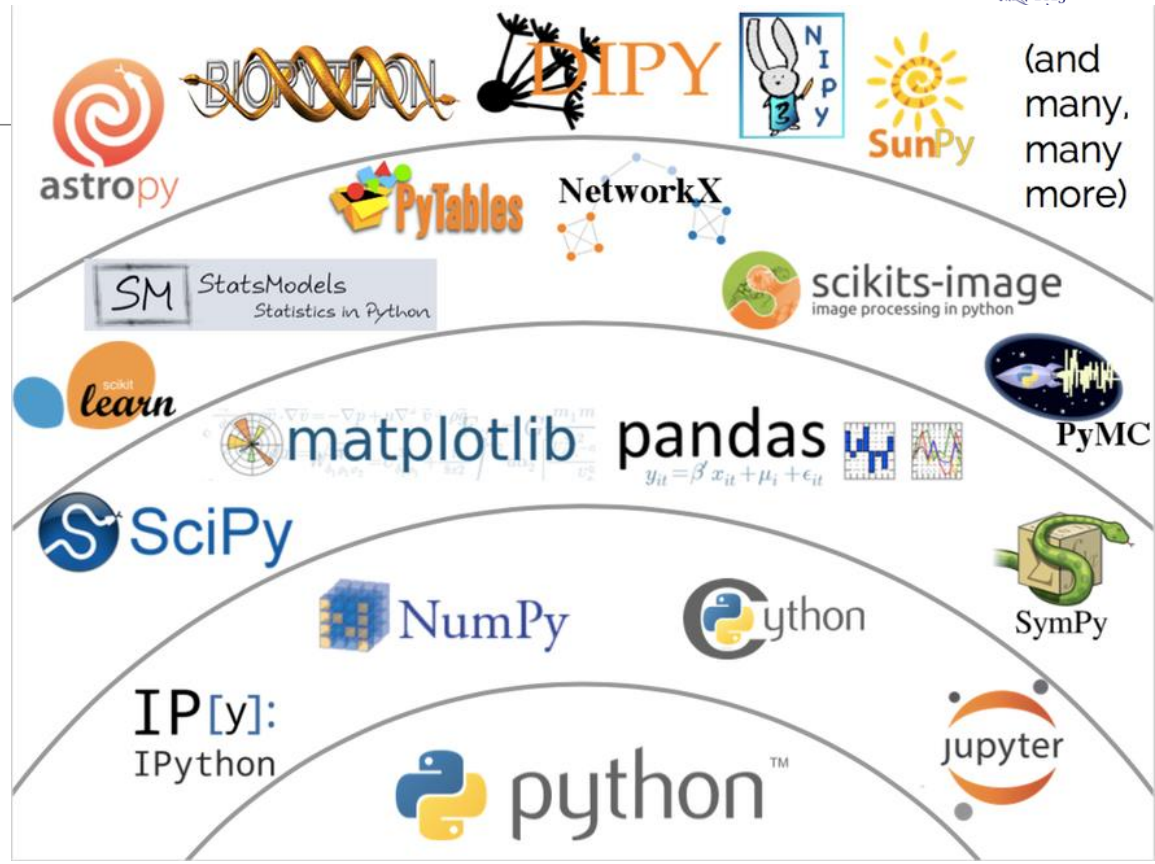
Machine Learning for *Computer Vision*

Dr. Sarwan Singh
Joint Director, NIELIT Chandigarh

Agenda

- About NIELIT
- Python –Programming constructs
- Data structures
- Built-in vs User Defined



Websites : geeksforgeeks.com, java2blog.com, tutorialpoints.com, docs.python.org

About NIELIT

- NIELIT – National Institute of Electronics and Information Technology
  - is a Capacity Building arm of the Ministry of Electronics and Information Technology (MeitY), Government of India
  - offers courses in IECT in the Formal as well as the Non-Formal Sectors of Education
  - is playing a key role in Digital India, Make in India & Skill India initiatives undertaken by Govt. of India
- Pan India presence – 47 NIELIT Centres
- Network of over 1100+ private accredited Centres and
- About 8500+ Facilitation Centres actively engaged in Digital Literacy.
- NIELIT – National Examination body
  - vast experience in conducting Online examinations and third party assessments.

about NIELIT Chandigarh

erstwhile

RCC

1978

Regional Computer Center (RCC)

