SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution
Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF CSE (IoT & CYBER SECURITY INCLUDING BLOCKCHAIN TECHNOLOGY)

19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

❖ **A readable, dynamic, pleasant, flexible, fast and powerful language**

Session wise Agenda

- **session 1 - List (Operations, Slice, Methods)**

- **Session 2 - List (Loop, Mutability)**

- **Session 3 - List (Aliasing, Cloning, Parameters)**

- **Session 4 - Tuples (Assignment, as return value)**

- **Session 5 - Dictionaries (operations and methods)**

- Session 6 - Advance List processing, List Comprehension

- Session 7 - Simple Sort, Histogram

- Session 8 - Student Mark Statement

- Session 9 - Retail Bill preparation

Recap

- Tuple → a sequence datatype holds heterogenous data like List.
- Tuple is immutable → values in the tuple can't be changed. But when a tuple contain list that time tuple becomes mutable as list can be changed.
- Tuple can be traversed through loop
- Tuple have in-built methods to perform operations on its values.
- Tuple assignment is done through using variables and comma separator. Multiple variables can hold a tuple value.

Dictionary

- Dictionaries are used to store data values in key:value pairs.

- A dictionary is a collection which is ordered*, changeable and does not allow duplicates.

- Dictionaries are written with curly brackets, and have keys and values

- Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be *immutable*.

- Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.

Contd..

```python
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

# using dict()
my_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

Contd..

```python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

```python
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
x = thisdict["model"]
```

```python
x = thisdict.get("model")
```

```python
x = thisdict.keys()
```

```python
x = thisdict.values()
```

```python
x = thisdict.items()
```

Contd..

```python
# get vs [] for retrieving elements
my_dict = {'name': 'Jack', 'age': 26}

# Output: Jack
print(my_dict['name'])

# Output: 26
print(my_dict.get('age'))

# Trying to access keys which doesn't exist throws error
# Output None
print(my_dict.get('address'))

# KeyError
print(my_dict['address'])
```

Adding Elements to dictionary

```python
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

Contd..

```python
# Changing and adding Dictionary Elements
my_dict = {'name': 'Jack', 'age': 26}

# update value
my_dict['age'] = 27

#Output: {'age': 27, 'name': 'Jack'}
print(my_dict)

# add item
my_dict['address'] = 'Downtown'

# Output: {'address': 'Downtown', 'age': 27, 'name': 'Jack'}
print(my_dict)
```

Deleting an element in Dictionary

```python
# create a dictionary
squares = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

# remove a particular item, returns its value
# Output: 16
print(squares.pop(4))

# Output: {1: 1, 2: 4, 3: 9, 5: 25}
print(squares)

# remove an arbitrary item, return (key,value)
# Output: (5, 25)
print(squares.popitem())

# Output: {1: 1, 2: 4, 3: 9}
print(squares)

# remove all items
squares.clear()

# Output: {}
print(squares)

# delete the dictionary itself
del squares

# Throws Error
print(squares)
```

Contd..

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
print(thisdict)
```

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```

Dictionary Methods

Sr.No.	Methods with Description
1	dict.clear() ↗ Removes all elements of dictionary *dict*
2	dict.copy() ↗ Returns a shallow copy of dictionary *dict*
3	dict.fromkeys() ↗ Create a new dictionary with keys from seq and values *set* to *value*.
4	dict.get(key, default=None) ↗ For *key* key, returns value or default if key not in dictionary

Dictionary Methods

5	dict.has_key(key) ☑
	Returns *true* if key in dictionary *dict*, *false* otherwise
6	dict.items() ☑
	Returns a list of *dict*'s (key, value) tuple pairs
7	dict.keys() ☑
	Returns list of dictionary dict's keys
8	dict.setdefault(key, default=None) ☑
	Similar to get(), but will set dict[key]=default if *key* is not already in dict
9	dict.update(dict2) ☑
	Adds dictionary *dict2*'s key-values pairs to *dict*
10	dict.values() ☑
	Returns list of dictionary *dict*'s values

Loop through Dictionary

Print all key names in the dictionary, one by one:

```
for x in thisdict:
    print(x)
```

```
for x in thisdict.keys():
    print(x)
```

Loop through Dictionary

Print all *values* in the dictionary, one by one:

```python
for x in thisdict:
    print(thisdict[x])
```

```python
for x in thisdict.values():
    print(x)
```

Loop through both *keys* and *values*, by using the `items()` method:

```python
for x, y in thisdict.items():
    print(x, y)
```

Summary

- Dictionary → a non-sequential data type which hold the value in Key:Value pairs

- The key and value can be of any type.

- Dictionary can be traversed with loop

- Dictionary have in-built methods to perform operations on values.

THANK YOU