IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

BUILT-IN DATA STRUCTURES IN PYTHON

SIDDHARTH NANDAKUMAR CHIKALKAR

BACHELOR OF COMPUTER APPLICATIONS, VIVEKANAND COLLAGE, KOHAPUR,INDIA

***ABSTRACT**: In 2020 data consuming is increasing and big amount of data has been stored by different organization and researcher have to understand the data is much important before launch some product. And understanding data we need to understand first is how many types or how many ways we can store that particulate data. So that why we have to understand data structures. Which type of data we have? that's why we have to learn what is the data structures. A data structures is special way to store and organize data in computer so it can be used efficiently. list, dictionary, tuple set, stack, queue, tree, linked list, graph, hash map are all data structures in python programming language, we can store the data in special way, so after that we can access that data effectively. each of every data structure mentioned has different way to organize the data. So, we can choose different type of data structures based on the requirements.*

Keyword: python programming, built-in data structures in python, data structures operation and functions.

I. INTRODUCTION :

In computer science the data structure is the way to store organise information or data is the way that allows you, to easily access or modify the value with them. We are live in information age now it's simple mean everything around us it's all about information a lot of information. The IT industry which you are working with is IT, means information technology, what they need is information (data) we have to store that data somewhere. how can we store data because after data processing we will get information. now the question is How we store the data ? data can be different type it can be normal data it can be complex data so it's all about how to store data .there is not that simple it's not just storing the data it's about how you can store data and process that data efficiently. if you understand what is data structures, it will be easier for you to understand how that the programming languages are work, and handle the data and perform certain operation with them. So, data structure is all about how can you structure your data so that you can store it and use it efficiently. there are certain algorithms you can implement on that data structures. We know about this concept of searching techniques, sorting techniques so we can apply all this technique to that data when your data is properly structured. python is an interpreted object-oriented high-level programming language python is often used as supportive language for software development and many other ways. in this article we will see the all built-in data structures in python programming language. And we will see kind of operation on this data structures.

II. DATA STRUCTURES IN PYTHONE

Every programme has variable value that can be need to store during programming execution of that programme .so data structures can be define as container are used to store this variable value for manipulating the available data. And we can say data structures are data organization for effectively use that data. In python programming language there are two type of data structures built in data structure and user define data structures. List, dictionary, tuple, set are the built-in data structure and stack, queue, tree, graph, hash map are the user-define data structures. Data structures are allowed to store information on hard disk. To handle a large amount of data we need a fast-processing speed that's way we need data structures, because we cannot store all type of data in one memory location, that would be very complicated to understand data and processing that data. using of data data structures, we can optimize memory usage and information can securely store in hard disk so we cannot lose our data. Main benefit of using data structures is, it saves data processing time every data structure has their rule and because of that data prepressing is done quickly in small data as well as in large amount of data.

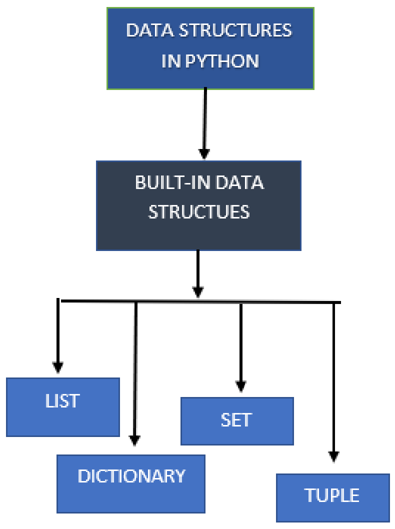

Figure 1.0 Built-in Data structures in python

III. List Data structure

List is the ordered sequence that can hold variety of object. They use [] bracket and commas to separate object in list. for example [1,2,3,5]. list support indexing and slicing. We can easily change data value in list. we could create list all mix object type. we can create list with different data type object. List use to store multiple elements in single variable .it can have any number of item and they may be different type like integer, string, float. List are also mutable means list element can be change after list created. we can perform many operations on list data structures to handle the data item efficiently.

