Day 2

Data Structures

1. Way of easily accessing, updating, storing data.
2. **Types:**
    a. Primitive Data Structures
        i. *List*
            1. Represented by square brackets “[ ]”
            2. Used maximum time.
            3. Dynamic in nature (grows according to data)
            4. Indexed, can manage duplicates
            5. Heterogeneous (can contain multiple data types)
            6. Can have deep and shallow copies:
                a. In deep copies, the list is actually copied to the other variable, this can be done by `list.copy()` method
                b. In shallow copy changes the actual list, it is just an another variable pointing to the original mem. reference, any changes in shallow copy will be reflected into new one.
            7. **METHODS**:
                a. `append`: adds item at the end of the list
                b. `extend`: extends the list by appending the other items (list) with the previous one
                c. `insert`: inserts the element at the given position without overwriting the existing element at that position
                d. `remove` : removes the first item, if not present throws error.

e. `pop`: Removes last item from list by default, else takes index and removes item.

f. `clear`: Removes all items in the list

g. `index`:

h. `count`:

i. `sort: (vs sorted(<list>)):` `sorted` is a method which takes a list and **RETURNS** the sorted list, while, `sort` is a list method which doesn't return the list but sorts the values in it, can be accessed by storing it in a variable.

j. `reverse`:

ii. *Set*

1. Represented by curly brackets "{ }"
2. Uniqueness, eliminates duplicate values.
3. Provides no index access, iteration is possible.
4. Heterogeneous
5. Use `set()` to create an empty set
6. Supports:
   a. intersection: `&`
   b. union: `|` pipe symbol
   c. Symmetric Difference: `^`
   d. Difference: `-`
7. METHODS:
   a. `add` : To add item into set
   b. `update` : add set
   c. `remove` : removes data
   d. `discard` : removes data without error

iii. *Tuple*

1. Represented by round brackets “( )”
2. Used by python for security since once entered data cannot be changed.
3. Static in nature, can be called as static list, immutable.
4. METHODS:
    a. `count`:
    b. `index`:

iv. *Dictionary*

1. Represented by curly brackets, takes key value pair, “{key;value}”.
2. Used for mapping
3. Heterogeneous
4. Keys needs to be unique, value can be duplicate.
5. Key based access.
6. METHODS:
    a. `dict.keys()`: will return a list of keys
    b. `dict.values()` : will return all values in the dict
    c. `dict.items()`: displays a list of tuple in key value pair

b. <u>In general (Language independent)</u>

i. Stack

ii. Queue

iii. Linked-list

iv. Trees

v. Graph