Built in Python Collections and Their Operations

Lists

a list is a sequence of zero or more Python objects, commonly called items. A list has a literal representation, which uses square

like strings, lists can be sliced and concatenated with the standard operators → results are lists

lists are mutable, therefore

① lists are new when operated on by slice and concat creates, not pieces of the origin list.

Tuple

tuple is an immutable sequence of items

tuple is essentially like a list without mutator methods

dictionaries

key value pairs, can sort them, we know [illegible]

Pattern Matching with Collections

ability to access several items at once by means of pattern matching

similar to a,b = function output

Note: Function

when a function does not include a return statement, it automatically returns the value None

you can define functions in any order in a module
as long as the function is not executed before completion.

Recursive Function

recursive function calls itself. To prevent a function from repeating itself indefinitely, it must contain a base case.

Python Notes

input function → standard function input waits for the user to enter text at the keyboard. 1

Using of __name__ == "__main__"

the purpose of this if statement is to allow the programmer either to run the module as a standalone program or to import it from the shell or another program.

- every Python module includes a set of built-in module variables, to which the Python virtual machine automatically assigns values when the module is loaded.
- If the module is being loaded as a standalone program (either by running it from a terminal prompt or by loading it from an idle window) - the module's __name__ variable is set to the string "__main__".

Otherwise it is set to the module name.

Strings and Their operations

as in other languages, a python string is a compound object that includes other objects, its characters.
However each character in a python string is itself a single character string

operators

Note. can use negative indexing. when an index is negative, Python adds this value to the string's length to locate the characters to be returned.

* strings are immutable → once you create, you cannot modify their internal contents

objects and method calls

methods operate on objects (functions are free form)
* if you try to run a method that an object doesn't recognize, Python raises an exception
use dir( ) to get the methods associated with a class
use(object.method) help to get a documentation

$1 - 1/3 + 1/5 - 1/7$

this is a summation
number of items just adds +2 to the denominator

my math skills are rusty

so you do a sum $\left[\frac{1}{n+2} \right.$

sum $\left[\frac{1^n}{n+2} \right]$

$1^0 = \frac{1}{2n+1}$

wrote the stats.py module
If we're writing a program that will put the line of text into a list.

$\boxed{\text{Chapter 2 : An overview of collections}}$

there are 4 general categories of collections
- linear
- hierarchical
- graph
- unordered

a collection: as the name implies, is a group of zero or more items that can be treated as a conceptual unit.
several in built python collections: string, list, tuple, set, dictionary

Linear collections

items in a linear collections are ordered by position. Each item except the first has has a unique predecessor, and each item except the last has a unique successor.

$$\begin{tikzcd} \boxed{D_1} \arrow[r, dash] & \boxed{e_2} \arrow[r, dash] & \boxed{D_3} \arrow[r, dash] & \boxed{D_4} \arrow[r, dash] & \boxed{D_5} \end{tikzcd}$$

Hierarchical collection

data items in a hierarchical collection are ordered in a structure resembling an upside down tree. Each data item except the one at the top has just one predecessor, called its parent, but potentially many successors, called children.

Higher-Order Functions

python functions are first-class data objects. This means you can assign them to variables, save them in data structures, pass them as arguments to other functions, and return them as the values of other functions. 3

A higher order function is a function that receives another function as an argument and applies it in some way

python includes two built-in higher order functions: map and filter. that are useful for processing iterable objects.

Map

this function expects a function and an iterable object as arguments and returns another iterable object wherein the argument function is applied to each item contained in iterable object.

Filter

filter expects a boolean function and an iterable object as arguments. the filter function returns an iterable object in which the items that are true in the boolean are returned in the resultant iterable object

Creating anonymous function with lambda

programmers can avoid defining one-line helper functions to pass to map/filter, on the fly.

lambda<argument list>: <expression>

Reading and Writing Objects with pickle

module for efficiently writing and storing data locally

Note: adopt a policy of writing individual items in a collection to a file and re-creating the collection from file inputs

Data ~~as~~ Classes

all data types in python are classes

Collection
- Graph Collection
- Hierarchical Collection
  - Binary Search Trees
  - Heap
- Linear Collection
  - List
    - Sorted List.
  - Queue
    - Priority Queue
  - Stack
  - String
- Unordered collection
  - Bag
    - Sorted Bag
  - Dictionary
    - Sorted Dictionary
  - Set
    - Sorted Set

