DATA STRUCTURES CHEAT SHEET

Python - Data Structure

Data Types

It is a way of organizing data that contains the items stored and their relationship to each other

The areas in which **Data Structures** are applied:

- Compiler design
- Operating system
- Database Management System
- Statistical Analysis Package
- Numerical Analysis
- Graphics
- Artificial Intelligence
- Simulations

Data structures can be used in the following areas:

- RDBMS: Array (Array of structure)
- Network data model: Graph
- Hierarchical Data model: Trees

Lists and Tuples in Python

Ordered sequence of values indexed by integer numbers. Tuples are immutable

- To initialize empty list /tuple:
  Syntax: Lists: myList = []
  Tuples: myTuple = ()
- To get an element in position x in list/tuple:
  Syntax: "x" in myListOrTuple
- Index of element 'X' of list/tuple
  Syntax: myListOrTuple.index("x") -- If not found, throws a ValueError exception
- Number of occurance of X in list/tuple:
  Syntax: myListOrTuple.count("x")
- Update an item of List/tuple:
  Syntax: Lists: myList[x] = "x“
  Tuples: tuples are immutable!
- Remove element in position X of list/tuple:
  Syntax: Lists: del myList[x]
  Tuples: tuples are immutable!
- To specify size of tuple/list:
  Synatx: len(myListOrTuple)
- Remove element in position X of list/tuple:
  Syntax: Lists: del myList[x]
  Tuples: tuples are immutable!
- Concatenate two lists/tuples:
  Lists: myList1 + myList2
  Tuples: myTuple1 + myTuple2
  Concatenating a List and a Tuple will produce a TypeError exception
- Insert element in position x of a list/tuple
  Syntax: Lists: myList.insert(x, "value")
  Tuples: tuples are immutable!
- Append "x" to a list/tuple:
  Syntax: Lists: myList.append("x")
  Tuples: tuples are immutable!
- Convert a list/tuple to tuple/list:
  Syntax: List to Tuple: tuple(myList)
  Tuple to List: list(myTuple)

Algorithm	Best case	Average case	Worst case	Remarks
Selection sort	½ n^2	½ n^2	½ n^2	n exchanges, quadratic is the best case
Insertion sort	n	¼ n^2	½ n^2	Used for small or partial-sorted arrays
Bubble sort	n	½ n^2	½ n^2	Rarely useful, Insertion sort can be used instead
Shell sort	n $\log_3$ n	unknown	c $n^{3/2}$	Tight code, Sub quadratic
Merge sort	½ n lg n	n lg n	n lg n	n log n guarantee; stable
Quick sort	n lg n	2 n ln n	½ n^2	n log n probabilistic guarantee; fastest in practice
Heap sort	$n^{\dagger}$	2 n lg n	2 n lg n	n log n guarantee; in place

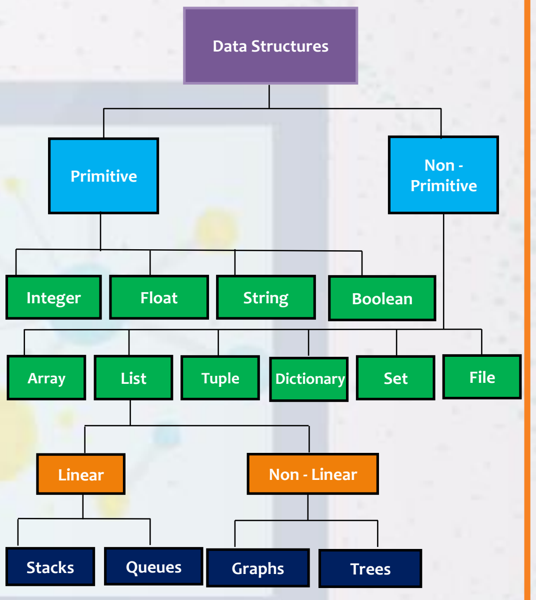
	Worst Case			Average Case		
Data Structure	Search	Insert	Delete	Search	Insert	Delete
Sequential search	n	n	n	n	n	n
Binary search	log n	n	n	log n	n	n
Binary search tree	n	n	n	log n	log n	sqrt(n)
Red-black BST	log n	log n	log n	log n	log n	log n
Hash table	n	n	n	$1^{\dagger}$	$1^{\dagger}$	$1^{\dagger}$

