DATA COLLECTION STRUCTURES

CHAPTER 3

LISTS

- A list is a sequence of values of any data type that can be accessed forward or backward.
- Each value is called an element or a list item.
- Lists are mutable, which means that you won't create a new list when you modify a list element.
- Elements are stored in the given order.
- Various operations can be conducted on lists such as insertion, sort, and deletion.

LISTS - ACTIVITIES

- Creating Lists
- Accessing Values in Lists
- Adding and Updating Lists
- Deleting List Elements
- Basic List Operations
- Indexing, Slicing, and Matrices

BUILT-IN LIST FUNCTIONS AND METHODS

Sr.No.	Function	Description
1	`cmp(list1, list2)`	Compares elements of both lists
2	`len(list1)`	Gives the total length of the list
3	`max(list1)`	Returns an item from the list with max value
4	`min(list1)`	Returns an item from the list with min value
5	`list(seq)`	Converts a tuple into list

Sr.No.	Methods	Description
1	`list.append(obj)`	Appends object `obj` to the list
2	`list.count(obj)`	Returns count of how many times `obj` occurs in the list
3	`list.extend(seq)`	Appends the contents of `seq` to the list
4	`list.index(obj)`	Returns the lowest index in the list that `obj` appears in
5	`list.insert(index, obj)`	Inserts object `obj` into the list at offset index
6	`list.pop(obj=list[-1])`	Removes and returns last object or `obj` from list
7	`list.remove(obj)`	Removes object `obj` from list
8	`list.reverse()`	Reverses objects of list in place
9	`list.sort([func])`	Sorts objects of list; use compare `func` if given

DICTIONARIES

- A dictionary is an unordered set of key-value pair; each key is separated from its value by a colon (:).
- The items (the pair) are separated by commas, and the whole thing is enclosed in curly braces ({}).
- Dictionary keys should be unique and should be of an immutable data type such as string, integer, etc.
- Dictionary values can be repeated many times, and the values can be of any data type.
- It's a mapping between keys and values; you can create a dictionary using the dict() method.

DICTIONARIES - ACTIVITIES

- Creating Dictionaries
- Updating and Accessing Values in Dictionaries
- Deleting Dictionary Elements

BUILT-IN DICTIONARY FUNCTIONS

No	Function	Description
1	`cmp(dict1, dict2)`	Compares elements of two dictionaries.
2	`len(dict)`	Gives the total length of the dictionary, i.e., the number of items in the dictionary.
3	`str(dict)`	Produces a printable string representation of a dictionary.
4	`type(variable)`	Returns the type of the passed variable. If the passed variable is a dictionary, then it would return a dictionary type.

No	Methods	Description
1	`dict1.clear()`	Removes all elements of dictionary `dict1`
2	`dict1.copy()`	Returns a copy of dictionary `dict1`
3	`dict1.fromkeys()`	Creates a new dictionary with keys from `seq` and values
4	`dict1.get(key, default=None)`	For the key name `key`, returns the value or default if key not in dictionary
5	`dict1.has_key(key)`	Returns true if key is in dictionary `dict1`, false otherwise
6	`dict1.items()`	Returns a list of `dict1`'s (key, value) tuple pairs
7	`dict1.keys()`	Returns list of the dictionary `dict1`'s keys
8	`dict1.setdefault(key, default=None)`	Similar to `get()`, but will set `dict1[key]=default` if key is not already in `dict1`
9	dict1.update(dict2)	Adds dictionary `dict2`'s key-values pairs to `dict1`
10	dict1.values()	Returns list of dictionary `dict1`'s values

TUPLES

- A tuple is a sequence just like a list of immutable objects.
- The differences between tuples and lists are that the tuples cannot be altered; also, tuples use parentheses, whereas lists use square brackets.

TUPLES - ACTIVITIES

- Creating Tuples
- Concatenating Tuples
- Accessing Values in Tuples

BASIC TUPLES OPERATIONS

Expression	Results	Description
`len((5, 7, 2,6))`	`4`	Length
`(1, 2, 3,10) + (4, 5, 6,7)`	`(1, 2, 3,10, 4, 5, 6,7)`	Concatenation
`('Hi!',) * 4`	`('Hi!', 'Hi!', 'Hi!', 'Hi!')`	Repetition
`10 in (10, 2, 3)`	`True`	Membership
`for x in (10, 1, 5): print x,`	`10 1 5`	Iteration

SERIES

- A series is defined as a one-dimensional labeled array capable of holding any data type (integers, strings, floating-point numbers, Python objects, etc.).

```
SeriesX = pd.Series(data, index=index),
```

SERIES - ACTIVITIES

- Creating a Series with index
- Creating a Series from a Dictionary
- Creating a Series from a Scalar Value
- Vectorized Operations and Label Alignment with Series
- Name Attribute

DATA FRAMES

- A data frame is a two-dimensional tabular labeled data structure with columns of potentially different types.
- A data frame can be created from numerous data collections such as the following:
  - A 1D ndarray, list, dict, or series
  - 2D Numpy ndarray
  - Structured or record ndarray
  - A series
  - Another data frame

DATA FRAMES - ACTIVITIES

- Creating Data Frames from a Dict of Series or Dicts
- Creating Data Frames from a Dict of Ndarrays/Lists
- Creating Data Frames from a Structured or Record Array
- Creating Data Frames from a List of Dicts
- Creating Data Frames from a Dict of Tuples
- Selecting, Adding, and Deleting Data Frame Columns
- Assigning New Columns in Method Chains
- Indexing and Selecting Data Frames
- Transposing a Data Frame
- Data Frame Interoperability with Numpy Functions

PANELS

- A panel is a container for three-dimensional data; it's somewhat less frequently used by Python programmers.
- A panel creation has three main attributes.
  - items: axis 0; each item corresponds to a data frame contained inside
  - major_axis: axis 1; it is the index (rows) of each of the data frames
  - minor_axis: axis 2; it is the columns of each of the data frames

PANELS - ACTIVITIES

- Creating a Panel from a 3D Ndarray
- Creating a Panel from a Dict of Data Frame Objects
- Selecting, Adding, and Deleting Items

SUMMARY

- How to maintain a collection of data in different forms.
- How to create lists and how to manipulate list content.
- What a dictionary is and the purpose of creating a dictionary as a data container.
- How to create tuples and what the difference is between tuple data structure and dictionary structure, as well as the basic tuple operations.
- How to create a series from other data collection forms.
- How to create data frames from different data collection structures and from another data frame.
- How to create a panel as a 3D data collection from a series or data frame.