DOEACC QUALITY ASSURANCE IN COMPUTER EDUCATION

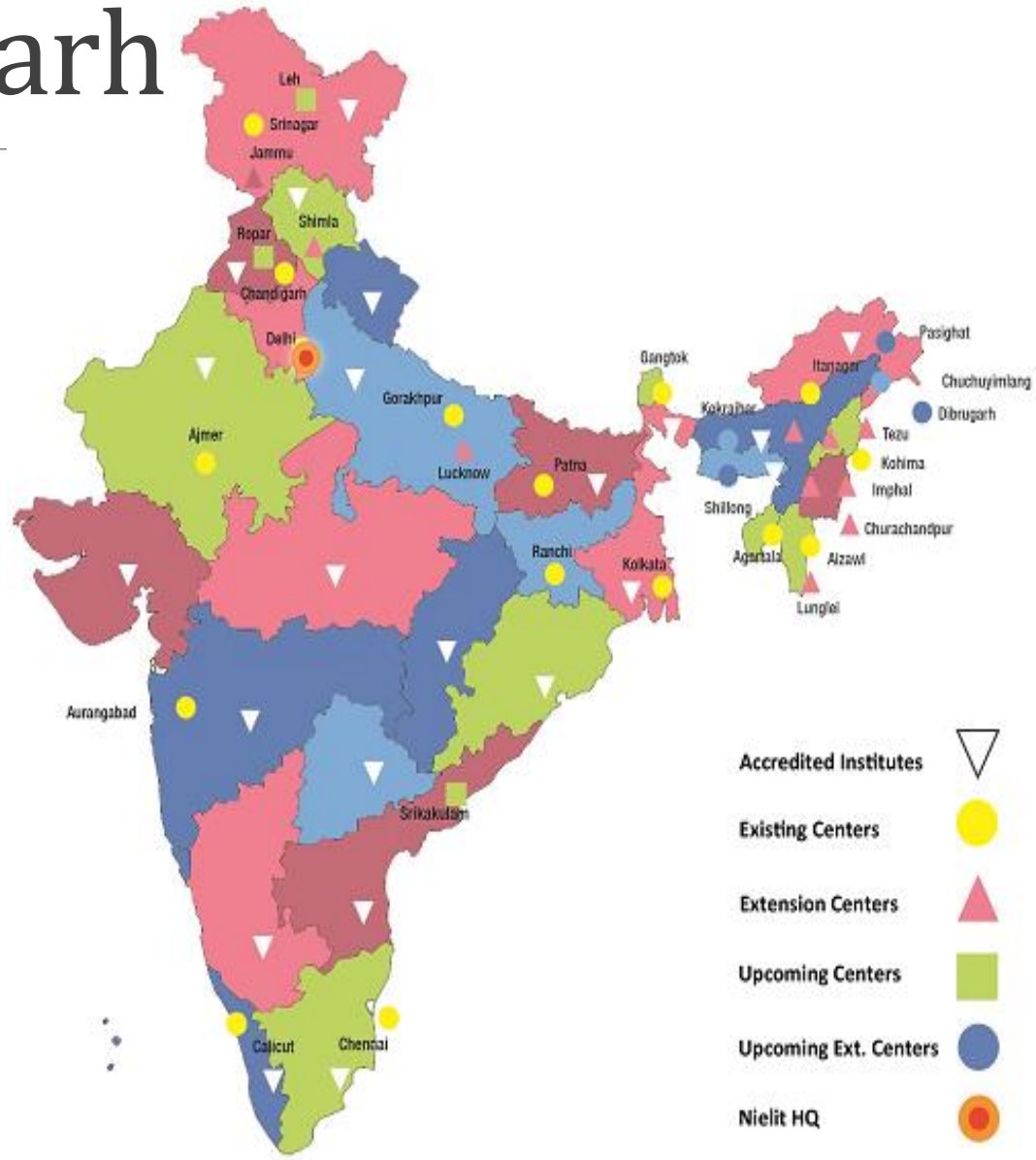
डी ओ ई ए सी सी

2003

DOEACC

रा.इ.सू.प्रौ.सं NIELIT

2012 onwards



NIELIT Chandigarh

- Education & Training
- Projects
- Software Development
- NCPUL
- Data Processing
- Examination Processing
- Facility Management
- Online Services

Training & Projects

- ESDM Scheme of Govt. of India
- Corporate trainings to various Govt. departments
- Training under DGE&T scheme
- Training employment exchange officers National Career Service Portal
- Training to Panchayati Raj Functionaries
- Training for more than 500 CSC VLEs
- Data Digitization of NPR
- Utility Billing projects for various states
- Software Development and total IT solutions for various Govt. departments
- Examination and Certification
- Facility Management

Basic operator

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators

  (+= , -= , *= , //= , %=)
- Logical Operators

  (and, or, not)
- Bitwise Operators

Arithmetic Operator
+ Addition
- Subtraction
* Multiplication
/ Division
% Modulus
** Exponent
// Floor division
9//2 -> 4