$1^{\dagger}$ - Uniform hashing assumption

Dictionaries

It is an unordered set of key value pairs

- Initialize an empty Dict
  Syntax: myDict = {}
- Add an element with key "k" to the Dict
  Syntax: myDict["k"] = value
- Update the element with key "k"
  Syntax: myDict["k"] = newValue
- Get element with key "k"
  Syntax: myDict["k"] -- If the key is not present, a KeyError is raised
- Check if the dictionary has key "k"
  Syntax: "k" in myDict
- Get the list of keys
  Syntax: myDict.keys()
- Get the size of the dictionary
  Syntax: len(myDict)
- Delete element with key "k" from the dictionary
  Syntax: del myDict["k"]
- Delete all the elements in the dictionary
  Syntax: myDict.clear()



Types of Data Structures

Primitive Data Structures:

- Integer: It is used to represent numeric data, more specifically whole numbers from negative infinity to infinity. Eg: 4, 5, -1 etc
- Float: It stands for floating point number. Eg: 1.1,2.3,9.3 etc
- String: It is a collection of Alphabets, words or other characters. In python it can be created by using a pair of single or double quotes for the sequence.

  Eg: x = 'Cake'

  y = ''Cookie''

Certain operations can be performed on a string:

- We can use * to repeat the string for a specific number of times. Eg: x*2
- String can be sliced, that is to select parts of the string. Eg: Coke

  ```
  z1 = x[2:]
  print(z1)
  # Slicing
  z2 = y[0] + y[1]
  print(z2)
  ```

  Output: ke
  Co
- To capitalize the strings
  Eg: str.capitalize('cookie')
- To retrieve the length of the strings
  Eg:

  ```
  str1 = "Cake 4 U"
  str2 = "404"
  len(str1)
  ```
- To replace parts of a string with another string
  - Eg: str1.replace('4 U', str2)

- Boolean: It is a built-in data type that can take the values TRUE or FALSE

Non- Primitive Data Structures:

- Array: It is a compact way of collecting data types where all entries must be of the same data type.

  Syntax of writing an array in python:

  ```
  import array as arr
  a = arr.array("I",[3,6,9])
  type(a)
  ```
- Linked list: List in Python is used to store collection of heterogeneous items. It is described using the square brackets [] and hold elements separated by comma

  Eg: x = [] # Empty list
  type(x)
  - The list can be classified into linear and non-linear data structures
  - Linear data structures contain Stacks and queues
  - Non-linear data structures contains Graphs and Trees
- Stack: It is a container of objects that can be inserted or removed according to LIFO(Last In First Out) concept. pop() method is used during disposal in Python

  Eg: stack.pop() # Bottom -> 1 -> 2 -> 3 -> 4 -> 5 (Top)
  stack.pop() # Bottom -> 1 -> 2 -> 3 -> 4 (Top)
  print(stack)
- Queue: It is a container of objects that can be inserted or removed according to **FIFO(First In First Out)** concept.
- Graph: It is a data structure that consists of a finite set of vertices called nodes, and a finite set of ordered pair (u,v) called **edges**. It can be classified as direction and weight
- Binary Tree: Tree is a **hierarchical data structure.** Here each node has at most two children
- Binary Search Tree: It provides moderate access/ search and moderate insertion/ deletion
- Heap: It is a complete tree and is suitable to be stored in an array, It is either MIN or Max
- Hashing: Collection of items that are stored in a way that it becomes easy to find them is **hashing**

Sets

It is an unordered collection with no duplicate elements. It supports mathematical operations like union, intersection, difference and symmetric difference.

- To initialize an empty set:
  Syntax: mySet = set()
- Initialize a non empty set
  Syntax: mySet = set(element1, element2...)
- To add element X to the set
  Syntax: mySet.add("x")
- Remove element "x" from a set:
  Syntax:
  Method 1: mySet.remove("x") -- If "x" is not present, raises a KeyErorr
  Method 2: mySet.discard("x") -- Removes the element, if present
- Remove every element from the set
  Syntax: mySet.clear()
- Check if "x" is in the set
  Syntax: "x" in mySet
- Size of the sets:
  Syntax: len(mySet)
- Union of two sets
  Syntax:
  Method 1: mySet1.union(mySet2)
  Method 2: mySet1 | mySet2
- Intersection of two sets
  Syntax:
  Method 1: mySet1.intersect(mySet2)
  Method 2: mySet1 & mySet2
- Difference of two sets
  Syntax:
  Method 1: mySet1.difference(mySet2)
  Method 2: mySet1 - mySet2
- Symmetric difference of two sets
  Syntax:
  Method 1: mySet1.symmetric_difference(mySet2)
  Method 2: mySet1 ^ mySet2