Creating list in python and accessing element.

```
 list = [1, 2, 'English', 'Hindi', 50.23]
 print (list)
[1, 2, 'English', 'Hindi', 50.23]
```

We can create list by enclosing [square] bracket with every element separated by commas. We can access list element and we could from indexing .to accessing the list element we have to use list name and square bracket and under square bracket we have to put element index number in list. Element indexing start with 0,1,2 …. n so first element index is 0.

```
list =[1,2,"english","hindi",55.23,"marathi"]
print (list [2])
print (list [3])
print (list [4])
English
Hindi
55.23
```

We can return end element of the list by using negative index. if we want to return the first element of the end of the list then we have to use -1 index. similarly, the second last element of the list index is -2.

```
list=[1,2,"english","hindi",55.23,"marathi"]
print(list[-2])
print(list[-3])
print(list[-4])
55.23
hindi
english
```

Adding Element in List

List is mutable means we can change value after list created. so, we can add new element to an existing list .to add element we can use append () and insert (). append function add element in end of the list and insert function can add element to a specific index position. append function add element only end of the list, append function can not add element in any specific position. append function dose not return new list when we use append function it modifies the existing list by adding the item to the end of the list.

```
  list =[1,2,"english","hindi",55.23,"marathi"]
list.append("india")
print(list)
[1, 2, 'english', 'hindi', 55.23, 'marathi', 'india']
```

Append function add value in end of the list if we want to add element on specific position on list then we have to use insert function.in insert function there are two parameters (index, element) first is index number specifying in which position to insert the value. And second parameter is an element of any type (string, number, object, etc).

```
list =[1,2,"english","hindi",55.23,"marathi"]
list.insert(3,"iceland")
print(list)
[1, 2, 'english', 'iceland', 'hindi', 55.23, 'marathi', 'india']
```

Removing any element in list

Removing any element from list is easy as adding them. And we can do by using remove () or pop () function method.in remove function we have to pass the parameter as which element you want to remove in list and that element should be exist in list. in single or double coat. Pop function remove element by their associate index number this is the two method of removing element in list.

```
list =["india","iceland","USA","russia"]
list.remove("USA")
list.pop(1)
print(list)
['india', 'russia']
```

Sorting list

We can sort element in list using sort() function.most of time we use this function we can sort list element accsending or dessending order.

```
list =[5,4,6,89,1,2,5]
list.sort()
print(list)
[1, 2, 4, 5, 5, 6, 89]
```

In sorting function all element should be integer, string is not sort by sort function. string is sorted using ASCII values of the character in the string. Each character in the string has an integer value associated with it. then we use this value to sort string.

Concatenating List

We can concatenate two or more list using + symbol. using plus sign symbol two list are added and this will return new list.

```
list = [1,2,3,4,5]
another_list = [6,7,8,9,10]
third_list = [11,12,13,14,15]
fourth_list = [16,17,18,19,20]
print(list+another_list+third_list+fourth_list)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

This is all operation we can perform in list data structures and access the data item with in it. in this data structure we can easily modify, concatenating, remove, sort the list using certain function and perform task efficiently. using list data structures, we can conveniently organize our data and perform different task.

IV. DICTIONARY DATA STRUCTURE

Dictionaries data structure unordered mapping for storing object. Dictionaries use key -value pairing instead. the key value pair allows user to grab any object in dictionaries without needing to know their index location. Dictionaries use curly braces and colon to signify the key and their associate value in list we use index location to grab any object in list but in dictionary we use key value and grab any object in dictionaries. Dictionaries are retrieved object by their key and they cannot be sorted. the syntax of dictionaries is below.

```
Variable name = {'key1':"value1",'key2':"value2",'k3':"value3"}
```

creating Dictionaries and accessing element

```
my_dict = {'apple':80,'banana':20,'kiwi':90}
print(my_dict['apple'])
print(my_dict['banana'])
print(my_dict['kiwi'])
80
20
90
```

In this above example apple is the key and 80 is the associate value of that key. if you want to return value then you have to pass key which that value is associated, not index number. Dictionaries are super flexible to data type they can hold. We can put dictionary inside the dictionary and assign key and access the element .