Comparison Operator
==
!=
<>
>
<
>=
<=

Bitwise Operator
& Binary AND
| Binary OR
^ Binary XOR
~ Binary Ones Complement
<< Binary Left Shift
>> Binary Right Shift

Decision making

```
x = int( input('enter marks'))
if (x>50) : print('pass')
else : print('fail')
```

Or

```
x = int( input('enter marks'))
if (x>50) :
    print('pass')
else :
    print('fail')
```

```
if expression1:
    statement(s)
    if expression2:
        statement(s)
    elif expression3:
        statement(s)
    elif expression4:
        statement(s)
    else:
        statement(s)
else:
    statement(s)
```

loops

while expression :

    statements()

```
i=0
while (i<5) :
    print (i, 'Jai Ho')
    i=i+1
```

```
0 Jai Ho
1 Jai Ho
2 Jai Ho
3 Jai Ho
4 Jai Ho
```

while expression :

    statements()

else :

    statements()

```
i=0
while (i<5) :
    print (i, 'Jai Ho')
    i=i+1
else :
    print (i, ' Its over now')
```

```
0 Jai Ho
1 Jai Ho
2 Jai Ho
3 Jai Ho
4 Jai Ho
5  Its over now
```

for loop

for iterating Variable in sequence

statement/s

```
In [3]: states=['J&K', 'HimachalPradesh','Punjab','Delhi']
        for st in states:
            print (st)
```

```
J&K
HimachalPradesh
Punjab
Delhi
```

```
In [4]: for st in range(len(states)):
            print (states[st])
```

```
J&K
HimachalPradesh
Punjab
Delhi
```

```
In [5]: for alpha in 'India':
            print(alpha)
```

```
I
n
d
i
a
```

for iterating Variable in sequence

statement/s

else:

statement/s

```
In [7]: for st in range(len(states)):
            print (states[st])
        else :
            print('------Its over ------ ')
```

```
J&K
HimachalPradesh
Punjab
Delhi
------Its over ------
```

Loop Control Statements

Break : Terminates loop statement

```
for alpha in 'Greatness':
    if alpha == 'n':
        break
    print ('letter ', alpha)
```

```
letter  G
letter  r
letter  e
letter  a
letter  t
```

continue : returns the control to the beginning of the while/for loop

```
for alpha in 'Greatness':
    if alpha == 'n':
        continue
    print ('letter ', alpha)
```

```
letter  G
letter  r
letter  e
letter  a
letter  t
letter  e
letter  s
letter  s
```

pass : is used when a statement is required syntactically but you do not want any command or code to execute

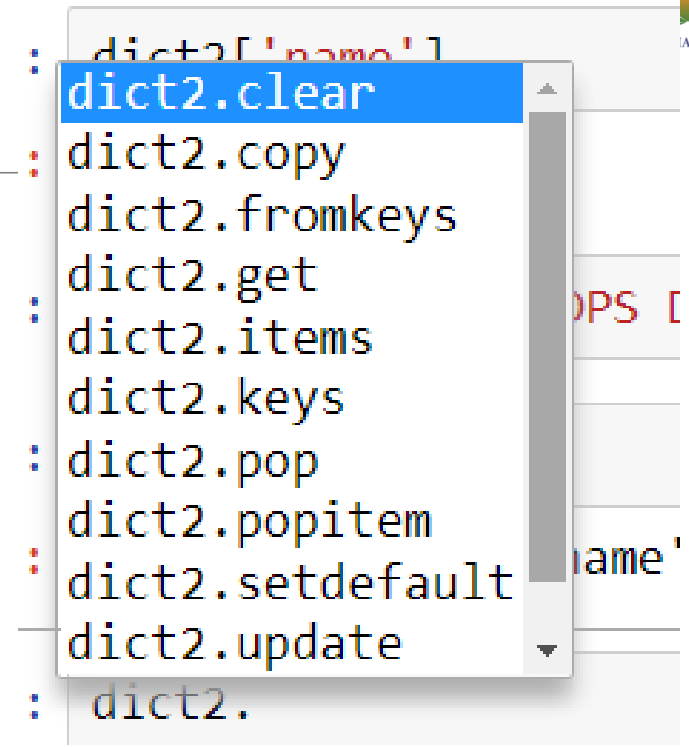
```
for alpha in 'Greatness':
    if alpha == 'n':
        pass
        print ('Pass block')
    print ('letter ', alpha)
```

```
letter  G
letter  r
letter  e
letter  a
letter  t
Pass block
letter  n
letter  e
letter  s
letter  s
```

Methods

Exercise :

