Introduction to Python (Day 2)

Presenter: Dr. Sourav Saha

Structure of programming language

- Writing output
- Storing values
- Operators and Expressions
- Input Operation
- Decision Control
  - If-else
  - Loop
- **Special Data Structures: List, Tuple, Set, Dictionary**
- Function
- File Handling

LIST

Lists are just like the arrays, declared in other languages. Lists need not be homogeneous always which makes it a most powerful tool in Python. A single list may contain DataTypes like Integers, Strings, as well as Objects. Lists are also very useful for implementing stacks and queues. Lists are mutable, and hence, they can be altered even after their creation.

Creating a List

```
# Creating a List with
# the use of Numbers
# (Having duplicate values)
List = [1, 2, 4, 4, 3, 3, 3, 6, 5]
print("\nList with the use of Numbers: ")
print(List)

# Creating a List with
# mixed type of values
# (Having numbers and strings)
List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
print("\nList with the use of Mixed Values: ")
print(List)
```

Creating a List

```
# Creating a List
List = []
print("Intial blank List: ")
print(List)
```

```
# Creating a Multi-Dimensional List
# (By Nesting a list inside a List)
List = [['Geeks', 'For'] , ['Geeks']]
print("\nMulti-Dimensional List: ")
print(List)
```

```
# List Initialization
l1 = [1]*10
print(l1)
```

Accessing Values in list

```
# Creating a List with
# the use of multiple values
List = ["Geeks", "For", "Geeks"]

# accessing a element from the
# list using index number
print("Accessing a element from the list")
print(List[0])
print(List[2])

# Creating a Multi-Dimensional List
# (By Nesting a list inside a List)
List = [['Geeks', 'For'] , ['Geeks']]

# accessing a element from the
# Multi-Dimensional List using
# index number
print("Acessing a element from a Multi-Dimensional list")
print(List[0][1])
print(List[1][0])
```

Accessing Values in list

```
# Advanced Accessing
L1 = [1,2,3,4,5,6,7,8,9,10]
print(L1[2:5])
print(L1[::2])
print(L1[1:8:2]) # L1[start:stop:step]
print(L1[1::2])
print(L1[-1]) # print last element
```

Updating values in list

```
# Updating values in list
l2 = [1,2,3,4,5,6,7,8,9,10]
l2[0] = l2[0] * 10
l2[5] = 60
l2.append(11)
del l2[3]
print(l2)
del l2[2:5] # delete elements in the index range 2 to 4
print(l2)
del l2[:] # delete all
print(l2)
```

Removing Elements from the List

```
List = [1, 2, 3, 4, 5, 6,
        7, 8, 9, 10, 11, 12]
print("Intial List: ")
print(List)

# Removing elements from List
# using Remove() method
List.remove(5)
List.remove(6)
print("\nList after Removal of two elements: ")
print(List)

# Removing elements from List
# using iterator method
for i in range(1, 5):
    List.remove(i)
print("\nList after Removing a range of elements: ")
print(List)

# Removing element from the
# Set using the pop() method
List.pop()
print("\nList after popping an element: ")
print(List)

# Removing element at a
# specific location from the
# Set using the pop() method
List.pop(2)
print("\nList after popping a specific element: ")
print(List)
```

Looping through list

```
# looping through list
l1 = [1,2,3,4,5,6,7,8,9,10]
sum = 0
for i in l1:
    sum += i
print("sum = ", sum)
print("avg = ", sum/len(l1))

for index, item in enumerate(l1):
    print(index,"]",item)
```

```
s = 0
cnt = 0
for i in range(1, len(l1), 2):
    s += i
    cnt += 1
print("sum = ", s)
print("avg = ", s/cnt)
```

List utility functions

```
# List utilities
l1.insert(5, 10) # list.insert(index, item)
print(l1)
l1.reverse()
print(l1)
l1.sort()
print(l1)
l3 = [100, 200, 300]
l1.extend(l3)
print(l1)
squareList = [i ** 2 for i in range(1, 11)]
print(squareList)
max = max(l1)
print(max)
print(l1.index(max))
# list cloning
l3 = l1 # same copy
l3[0] = -1
l4 = l1[0:len(l1)] # cloning
print(l4)
# concatenation
l3 = l1 + l4
print(l3)
```

TUPLE

Tuple is a collection of Python objects much like a list. The sequence of values stored in a tuple can be of any type, and they are indexed by integers. The important difference between a list and a tuple is that tuples are immutable.

