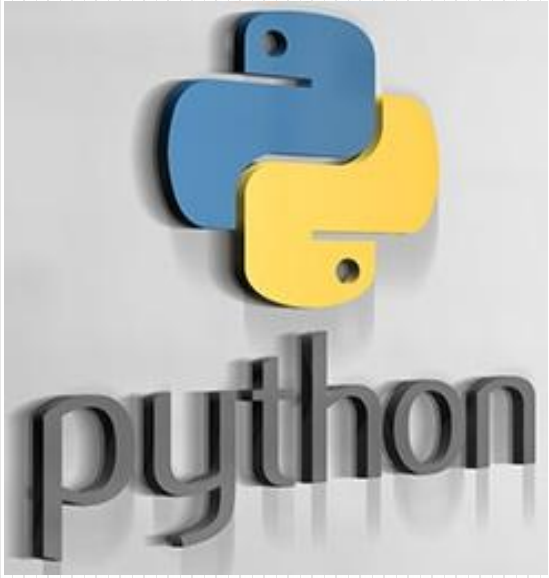
UNIT-II



K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Mutable V/s Immutable Data Types

Mutable Data Type	Immutable Data Type
Sequences can be modified after creation	Sequences cannot be modified after creation
Ex: Lists, Dictionary, Sets	Ex: Strings, Tuples
Operations like add, delete and update can be performed	Operations like add, delete and update cannot be performed

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Dictionaries

- Python dictionary is an unordered collection of items
- Dictionaries are mutable i.e., it is possible to add, modify and delete key-value pairs
- Keys are used instead of indexes
- Keys are used to access elements in dictionary and keys can be of type- strings,number,list etc
- A list of elements with key and value pairs (seperated by symbol:) inside curly braces
- The key must be unique[Immutable], separated by colon (:) and enclosed with curly braces

- While other compound datatypes have only value as an element, a dictionary has a set of key & value pair known as item.
- Creating a dictionary is as simple as placing items inside curly braces {} separated by comma.
- We can also create a dictionary using the built-in function dict( )
- Dictionary is known as Associative Array
- Ex: plant = { } # we can have our own index

```
plant [1] = "Rose"
plant[2] = "Lotus"
plant["name’"] = "Jasmin"
plant ["color"] = "Green"
print(plant)
```

Examples:

- # creation of 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')])

phonebook = { } # creation of empty Dictionary
phonebook = { “ Ravi”: 9247448766} # Dict. with K-V pair
phonebook={“Ravi”:9247448766,”Rahul”:9985933931}
# Dict. With 2 K-V Pairs

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Accessing Elements in a Dictionary

- While indexing is used with other container types to access values, dictionary uses keys.
- Key can be used either inside square brackets or with the get( ) method.
- The difference while using get( ) is that it returns None instead of KeyError, if the key is not found.

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Example

- my_dict = {'name':'student', 'age': 26}
- print (my_dict['name'])
- print (my_dict.get('age'))
- print (my_dict.get('address'))
- print (my_dict['address'])

- **Output**

  student
  26
  None
  KeyError: 'address'

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Changing or Adding Elements in a Dictionary

- Dictionary are mutable.
- We can add new items or change the value of existing items using assignment operator.
- If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

Example

- my_dict={'age': 26, 'name': 'Rahul'}
- my_dict['age'] = 27 # update value
- print my_dict

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

Output

{'age': 27, 'name': 'Ranjit'}

{'age': 27, 'name': 'Ranjit', 'address': 'Downtown'}

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Deleting or Removing Elements from a Dictionary

- We can remove a particular item in a dictionary by using the method pop().
- This method removes as item with the provided key and returns the value.
- The method, popitem() can be used to remove and return an arbitrary item (key, value) form the dictionary.
- All the items can be removed at once using the clear() method.
- We can also use the del keyword to remove individual items or the entire dictionary itself.

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Example

```
squares = {1:1, 2:4, 3:9, 4:16, 5:25}  # create a dictionary
print  squares.pop(4)  # remove a particular item
print squares
```

Output: 16

{1: 1, 2: 4, 3: 9, 5: 25}

```
squares = {1:1, 2:4, 3:9, 4:16, 5:25}  # create a dictionary
print  squares.popitem()  # remove an arbitrary item
```

Output: (5, 25)

```
squares = {1:1, 2:4, 3:9, 4:16, 5:25}  # create a dictionary
del squares[5]  # delete a particular item
print squares
```

Output: {1: 1, 2: 4, 3: 9, 4: 16}

```
squares = {1:1, 2:4, 3:9, 4:16, 5:25}  # create a dictionary
squares.clear()  # remove all items
print  squares
```

Output: {}

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Dictionary Comprehension

- Dictionary comprehension is an elegant and concise way to create new dictionary from an iterable in Python.
- Dictionary comprehension consists of an expression pair (key: value) followed by for statement inside curly braces {}.
- Here is an example to make a dictionary with each item being a pair of a number and its square.