- Write a python function to get all the string elements inside tuple passed as an argument (nested tuple)
  - Without recursion
  - With recursion
- Redefined method to accept list as an argument



Data structures in Computer Science

Linear Data Structure

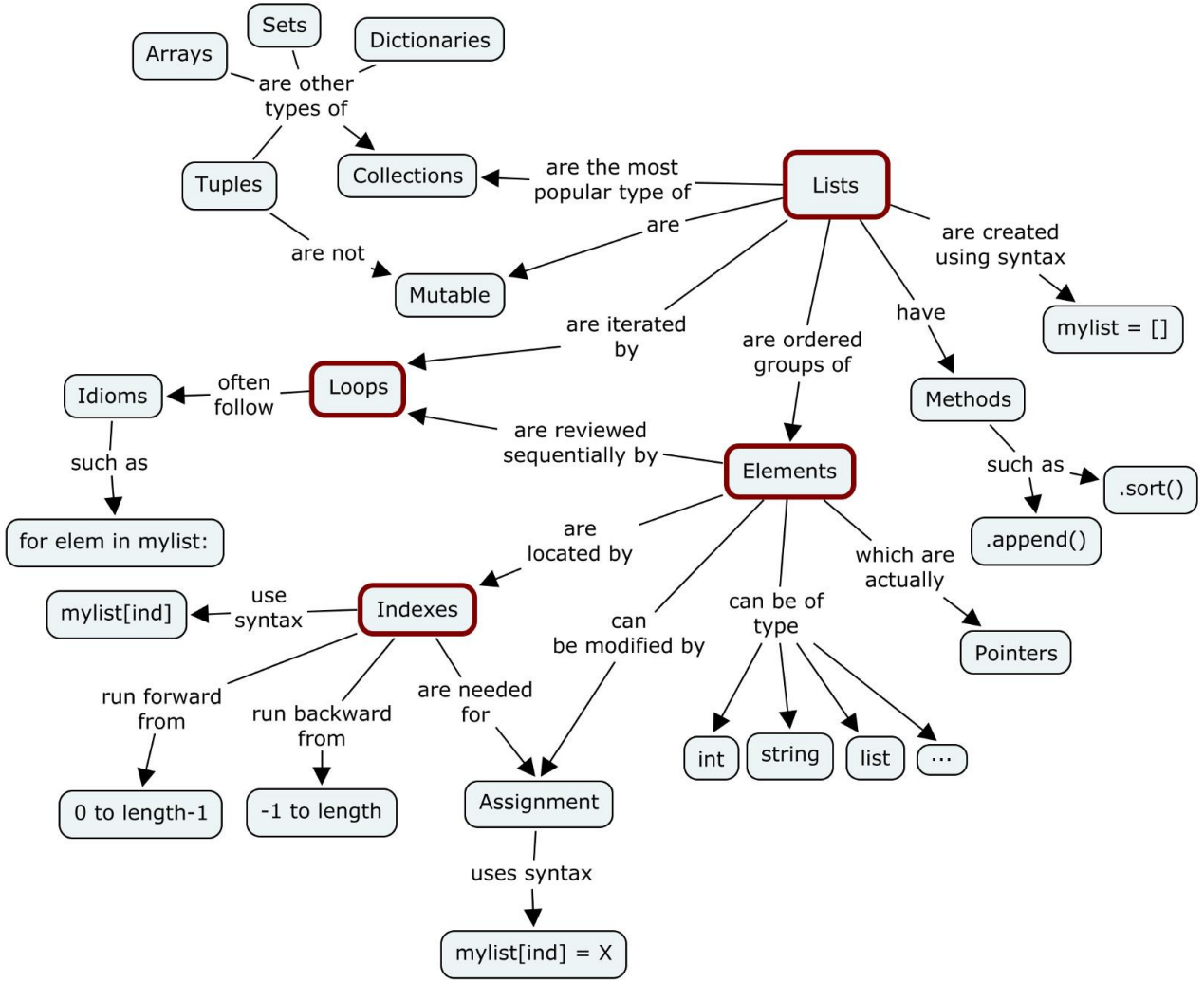
- **Array** – It is a sequential arrangement of data elements paired with the index of the data element.
- **Linked List** – Each data element contains a link to another element along with the data present in it.
- **Stack** – It is a data structure which follows only to specific order of operation. LIFO(last in First Out) or FILO(First in Last Out).
- **Queue** – It is similar to Stack but the order of operation is only FIFO(First In First Out).
- **Matrix** – It is two dimensional data structure in which the data element is referred by a pair of indices.