CREATING TUPLES

```
# Creating an empty tuple
Tuple1 = ()
print("Initial empty Tuple: ")
print (Tuple1)

# Creating a Tuple with
# the use of Strings
Tuple1 = ('Geeks', 'For')
print("\nTuple with the use of String: ")
print(Tuple1)

# Creating a Tuple with
# the use of list
list1 = [1, 2, 4, 5, 6]
print("\nTuple using List: ")
print(tuple(list1))
```

Accessing tuple elements

```
# accesing tuple elements
tup1 = (1,2,3,4,5,6,7,8,9,10)
print(tup1[3:6])
print(tup1[:4])
print(tup1[4:])
print(tup1[1:6:2])
tup1 = (v1, v2, v3) = (1, 2, 3)
v1 = 10
print(tup1)
print(v1, v2, v3)
# updating and deleting tuple elements are not possible
#tup1[0] = 2
```

LOOPING

```
# looping through tuple
t1 = (1,2,3,4,5,6,7,8,9,10)
sum = 0
for i in t1:
    sum += i
print("sum = ", sum)
print("avg = ", sum/len(t1))

for index, item in enumerate(t1):
    print(index,"]",item)


s = 0
cnt = 0
for i in range(1, len(t1), 2):
    s += i
    cnt += 1
print("sum = ", s)
print("avg = ", s/cnt)
```

Zip function

```
# zip utility function
tup1 = (1,2,3,4,5)
list1 = ['a', 'b', 'c', 'd', 'e']
print(list((zip(tup1, list1))))
print(tuple((zip(tup1, list1))))
tup2 = ['a', 'b', 'c', 'd', 'e']
list2 = [10, 11, 12, 13, 14]
print(list((zip(list1, list2))))
print(tuple((zip(tup1, tup2))))
```

SETS

In Python, **Set** is an unordered collection of data type that is iterable, mutable and has no duplicate elements.

Set in Python is equivalent to sets in mathematics. The order of elements in a set is undefined though it may consist of various elements. Elements of a set can be added and deleted, elements of the set can be iterated, various standard operations (union, intersection, difference) can be performed on sets. Besides that, the major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set.

Creating set

```
# Creating a Set
set1 = set()
print("Intial blank Set: ")
print(set1)

# Creating a Set with
# a mixed type of values
# (Having numbers and strings)
set1 = {1, 2, 'Geeks', 4, 'For', 6, 'Geeks'}
print("\nSet with the use of Mixed Values")
print(set1)

# Creating a Set with
# a List of Numbers
# (Having duplicate values)
set1 = set([1, 2, 4, 4, 3, 3, 3, 6, 5])
print("\nSet with the use of Numbers: ")
print(set1)
```

Adding Elements to a Set

Elements can be added to the Set by using built-in add() function. Only one element at a time can be added to the set by using add() method, loops are used to add multiple elements at a time with the use of add() method whereas for addition of two or more elements Update() method is used.

Adding Elements to a Set

```
# Creating a Set
set1 = set()
print("Intial blank Set: ")
print(set1)

set1 = {1, 2, 3, 'a'}
# Adding element to the Set
set1.add(18)
set1.add(9)
set1.add(18) # duplicate is ignored
set1.add(28)
print("\nSet after Addition of Three elements: ")
print(set1)

# Adding lists to the Set
set1.add((4, 5))
print("\nSet after Addition of a Tuple: ")
print(set1)

# Addition of elements to the Set
# using Update function
set1.update([10, 11])
print("\nSet after Addition of elements using Update: ")
print(set1)
```