```
my_dict = {'key1':"value1",'key2':[1,2,3,4],'k3':{'k4':"value4"}}
print(my_dict['k3']['k4'])
value4
```

in above example in my_dict there are three key and third key have another dictionary and in that dictionary k4 is the key and value4 is that key associated value ,if you want to return value4 then we have to pass first k3 key parameter and then we have to pass that associate dictionary's key value which is k4 and then we will get that element. It is not usual but it's important to understand how dictionary work.

adding element and other function

we can add element using my_dict['key1'] = "new_data_value" this method we can add many elements at one time. we cannot assign direct value we can assign value through keys. And we can change values of existing key values using update function. if you want to see all keys in dictionary then we have my_dict. keys () similarly we want to see data values in dictionary then we have my_dict.values() this is two way we can see data values and keys, see below example:

```
my_dict = {'subject':("science","mathematics","geography","history") }
 my_dict['programming_languages']= "python"," r_programming","c","c++"       #adding another key

print(my_dict)
my_dict.values()

{'subject': ('science', 'mathematics', 'geography', 'history'), 'programming_languages': ('python', ' r_programming', 'c', 'c++')}


dict_values([('science', 'mathematics', 'geography', 'history'), ('python', ' r_programming', 'c', 'c++')])
print( my_dict.keys())
dict_keys(['subject', 'programming_languages'])
print( my_dict.clear())
{}
```

in above example we first create dictionary and assign key and their value. and after that we adding new paring key and values and then we call. values () and. keys () function to seeing keys and values in dictionaries. if you want to delete every element in dictionary then we have to use clear () function using clear () function we can remove all individual element in dictionary.

dictionaries data structure it is to flexible to store data and accessing efficiently.as comparing other data structures are only hold the element. dictionaries hole key and element values that's why we can easily access element effectively using key only, rather than using an indexing value of every element. The keys of dictionary's can be any data type.it can be integer, floating point number, string, Boolean etc. but we cannot maintain any order in dictionary data structures. And sort function has not applicable on dictionary data structure.

V. Set data structure

Set is unordered collection of unique element means there are one representative of same objects we cannot have element more than one time. It is used to store multiple items in single variable. Sets are unchangeable means we cannot change item after set has been created. Set cannot have multiple items .it makes sets are highly efficient to remove duplicate value from list data structure and tuple data structures and perform a common maths operation like union and intersections. we cannot modify element contained in the set. sets are immutable type.

Creating sets and other function

```
my_set = set()              #example 1
my_set.add(1)
my_set.add(2)
my_set.add(3)
my_set.add(4)
my_set.add(4)
my_set.add(4)
my_set.add(4)
print(my_set)
{1, 2, 3, 4}

  x = set('grammar')     #example 2
    print(x)
    {'g', 'r', 'a', 'm'}
```

in above example we create a set store in my_set variable and add element in set using add function. the output look like a dictionary because curly braces but it's not dictionary it's just unordered collection of any item you want. We added four integer number four time but it prints only once that is set, if any item in list present more then one then it shows only ones .in example 2 we can see the grammar word. when we store in set and print. it returns only g, r, a, m that is a sets characteristic it shows only unique item .sets output is always in curly braces so if we cast list to set then your output will be in curly braces.

```
x = {1,2,3,4,5,6,7}
y = {5,6,7,8,9,10}
print(x.union( y))
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

x.difference_update(y)
print(x)
{1, 2, 3, 4}

z = intersection(y)
print(z)
{5, 6, 7}

listt1 = [ 'apple','banana','kiwi','oranges']
print (set(list1))
{'kiwi', 'oranges', 'banana', 'apple'}

tuple1 = ('one','two','three','four')
print(set(tuple1))
{'four', 'one', 'two', 'three'}
```

In above example we create x and y set then we use union function. the union () method returns all element in original set and all element in specific set. We can specify many sets separated by commas. We can remove item which are present in both sets, for that we have to use different_update () this function removes the item that exits in both sets. Next, we use intersection function. the intersection method returns a set that contain similarity between one or more sets. Means the return sets only have those value which are present in both sets. Next, we cast the list and tuple data structures in sets. We can cast list and tuple to passing list variable in set function. after casting list and tuple, in set we can perform union, intersection and other set methods on list and tuple. This is all about set data structure. this structure does not have multiple occurrences of the same element that’s why set be the highly useful to efficiently remove duplicate value from list and tuple data structures, and perform common mathematical operation union and intersection.

VI. Tuple Data structure

Tuple is collection of data structures which are immutable .it is also like list data structures but list is enclose with square bracket and tuple are enclose with parentheses. they are almost like list but with some distinction. Tuple store a fixed set of elements it does not allowed to change tuple element after tuple created. list has that provision to update content.