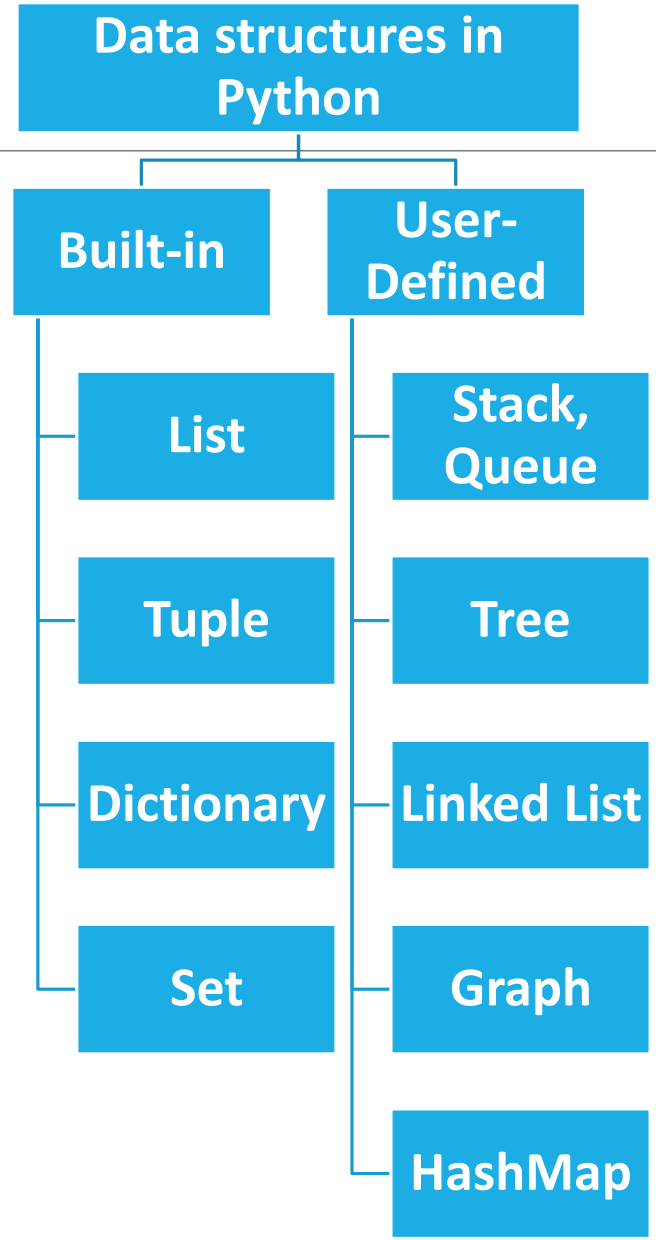
Data structures in Computer Science

Non-Liner Data Structures

- **Binary Tree** – It is a data structure where each data element can be connected to maximum two other data elements and it starts with a root node.
- **Heap** – It is a special case of Tree data structure where the data in the parent node is either strictly greater than/ equal to the child nodes or strictly less than it's child nodes.
- **Hash Table** – It is a data structure which is made of arrays associated with each other using a hash function. It retrieves values using keys rather than index from a data element.
- **Graph** – It is an arrangement of vertices and nodes where some of the nodes are connected to each other through links.