Removing Elements to a Set

```
# Creating a Set
set1 = set([1, 2, 3, 4, 5, 6,
                        7, 8, 9, 10, 11, 12])
print("Intial Set: ")
print(set1)

# Removing elements from Set
# using Remove() method
set1.remove(5)
set1.remove(6)
print("\nSet after Removal of two elements: ")
print(set1)

# Removing elements from Set
# using Discard() method
set1.discard(8)
set1.discard(9)
print("\nSet after Discarding two elements: ")
print(set1)

# Removing element from the
# Set using the pop() method
set1.pop()
print("\nSet after popping an element: ")
print(set1)

# Removing all the elements from
# Set using clear() method
set1.clear()
```

Looping through set

```
# looping through set
t1 = {1,2,3,4,5,6,7,8,9,10}
sum = 0
for i in t1:
    sum += i
print("sum = ", sum)
print("avg = ", sum/len(t1))

for index, item in enumerate(t1):
    print(index,"]",item)

sum = 0
for i in range(1, len(t1), 2):
    sum += i
print("sum = ", sum)
print("avg = ", sum/len(t1))
```

Set utilities

```
# set utilities
item = 2
if item in t1:
    print("item is included")
else:
    print("item is not included")

s = {1, 2, 4}
t = {1, 9, 4}
if s == t:
    print("s and t are equal")
else:
    print("s and t are not equal")
if s.issubset(t1):
    print("s is subset of t1")
else:
    print("s is not subset of t1")

t1.add(80)
t1.add(90)
print(s.union(t1))
s.update([100, 300, 200])
print(s.intersection(t1))
print(s.difference(t1))
```

