and time types
 - 8.1.1. Available Types
 - 8.1.2. `timedelta` Objects
 - 8.1.3. `date` Objects
 - 8.1.4. `datetime` Objects
 - 8.1.5. `time` Objects
 - 8.1.6. `tzinfo` Objects
 - 8.1.7. `timezone` Objects
 - 8.1.8. `strftime()` and `strptime()` Behavior
- 8.2. `calendar` — General calendar-related functions
- 8.3. `collections` — Container datatypes
 - 8.3.1. `ChainMap` objects
 - 8.3.1.1. `ChainMap` Examples and Recipes
 - 8.3.2. `Counter` objects
 - 8.3.3. `deque` objects
 - 8.3.3.1. `deque` Recipes
 - 8.3.4. `defaultdict` objects
 - 8.3.4.1. `defaultdict` Examples
 - 8.3.5. `namedtuple()` Factory Function for Tuples with Named Fields
 - 8.3.6. `OrderedDict` objects
 - 8.3.6.1. `OrderedDict` Examples and Recipes
 - 8.3.7. `UserDict` objects
 - 8.3.8. `UserList` objects
 - 8.3.9. `UserString` objects
- 8.4. `collections.abc` — Abstract Base Classes for Containers
 - 8.4.1. Collections Abstract Base Classes
- 8.5. `heapq` — Heap queue algorithm
 - 8.5.1. Basic Examples
 - 8.5.2. Priority Queue Implementation Notes
 - 8.5.3. Theory
- 8.6. `bisect` — Array bisection algorithm
 - 8.6.1. Searching Sorted Lists

- 8.6.2. Other Examples
- 8.7. array — Efficient arrays of numeric values
- 8.8. weakref — Weak references
 - 8.8.1. Weak Reference Objects
 - 8.8.2. Example
 - 8.8.3. Finalizer Objects
 - 8.8.4. Comparing finalizers with `__del__()` methods
- 8.9. types — Dynamic type creation and names for built-in types
 - 8.9.1. Dynamic Type Creation
 - 8.9.2. Standard Interpreter Types
 - 8.9.3. Additional Utility Classes and Functions
 - 8.9.4. Coroutine Utility Functions
- 8.10. copy — Shallow and deep copy operations
- 8.11. pprint — Data pretty printer
 - 8.11.1. PrettyPrinter Objects
 - 8.11.2. Example
- 8.12. reprlib — Alternate `repr()` implementation
 - 8.12.1. Repr Objects
 - 8.12.2. Subclassing Repr Objects
- 8.13. enum — Support for enumerations
 - 8.13.1. Module Contents
 - 8.13.2. Creating an Enum
 - 8.13.3. Programmatic access to enumeration members and their attributes
 - 8.13.4. Duplicating enum members and values
 - 8.13.5. Ensuring unique enumeration values
 - 8.13.6. Using automatic values
 - 8.13.7. Iteration
 - 8.13.8. Comparisons
 - 8.13.9. Allowed members and attributes of enumerations
 - 8.13.10. Restricted subclassing of enumerations
 - 8.13.11. Pickling
 - 8.13.12. Functional API
 - 8.13.13. Derived Enumerations
 - 8.13.13.1. IntEnum
 - 8.13.13.2. IntFlag
 - 8.13.13.3. Flag
 - 8.13.13.4. Others
 - 8.13.14. Interesting examples
 - 8.13.14.1. Omitting values
 - 8.13.14.1.1. Using `auto`
 - 8.13.14.1.2. Using `object`
 - 8.13.14.1.3. Using a descriptive string
 - 8.13.14.1.4. Using a custom `__new__()`
 - 8.13.14.2. OrderedEnum

- 8.13.14.3. DuplicateFreeEnum
- 8.13.14.4. Planet
- 8.13.15. How are Enums different?
 - 8.13.15.1. Enum Classes
 - 8.13.15.2. Enum Members (aka instances)
 - 8.13.15.3. Finer Points
 - 8.13.15.3.1. Supported `__dunder__` names
 - 8.13.15.3.2. Supported `_sunder_` names
 - 8.13.15.3.3. Enum member type
 - 8.13.15.3.4. Boolean value of Enum classes and members
 - 8.13.15.3.5. Enum classes with methods
 - 8.13.15.3.6. Combining members of Flag