Data structures in Python





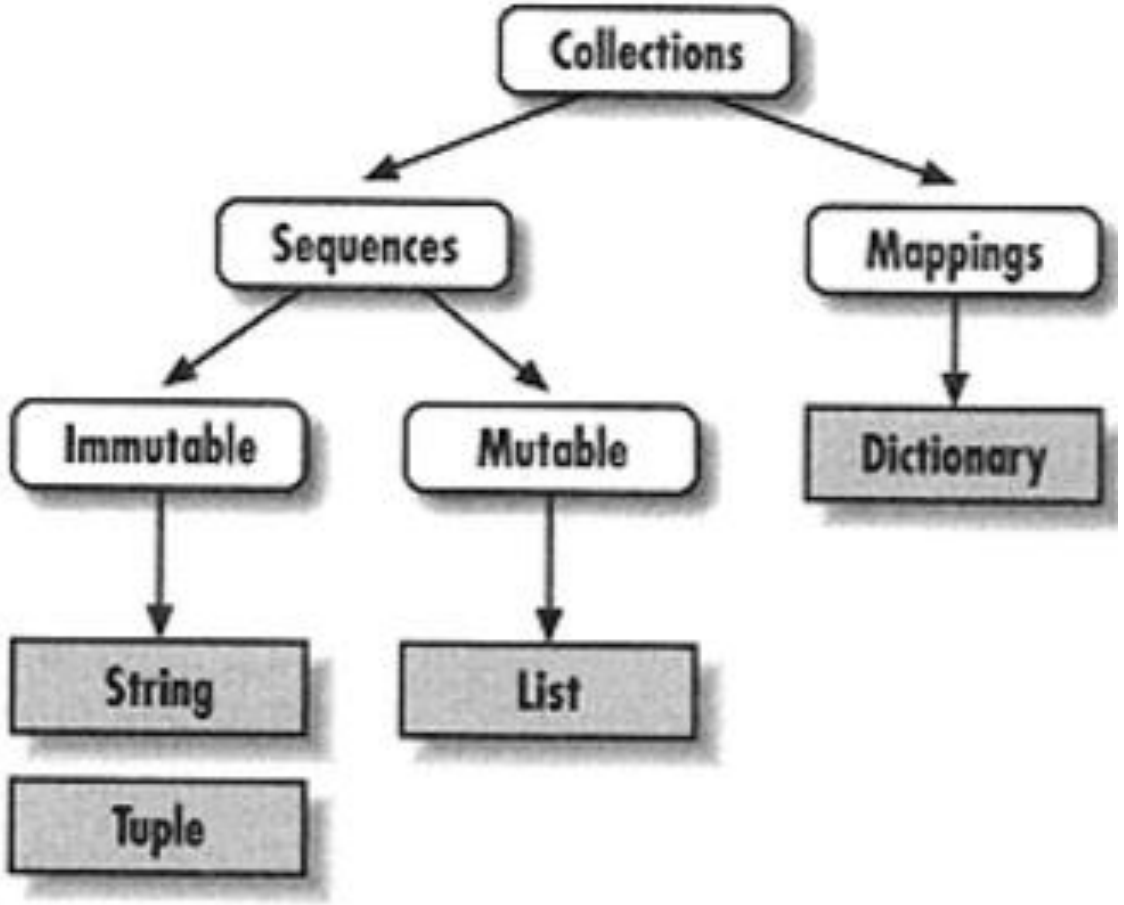
Python – Data structures

- **List –** It is similar to array with the exception that the data elements can be of different data types. You can have both numeric and string data in a python list.
- **Tuple –** Tuples are similar to lists but they are immutable which means the values in a tuple cannot be modified they can only be read.
- **Dictionary –** The dictionary contains Key-value pairs as its data elements.
- **Sets -** collection of unordered elements that are unique

User-Defined Data Structures - Python

- Arrays vs. List
- Stack
- Queue
- Trees
- Linked Lists
- Graphs
- HashMaps

Collection



"Assignment Creates References, Not Copies"

List

- Python has six built-in **sequence types** : strings, Unicode strings, lists, tuples, buffers, and xrange objects. (source : https://docs.python.org/2.4/lib/typesseq.html) .
- List is one of the popular sequence in Python.
- List is collection of objects (ordered sequence of data similar to String except that String can only hold characters)
- List need not be homogeneous , (its heterogeneous) and it is mutable
- List is arbitrarily nestable
- Arrays of object references- lists contain zero or more references to other objects (like array of pointers in C Language)

List

- Each element of List is positioned/indexed starting from 0
- Operation on Strings like indexing, slicing, adding, multiplying, and checking for membership are all available in Lists
- E.g. studentRec = [‘Amrit’, ‘kumar’ , 21, 2000]

  recFields = [‘firstname’ , ‘lastname’, ‘rollno’, ‘fee’]

```
studentRec = ['Amrit', 'kumar' , 21, 2000]

StudentList = [2, studentRec , ['Amit','jain',10,4000]]

StudentList

[2, ['Amrit', 'kumar', 21, 2000], ['Amit', 'jain', 10, 4000]]

StudentList[1]

['Amrit', 'kumar', 21, 2000]
```

Basic operations

Basic operation on List are similar to Strings

Expression	Description
len	Length
List1 + list2	Concatenation
List * 2	Repetition
‘elem’ in List	Membership
for x in List:	Iteration

```
StudentList
```

```
[2, ['Amrit', 'kumar', 21, 2000], ['Amit', 'jain', 10, 4000]]
```

```
StudentList[1]
```

```
['Amrit', 'kumar', 21, 2000]
```

```
StudentList[1:]
```

```
[['Amrit', 'kumar', 21, 2000], ['Amit', 'jain', 10, 4000]]
```

```
'Amrit' in StudentList
```

```
False
```

```
2 in StudentList
```

```
True
```

```
'Amrit' in StudentList[1]
```

```
True
```

Built-in functions and Methods

- Min (list)
- Max (list)
- Len (list)

```
list1 = [10,3,5,14,21,9,13]
print(list1)
```

```
[10, 3, 5, 14, 21, 9, 13]
```

```
list1[5:7]
```

```
[9, 13]
```

```
del list1[5]
list1[5:7]
```

```
[13]
```

```
list1 = [10,3,5,14,21,9,13]
print(list1)
```

```
[10, 3, 5, 14, 21, 9, 13]
```

