Introduction to Python

Class 3a and 4a

Tuples:

We use round braces for tuples They can be heterogeneous, but tuples are immutable that means we cannot change its value after it's the creation.

```
tuple1=( 1 , "Hello" , 4.5 )
print(tuple1[ 1 ]) #Hello
tuple1[ 1 ]= "Bye" #Error, tuple elements cannot be edited
tuple1=tuple1+( 8 , "Bye" ) #Works, tuples can be edited as a whole
print(tuple1)
```

We can use a del keyword to delete the whole tuple.

Indexing in the tuple is the same as list and string. Creating single tuple we have to use ‘,’ at the end.

```
del tuple1
print(tuple1) #Error, variable undefined
tuple1=( 1 , "Hello" , 4.5 )
print(tuple1[ 1 ],tuple1[ -1 ]) #Hello 4.5
print(tuple1[ 1 : 3 ],tuple1[ -3 : -1 ]) #('Hello', 4.5) (1, 'Hello')
one_tuple1=(3,)
```

Below are some functions that take tuple1le as a parameter.

```
print(tuple1le([ 1 , 4 , 2 , 4 , "hello" ])) #(1, 4, 2, 4, 'hello')
print(sum(tuple1)) # error for string and int.
print(any(( 0 , 1 , 2 , 3 )) #true
print(all(( 0 , 1 , 2 , 3 )) #false
any() will return true if any value is true(non zero) , and all() will return true if all values of
tuple1le are true.
```

Below are some functions on the tuple1le object.

```
tuple1=( 1 , "Hello" , 4.5 )
print(tuple1.count( "hello" )) #1
print(tuple1.index( 4.5 )) #2
```

Sets:

A set object is an unordered collection of distinct objects. Mostly sets are used for mathematical functions such as union , intersection etc. Python gives two implementation of sets : sets and frozenset only difference is that sets are mutable and frozensets are not. Orders in sets does not matter and it has only distinct elements sets are enclosed in curly brackets.

```
set1={ "ISTE" , "PYTHON" , "SMP" }

set2=set([ "SMP" , "PYTHON" , "ISTE" ])

print(type(set1)) #<class 'set'>

print(set1==set2) #True

set1={ "ISTE" , "PYTHON" , "SMP" }

print(len(set1), "ISTE" not in set1) #3 False
```

Disjoint will return true if given sets are disjoint.

```
set1={ "ISTE" , "PYTHON" , "SMP" }

set2={ "ISTE" , "SMP" , "PYTHON" }

print(set1.isdisjoint(set2)) #False
```

There are lot ways of doing same operation on sets , issubset() is same as <= but < will check for whether set1 is proper subset of set2 or not , similarly for issuperset ().

```
print(set1.issubset(set2),set1<=set2,set1<set2)

#True True False
```

Following snippet depicts basic mathematical operation on sets. Note that both functions

and operator does the same job on sets.

```
set1={ "ISTE" , "PYTHON" }

set2={ "ISTE" , "SMP" }

set3={ 5 }

print(set3.union(set1,set2))

print(set1|set2|set3)


print(set1.intersection(set2,set3))

print(set1&set2&set1)
print(set1.difference(set3,set2))

print(set1-set2-set3)


print(set1.symmetric_difference(set2))

print(set1^set2)


#{5, 'PYTHON', 'SMP', 'ISTE'}

 #set()

#{'PYTHON'}

#{'PYTHON', 'SMP'}
```

Following snippet depicts manipulation of python set.

```
set1={ "ISTE" , "PYTHON" }

set1.add( "SMP" ) #{'PYTHON', 'ISTE', 'SMP'}

set1.remove( "ISTE" ) #{'PYTHON', 'SMP'}

set1.discard( 5 ) #{'PYTHON', 'SMP'}

set1.pop() #one of the value is popped

set1.clear() #set()
```

difference between discard and remove is discard will not raise error if element is not found, but remove will.

Sets can be used to remove repeated objects from another data structure.

```
list1=['a','b','c','a','d','b']

print(list(set(list1))) # consists of ‘a’, ’b’, ’c’, ’d’
```