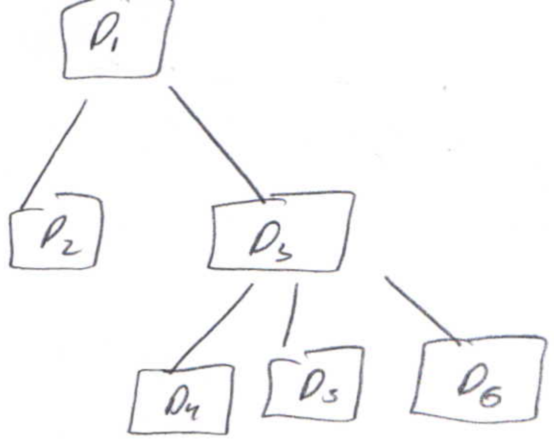
Note that a type name in this taxonomy doesn't imply a particular implementation of a collection

Operation on a collection

The manipulations that you can perform on a collection vary with the type of collection being used, but generally, the operations fall into several broad categories

Fundamental Operations on All Collection Types

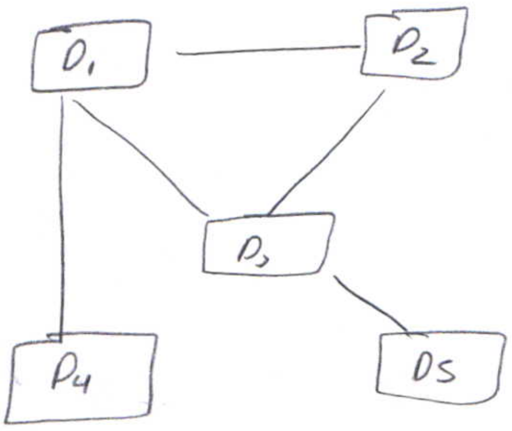
1) Determine the size: use python's len function to obtain the # of items currently in a collection
2) Test for item membership: use Python's in operator to search for a given target item in the collection. Return True if the item is found, or False otherwise
3) Traverse the collection: use python's for loop to visit each item in the collection. The order for which the items are visited depends on the collection.
4) Obtain a string representation: use Python's str function to obtain the string representation of the collection.
5) Test for equality: use Python's == operator to determine whether two collections are equal. Two collections are equal if they are of the same type and contain the same items. The order in which pairs of items are compared depends on the collection.



a file directory system, a company organ tree, and a book's table of contents can be thought of as hierarchical collections. 5

Graph Collections

type of collection in which each data item can have many predecessors and many successors.



the diagram shows, all elements connected to D3 are considered to be both its predecessors and its successors, and they are also called its neighbors

examples of graphs are maps of airline routes between cities, electrical

Unordered Collections

items in an unordered collection are not in any order, and its not possible to meaningfully speak of

Sorted Collections

a sorted collection imposes a natural ordering on its items.
To impose a natural ordering, there must be some rule for comparing items such that item$_i$ <= item$_{i+1}$, for the items visited in a sorted collection.

* sorted collection need not be linear or ordered by position.. A sorted collection allows the client to visit all of the items in a sorted order

when you do this cloning.

list2 = list(list1)

not only do the 2 lists have the same structure, but they share the same items. That is, the list function makes a shallow copy of its argument list. These items are not themselves cloned before being added to the new list; instead, mere references to these objects are copied.

↓

this is not an issue when the items are immutable within the list. however, when the collections are sharing mutable items, this can cause side effects.

↓

this is when creating a deep copy is necessary, which can also be done by writing a for loop over the source collection, which explicitly clones its items before adding them to a new collection.

Iterators and Higher Order Functions

each type of collection supports an iterator or for loop, an operation that iterates over the collection's items.

↓

for loop serves as as base for → due to this, these functions are usable across all collection types

- sum
- max
- min
- map
- filter
- reduce
- etc.

collections also known as abstract data type

python usually focuses on only one implementation of each of the available collection types.

→ languages like Java however, provide several.

goal of the book is partially to build your own collection implementations and references.

6) Concatenate the two collections: use python's + operator to obtain a new collection of the same type as the operands, and containing the items of both operands.

7) convert to another type of collection: create a new collection with the same items as a source collection. Cloning is a special case of type conversion, where two collections are of the same type.

8) Insert an item: add an item to a collection, possibly at a given position.

9) Remove an item. Remove an item from a collection, possibly at a given position.

10) Replace an item: combine removal and insertion into one operation

11) Access or retrieve an item: obtain an item, possibly at a given position.

Note: there is no single name for the insertion, removal, replacement or access operations in Python.

some common variation.

pop to remove from list (has for dict) at certain index
remove to remove

Type Conversion

you can convert one type of collection to another type of collection in a similar manner as type casting.

Note: some conversions like dict for dictionaries, expect more specific types of iterable object as arguments, like a list of tuples.

Cloning and Equality

a special type of type conversion is cloning, which returns an exact copy of the argument to the conversion function.

This should be the case when the argument's type is the same as the conversion function.