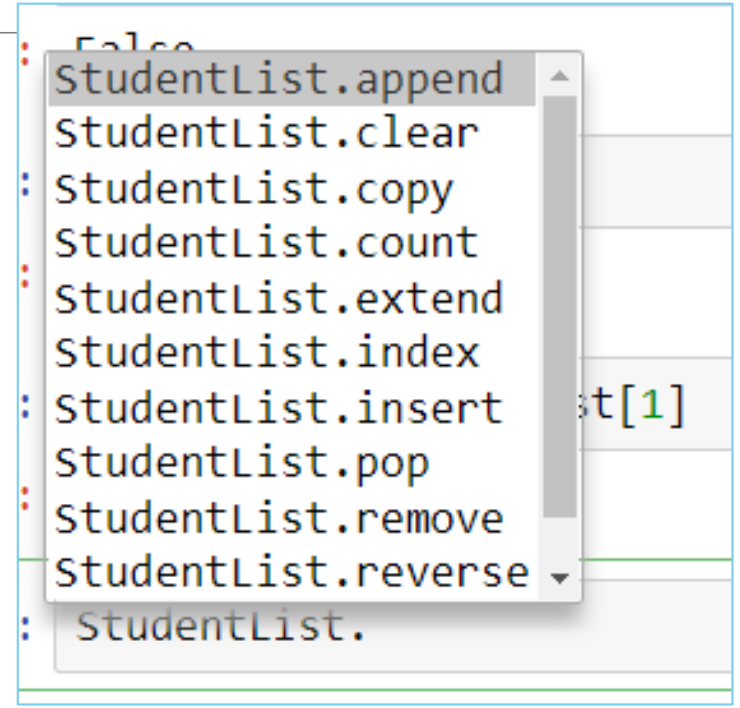
```
list1[5:7] #slicing
```

```
[9, 13]
```

```
list1[2:4] = [] #shrinking list
```

```
print(list1)
```

```
[10, 3, 21, 13]
```




```
list("34Amrit") #converting String to List
```

```
['3', '4', 'A', 'm', 'r', 'i', 't']
```

Zip

- The purpose of zip() is to **map the similar index of multiple containers** so that they can be used just using as single entity.
- passing two iterables, like lists, zip() enumerates them together

- Practical use: student database or scorecard or any other utility that requires mapping of groups.

```
StudentList[1]
```

```
['Amrit', 'kumar', 21, 2000]
```

```
recFields = ['firstname' , 'lastname', 'Rollno', 'fee']
```

```
StudentRecPrint = zip(recFields, StudentList[1]) #zip to map values
stuList = list(StudentRecPrint) #converting to list
print(stuList) #print list
```

```
[('firstname', 'Amrit'), ('lastname', 'kumar'), ('Rollno', 21), ('fee', 2000)]
```

```
header, sturecord= zip(*stuList) #unzipping values
print (header, '\n', sturecord)
```

```
('firstname', 'lastname', 'Rollno', 'fee')
 ('Amrit', 'kumar', 21, 2000)
```

LIST Equivalence/reference

== equality operator determines if two lists contain the same elements

- is operator determines if two variables alias the same list
- The association of a variable with an object is called a reference
- Aliase : An object with more than one reference has more than one name

```
a=[10,20,30,40]
```

```
b=a
c=[10,20,30,40]
```

```
print (" List a: " ,a , " id(a): ", id(a))
print (" List b: " ,b , " id(b): ", id(b))
print (" List c: " ,c , " id(c): ", id(c))
```

```
List a:  [10, 20, 30, 40]  id(a):  1326451643144
List b:  [10, 20, 30, 40]  id(b):  1326451643144
List c:  [10, 20, 30, 40]  id(c):  1326450352200
```

```
b[2] = 35
c[2] = 35
print (" List a: " ,a , " id(a): ", id(a))
print (" List b: " ,b , " id(b): ", id(b))
print (" List c: " ,c , " id(c): ", id(c))
```

```
List a:  [10, 20, 35, 40]  id(a):  1326451643144
List b:  [10, 20, 35, 40]  id(b):  1326451643144
List c:  [10, 20, 35, 40]  id(c):  1326450352200
```

```
a==b
```

```
True
```

```
a is b
```