```
tup = (1,2,3,4,5)          #creating tuple
tup1 = (6,7,8,9,10)
print(tup+tup1)            #concating tuple
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
top3 = (tup, tup1)         #nested tuple
print(top3)
((1, 2, 3, 4, 5), (6, 7, 8, 9, 10))

new = ('English',) *5           #repetition in tuple
('English', 'English', 'English', 'English', 'English')

new1 = (1,2,3,4,5,6,7,8,9,10)        #slicing tuple
print(new1[1:5])
(2, 3, 4, 5)

new1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
new1[1] = 10

Type Error                          Traceback (most recent call last)
<ipython-input-28-eb942a820f92> in <module>
----> 1 new1[1] = 10

Type Error: 'tuple' object does not support item assignment

x = ('India','USA','Russia','Dubai','France','Ireland','Iceland')
y = x [1:]
```

```
('USA', 'Russia', 'Dubai', 'France', 'Ireland', 'Iceland')
```

In above tuple example we create tuple and create tuple concatenation as well as index slicing .and we make tuple nested means storing the existing two tuple variable element to another variable and its output return nested tuple. then we use repetition method in tuple means we can return element multiple time using multiple * sign after enclosed parentheses. This method also has in list data structures. next we use index slicing method we return range of item. next we try to modify element using index number but cannot do that. and, we got an error which is type error, tuple object does not have item assignment. Tuples are immutable so we cannot change element value. but we can assign one tuple value to another tuple as well as we can change tuple variable name also. in above example variable x element, we assign to y variable using indexing. tuples are immutable but we can assign elements to another variable. We cannot change element as well as we cannot remove any value in tuple. after store tuple element, then in memory, they are store in fixed location.so in tuple data structure we can't remove any data item. this is all about tuple data structure.

VII. Conclusion

Data structures are providing efficient processing of small and as well as large amount of data. we can perform different operation on data using data structures. sets data structures is helpful to removing duplicate value in data and return only unique values. Using data structures simply encourage reusability in long run as well. Using Indexing slicing we can easily grab and delete any element in data structures. Use of proper data structures can help to save lot of time of data processing while operations. Data structures store data in hard disk so data safe while we performing any task on that data, and we can access data anytime it is secure way to store the data.

VIII. REFERENCE

1. 5. Data Structures — Python 3.9.1rc1 documentation
2. Introduction to Data Structure (w3schools.in)
3. Data Structures - GeeksforGeeks
4. Data Structures: Python Tutorial - DataCamp
5. FUNDAMENTALS OF PYTHON: DATA STRUCTURES book
6. Common Python Data Structures (Guide) – Real Python
7. Data Structures in Python | Different Types of Data Structures in Python (analyticsvidhya.com)
8. Data Structures — Python for you and me 0.4.beta1 documentation (pymbook.readthedocs.io)
9. Data Structures — Python for you and me 0.4.beta1 documentation (pymbook.readthedocs.io)
10. Built-In Data Structures | A Whirlwind Tour of Python (jakevdp.github.io)
11. Python Data Structures - Lists, Tuples, Sets, Dictionaries - DataFlair (data-flair.training)
12. Python Data Structures (devopedia.org)
13. Python Data Structures – Vegibit
14. Data Structures in Java & Python and their implementations: Lists, Arrays, Tuples - Stack Overflow
15. Python - Data Structure - Tutorialspoint
16. Python Morsels
17. Dictionary Data Structures in Python 3 | DigitalOcean
18. 5. set Data Structure — Python Tips 0.1 documentation
19. Learn Data Structures and Algorithms (programiz.com)
20. Data Structures — Werkzeug Documentation (1.0.x) (palletsprojects.com)