Data Structures in Python

Python has four basic inbuilt data structures namely Lists, Dictionary, Tuple and Set. These almost cover 80% of our real world data structures. This article will cover the above mentioned topics.

Above mentioned topics are divided into four sections below.

Lists: Lists in Python are one of the most versatile collection object types available. The other two types are dictionaries and tuples, but they are really more like variations of lists.

- Python lists do the work of most of the collection data structures found in other languages and since they are built-in, you don’t have to worry about manually creating them.
- Lists can be used for any type of object, from numbers and strings to more lists.
- They are accessed just like strings (e.g. slicing and concatenation) so they are simple to use and they’re variable length, i.e. they grow and shrink automatically as they’re used.

- In reality, Python lists are C arrays inside the Python interpreter and act just like an array of pointers.

------X-

Topic of lists will be covered in detail further.

------X-

Dictionary: In python, dictionaries are similar to hash or maps in other languages. It consists of key value pairs. The value can be accessed by a unique key in the dictionary.

- Keys are unique & immutable objects.

Syntax:

```
dictionary = {"key name": value}
```

------X---------X

It is very important to notice that the key must be unique and that no duplicate keys are allowed. If you try to add a duplicate key, python will instead update the previous key rather than adding a duplicate key.

```
mydict = dict({
    'Name' : 'Rahul',
    'age' : 20
})
print(mydict)
```

```
{'Name': 'Rahul', 'age': 20}
```

```
mydict['Name'] = 'Rajesh'
print(mydict)
```

```
{'Name': 'Rajesh', 'age': 20}
```

As we can see, instead of assigning two values to the variable ‘name’, python replaces the name ‘Rahul’ with ‘Rajesh’

Tuple : Python tuples work exactly like Python lists except they are immutable, i.e. they can’t be

changed in place. They are normally written inside parentheses to distinguish them from lists (which use square brackets), but as you’ll see, parentheses aren’t always necessary. Since tuples are immutable, their length is fixed. To grow or shrink a tuple, a new tuple must be created.

Here’s a list of commonly used tuples:

```
() An empty tuple
t1 = (0, ) A one-item tuple (not an expression)
t2 = (0, 1, 2, 3) A four-item tuple
t3 = 0, 1, 2, 3 Another four-item tuple (same as prior line,
just minus the parenthesis)
t3 = (‘abc’, (‘def’, ‘ghi’)) Nested tuples
t1[n], t3[n][j] Index
t1[i:j], Slice


len(tl) Length
```

- Python3

```
# Python program to
illustrate
# tuple
tup = (1, "a", "string",
1+2)
```

```
print (tup)
print (tup[1])
```

Output:

```
(1, 'a', 'string', 3)


a
```

---------X----------X

Tuples do not support item assignment like dictionaries and lists.

```
tup = (1,2,3,4)
```

```
tup[2] = 0
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-ae5d772c7885> in <module>
----> 1 tup[2] = 0

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

Sets: Unordered collection of unique objects.

- Set operations such as union (|) , intersection(&), difference(-) can be applied on a set.
- Frozen Sets are immutable i.e once created further data can't be added to them
- {} are used to represent a set.Objects placed inside these brackets would be treated as a set.

- Python

```
# Python program to demonstrate
working of
# Set in Python

# Creating two sets
set1 = set()
set2 = set()

# Adding elements to set1
for i in range(1, 6):
    set1.add(i)

# Adding elements to set2
for i in range(3, 8):
    set2.add(i)

print("Set1 = ", set1)
print("Set2 = ", set2)
print("\n")
```

Output:

```
('Set1 = ', set([1, 2, 3, 4, 5]))


('Set2 = ', set([3, 4, 5, 6, 7]))
```

--------X---------X

If we convert a list with non-unique elements into a set, the set will only contain unique elements.

```
mylist = [1,1,1,2,2,2,3,3,4,4,5,5,6,6]
myset = set(mylist)
print(myset)
```

```
{1, 2, 3, 4, 5, 6}
```

Tuples and sets have *similar* behaviour. The only difference being that tuples can contain duplicate elements.

It is to be noted that neither tuples or sets can be sorted using python's inbuilt sort functions, because they are *unordered collections.*