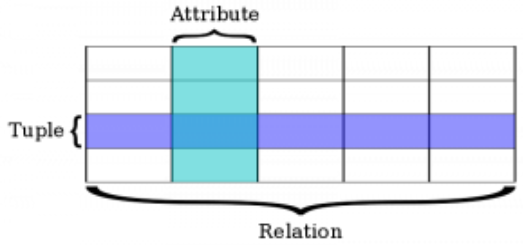
```
True
```

```
b==c
```

```
True
```

```
b is c
```

```
False
```



Python-Tuples

- ANOTHER TYPE OF SEQUENCE LIKE LIST
- IMMUTABLE
- USES ( )
- COMMA-SEPARATED LIST OF VALUES

- Tuples are immutable, cannot update or change the values
- Tuples can be concatenated (+) , deleted using del
- Other basic operation like list are same : indexing, slicing, matrixes

```
tpl = () #empty tuple
tpl
```

```
()
```

```
tpl = (10)
```

```
tpl[0]
```

```
----------------------------------------
TypeError                                 Tr
<ipython-input-91-20e03974e213> in <module>(
----> 1 tpl[0]

TypeError: 'int' object is not subscriptable
```

```
tpl = (10,)
tpl[0]
```

```
10
```

```
tpl[0]=20
```

```
----------------------------------
TypeError
<ipython-input-93-2d7cb66a897d> in
----> 1 tpl[0]=20

TypeError: 'tuple' object does not support
```

```
tpl1=(1,2)
```

```
tpl2 = tpl + tpl1
tpl2
```

```
(10, 1, 2)
```

sequence packing-unpack

- packing always creates tuple
- unpacking works for any sequence
- Parentheses is optional while packing

```
tpl = (10,'amrit', 2000.50)
```

```
rno, name, fee = tpl #unpacking
```

```
print("tuple-tpl : ", tpl)
print('Rno  :',rno)
print('Name :',name)
print('fee  :',fee)
```

```
tuple-tpl :  (10, 'amrit', 2000.5)
Rno  : 10
Name : amrit
fee  : 2000.5
```

```
tpl2 = rno, name, fee # packing
```

```
tpl2
```

```
(10, 'amrit', 2000.5)
```

Changing element of a tuple

Immutable Types Can't Be Changed in Place

```
T = (1, 2, 3)
T[2] = 4                # error!
```

```
----------------------------------------------------------
TypeError                                 Traceback (
<ipython-input-11-7bca93914e13> in <module>()
      1 T = (1, 2, 3)
----> 2 T[2] = 4                # error!
      3 T = T[:2] + (4,)        # okay: (1, 2, 4)
      4 print(T)

TypeError: 'tuple' object does not support item assig
```

```
T = T[:2] + (4,)        # okay: (1, 2, 4)
print(T)
```

```
(1, 2, 4)
```

Tuple assignment

swap a and b

```
Temp = a
a =  b
b = temp
```

- tuple assignment is more elegant

```
a, b = b, a
```

```
a, b = 1, 2, 3  # error
```

ValueError: too many values to unpack

```
email = 'monty@python.org'  un
user, domain = email.split('@')
```

Comparing tuple

```
(0, 1, 2) < (0, 3, 4)
True
```

Python-Dictionary

- KEY : VALUE PAIR SEPARATED WITH :
- USES CURLY BRACKETS { }
- KEYS ARE UNIQUE IN A DICTIONARY, VALUES MAY NOT
- VALUES OF A DICTIONARY CAN BE OF ANY TYPE, BUT THE KEYS MUST BE OF AN IMMUTABLE DATA TYPE SUCH AS STRINGS, NUMBERS, OR TUPLES

Updation

- dict2['school']='DPS Delhi'

Deletion

- del dict1 ['name']; # remove entry with key 'Name
- dict1.clear(); # remove all entries in dict1
- del dict1 ; # delete entire dictionary

```
dict1= {} #empty dictionary
```

```
print(len(dict1)); print(dict1);
```

```
0
{}
```

```
dict2 = {'rno':10,'name':'amrit', 'fee':2000.50 }
```

```
dict2
```

```
{'fee': 2000.5, 'name': 'amrit', 'rno': 10}
```

```
dict2['name']
```

```
'amrit'
```

Uses of Dictionary

- Web crawlers use dictionary for storing data
- Store database settings
- Representing application templates
- In unit tests, sample data for web forms (mapping field name to value)
- Mapping objects on game field – literally maps
- Building indexes of contents – phone book
- Exchanging data over Internet – JSON

- **Lists:** used for collection of similar/non similar items.
- **Tuples** are generally used for smaller groups of similar items

As today, more and more web apps are moving to the RESTFul design pattern, and JSON is key to the client/server data exchange.


Q
A


LOGIC
CREATIVE
SCIENCE
ART
SENSE OF TIME
INTUITION
LANGUAGE
IMAGINATION