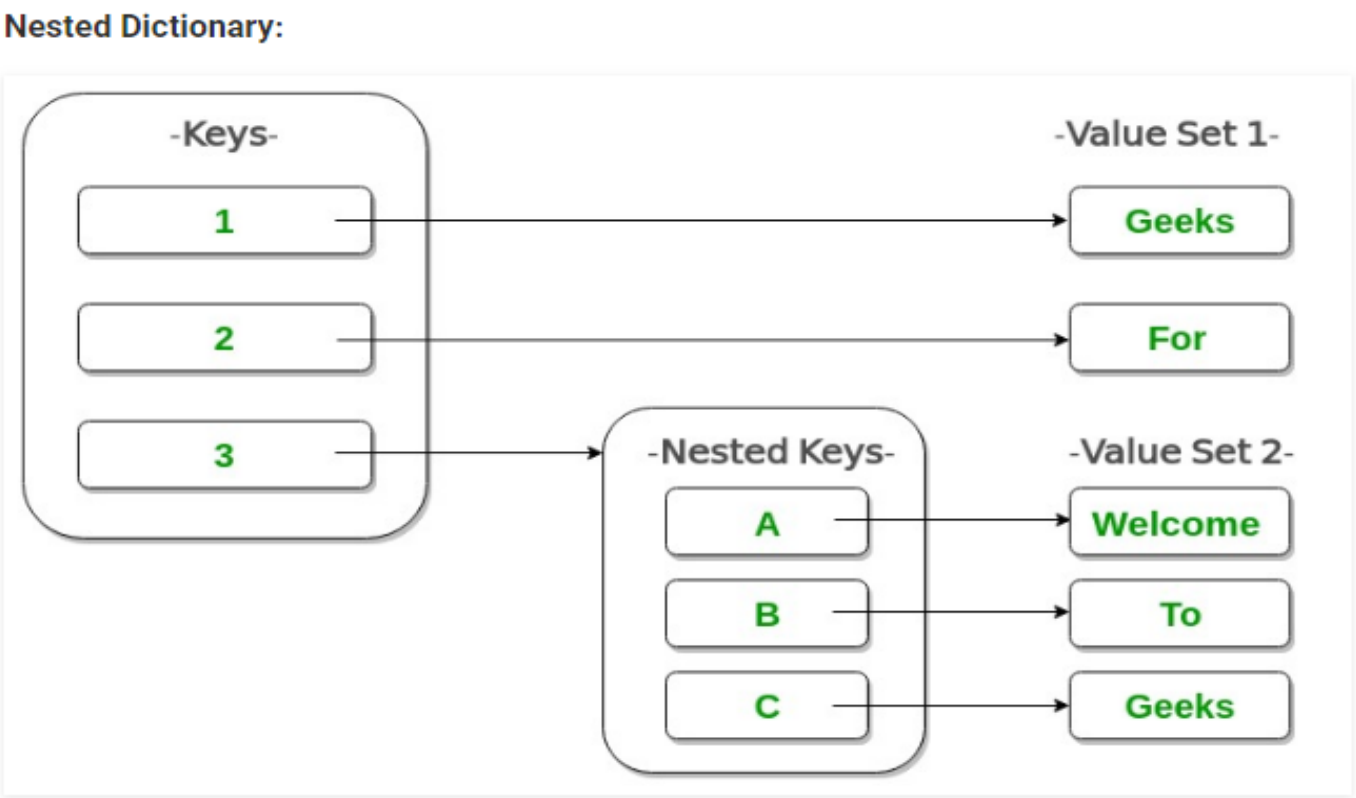
DICTIONARY

Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a ‘comma’.

Creating a Dictionary

```
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)

# Creating a Dictionary
# with Integer Keys
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print("\nDictionary with the use of Integer Keys: ")
print(Dict)

# Creating a Dictionary
# with Mixed keys
Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}
print("\nDictionary with the use of Mixed Keys: ")
print(Dict)

# Creating a Dictionary
# with dict() method
Dict = dict({1: 'Geeks', 2: 'For', 3:'Geeks'})
print("\nDictionary with the use of dict(): ")
print(Dict)

# Creating a Dictionary
# with each item as a Pair
Dict = dict([(1, 'Geeks'), (2, 'For')])
print("\nDictionary with each item as a pair: ")
print(Dict)
```

Nested dictionary




```
# Creating a Nested Dictionary
# as shown in the below image
Dict = {1: 'Geeks', 2: 'For',
                3:{'A' : 'Welcome', 'B' : 'To', 'C' : 'Geeks'}}

print(Dict)
```

Adding elements to a Dictionary

```
# Adding elements one at a time
Dict[0] = 'Geeks'
Dict[2] = 'For'
Dict[3] = 1
print("\nDictionary after adding 3 elements: ")
print(Dict)

# Adding set of values
# to a single Key
Dict['Value_set'] = 2, 3, 4
print("\nDictionary after adding 3 elements: ")
print(Dict)

# Updating existing Key's Value
Dict[2] = 'Welcome'
print("\nUpdated key value: ")
print(Dict)
```

Accessing elements from a Dictionary

```
# Python program to demonstrate
# accesing a element from a Dictionary

# Creating a Dictionary
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# accessing a element using key
print("Acessing a element using key:")
print(Dict['name'])

# accessing a element using key
print("Acessing a element using key:")
print(Dict[1])

# accessing a element using get()
# method
print("Acessing a element using get:")
print(Dict.get(3))
```

Removing Elements from Dictionary

```
# Initial Dictionary
Dict = { 5 : 'Welcome', 6 : 'To', 7 : 'Geeks',
         'A' : {1 : 'Geeks', 2 : 'For', 3 : 'Geeks'},
         'B' : {1 : 'Geeks', 2 : 'Life'}}
print("Initial Dictionary: ")
print(Dict)

# Deleting a Key value
del Dict[6]
print("\nDeleting a specific key: ")
print(Dict)

# Deleting a Key from
# Nested Dictionary
del Dict['A'][2]
print("\nDeleting a key from Nested Dictionary: ")
print(Dict)

# Deleting a Key
# using pop()
Dict.pop(5)
print("\nPopping specific element: ")
print(Dict)

# Deleting a Key
# using popitem()
Dict.popitem()
print("\nPops first element: ")
print(Dict)

# Deleting entire Dictionary
Dict.clear()
print("\nDeleting Entire Dictionary: ")
print(Dict)
```

Looping through Dictionary

```
# Looping through Dictionary
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}
for key in Dict:
    print(key, end=' ')
print()
for val in Dict.values():
    print(val, end = ' ')
print()
for key, val in Dict.items():
    print(key, ":", val, end = ', ')
print()
```

Thank You & Stay Tuned!