*DATA STRUCTURE

→ Python data structure are ways of organizing and sorting data so that they can be accessed and modified efficiently.

→ Python provides both build in data structure and allow us to implement user defined data structure.

list : []

tuple : ()

Set : {}

dict : {key : values}

Array : Homogeneous list of values.

→

```
num = list(map(int, input("enter the value").split()))
```

Enter the value : 1, 2, 3, 4, 5

```
num
```

[1, 2, 3, 4, 5]

LIST

→ [], () Both will represent list.

→ []: It is a Heterogenous data Collector and it is ordered, mutable and allows duplicates.

→ Heterogenous data Collector: Collects all kind of data.

→ Ordered data: Hold the position.

→ mutable: any kind of changes can be done.

→ eg: l1 = ["Varsha", "30-10-2002", 5.9, True, 2+5j]

l1

['Varsha', '30-10-2002', 5.9, True, (2+5j)]

```
for i in l1:
    print(i)
```

Varsha
30-10-2002
5.9
True
2+5j.

l1.append(25)

['Varsha', '30-10-2002', 5.9, True, (2+5j), 25]

- Functions

Append(): add.

Pop(): Delete.

Split(): Split the list.

Insert(): to insert value to specific position.

remove(): to delete value.

clear(): Delete entire list.

Sort(): arranging either in ascending or descending order.

eg: Create evens, odd, prime number list from 1 to 20 num.

```
en=[]
on=[]
pn=[]
for i in range(1,21,1):
    if(i%2==0):
        en.append(i)
    else:
        on.append(i)
    for j in range(2,i,1):
        if(i%j==0):
            break
    else:
        pn.append(i)
print(en,on,pn)
```

output

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

[1, 2, 3, 7, 11, 13, 17, 19]

* Slicing

[Start position: Stop position: Step size]

default Start position: 0

Stop position is end ... If we don't mention the value it will go upto last.

Step size 1

Start value should be always lesser than the stop value.

eg: Print elements from 'a' to True.

```
l1=['a',2,'e',9.5,'Harsha',True,'Tech',34]
l1[0:6]
```

→ Print odd position elements.

```
l1[1:9:2]
```

[2, 9.5, True, 34]

→ Print position of elements 1st position is div by 3.

```
l1[3::3]
```

output:

```
[9.5, 'treeb']
```

→ write a program to find the max item from the list without using max function.

```
[4,6,1,9,2]
l2 = [4,6,1,9,2]
```

max(l2) with max function.

9.

or

```
l2=[4,6,1,9,2]
max_num = l2[0]
for i in l2:
    if(i>max_num):
        max_num=i
print(max_num)
```

→ To arange in ascending order

```
l2 = [4,6,1,9,2]
mn = l2.sort(reverse=False)
print(l2)
```

[1,2,4,6,9]

descending order

```
mn = l2.sort(reverse=True)
print(l2)
```

[9,6,4,2,1]

→ Reverse=False is ascending ~~order~~ the value defaultly.

- l3 = [1,2,3,3,3,4,4,5,6,7,8,9,9]

Remove all dupes.

```
new_l3 = []
for i in l3:
    if i not in new_l3:
        new_l3.append(i)
print(new_l3)
```

output:

[1,2,3,4,5,6,7,8,9]

- l1 = [1,2,3,4,5]
  l2 = [4,5,6,7,8]

~~inter=[]~~

~~for i in l1:~~

~~if i in l2:~~

~~pass~~

```
uni=[]
for i in l1:
    if i not in l2:
        uni.append(i)

for i in l2:
    if i not in uni:
        uni.append(i)

print(uni)
```

Output:

[1, 2, 3, 4, 5, 6, 7, 8]

* TUPLE

→ It is represented by tuple() or ()

→ It is ordered, immutable, allow, duplicates.

→ All data structures are heterogenous.

→ We can't add or delete or update in tuple but we can update it by converting tuple into list and then after updation can be converted back to tuple from list.

→ eg: t1 = (08, 'varsha', True, 79.0, 'jay')

```
Temp = list(t1)
Temp
```

[08, 'varsha', True, 79.0 'jay']

→ eg: Sort a tuple of tuples by 2nd item.

tuple1 = (('a', 23), ('b', 37), ('c', 11), ('d', 29))

expected:

(('c', 11), ('a', 23), ('d', 29), ('b', 37))

```
tuple2 = list(tuple1)
tuple2
```

[('a', 23), ('b', 37), ('c', 11), ('d', 29)]

```
tuple1 = (('a', 23), ('b', 37), ('c', 11), ('d', 29))
print (tuple1)
temp = sorted (list (tuple1), key = lambda x : x[1])
tuple1 = tuple (temp)
tuple1.
```

Output: (('c', 11), ('a', 23), ('d', 29), ('b', 37))

* SET { }

→ A set is an unordered, mutable collection of unique elements.
→ define using set() or { }
→ It automatically delete duplicates.
→ Heterogenous in nature.
→ add() used to add values, add doesn't work in list.
→ pop(): ~~dot~~ delete in set it will randomly delete a value.
→ remove(val) a specific value given in function will be deleted.

Eg: S1 = {28, 'varsha', True, 29.0, 'Jay'}

S1

{28, 'varsha', True, 29.0, 'Jay'}

→ S1.add(12)

{28, 12, 'varsha', True, 29.0, 'Jay'}

→ S1.pop()

It randomly deletes a value from any position.

→ S1.remove(28)

Eg: S1 = {1, 2, 3, 4, 5}
S2 = {4, 5, 6, 7, 8}

```
print(S1.intersection(S2))
print(S1.union(S2))
print(S1.difference(S2))
print(S2.difference(S1))
print(S.symmetric difference(S2))
```

* DICTIONARIES (dict{})

→ Fundamental data structure, that stores mappings of unique keys to values.

→ They are mutable.

```
D1 = {'name': 'Yamshi', 'age': 23, 'place': 'chennai'}
```

→ updating value.

```
D1['age'] = 24
print(D1)
```