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Example

```
squares = {x: x*x for x in range(6)}
print squares
```

Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

```
#This code is equivalent to

squares = {}
for x in range(6):
   squares[x] = x*x
print squares
```

Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Dictionary Comprehension

- A dictionary comprehension can optionally contain more for or if statements.
- An optional if statement can filter out items to form the new dictionary. Here are some examples to make dictionary with only odd items.

Example

```
odd_squares = {x: x*x for x in range(11) if x%2 == 1}
print  odd_squares
```

Output: {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Dictionary Membership Test

- We can test if a key is in a dictionary or not using the keyword in. Notice that membership test is for keys only, not for values.

Example

```
squares ={1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print 1 in squares
print 2 not in squares
# membership tests for key only not value
print 49 in squares
```

Output

```
True
True
False
```


K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Iterating Through a Dictionary

Using a for loop we can iterate though each key in a dictionary.

Example:

```
squares={1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
for i in squares:
    print(squares[i])
```

Output:

```
1
9
25
49
81
```


K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Built-in Functions with Dictionary

- all( ) Return True if all keys of the dictionary are true (or if the dictionary is empty).
- any( ) Return True if any key of the dictionary is true. If the dictionary is empty, return False.
- len ( ) Return the length (the number of items) in the dictionary.
- cmp(d1,d2) Compares items of two dictionaries.
- sorted ( ) Return a new sorted list of keys in the dictionary.

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

any( ) & all( )

- any(l == 't' for l in 'python')
  # Returns True. Same as: 't' in 'python‘
- all(l == 't' for l in 'python')
  # Returns False. Not all of the letters are 't'.

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Ex:

```
square={0: 0, 0: 9, 0: 7, 0: 12}
squares={1: 1, 3: 9, 5: 25, 7: 49}
print len(squares)
print sorted(squares)
print cmp(squares,square)
```

Output: 4

[1, 3, 5, 7]

1

Ex:

```
squares={}
print all(squares)
print any(squares)
```

Output: True

False

Dictionary Methods
str( ) Method

- The method str() produces a printable string representation of a dictionary.

Syntax: str(dict)

Example

```
dict = {'Name': 'Zara', 'Age': 7};
print ("Equivalent String : %s" % str (dict))
```

Output

```
Equivalent String : {'Name': 'Zara', 'Age': 7}
```

copy() Method

- The method copy() returns a shallow copy of the dictionary.

Syntax: dict.copy()

Example

```
dict1 = {'Name': 'Zara', 'Age': 7};
dict2 = dict1.copy()
print ("New Dictinary : %s" %  str(dict2))
```

Output

```
New Dictinary : {'Age': 7, 'Name': 'Zara'}
```

fromkeys( ) Method

- The method fromkeys( ) creates a new dictionary with keys from seq and values set to value.
- If we dont specify a value in Dict. it assumes as keyword None

Syntax: dict.fromkeys(seq[, value]))

Example

```
seq = ('name', 'age', 'sex')
dict = dict.fromkeys(seq)
print "New Dictionary : %s" %  str(dict)
dict = dict.fromkeys(seq, 10)
print "New Dictionary : %s" %  str(dict)
```

Output

```
New Dictionary : {'age': None, 'name': None, 'sex': None}
New Dictionary : {'age': 10, 'name': 10, 'sex': 10}
```

get( ) Method

- The method get() returns a value for the given key.
- If key is not available then returns default value None.

Syntax: dict.get(key, default=None)

Example

```
dict = {'Name': 'Zara', 'Age': 27}
print "Value : %s" %  dict.get('Age')
print "Value : %s" %  dict.get('Sex', "Never")
print ("Value : %s" %  dict.get('Sex'))
```

Output

```
Value : 27
Value : Never
 Value : None
```

has_key( ) Method

- The method has_key() returns true if a given key is available in the dictionary, otherwise it returns a false.

Syntax: dict.has_key(key)

Example

```
dict = {'Name': 'Zara', 'Age': 7}
print "Value : %s" %  dict.has_key('Age')
print "Value : %s" %  dict.has_key('Sex')
```

Output

```
Value : True
Value : False
```

items() Method

- The method items() returns a list of dict's (key, value) tuple pairs

Syntax: dict.items()

Example

```
dict = {'Name': 'Zara', 'Age': 7}
print "Value : %s" %  dict.items()
```

Output

```
Value : [('Age', 7), ('Name', 'Zara')]
```

keys( ) Method

- The method keys() returns a list of all the available keys in the dictionary.

Syntax: dict.keys()

Example

```
dict = {'Name': 'Zara', 'Age': 7}
print ("Value : %s" %  dict.keys())
```

Output

Value : dict_keys(['Name', 'Age'])

setdefault() Method

- The method setdefault() is similar to get(), but will set dict[key]=default if key is not already in dict.

Syntax: dict.setdefault(key, default=None)

Example

```
dict = {'Name': 'Zara', 'Age': 7}
print "Value : %s" %  dict.setdefault('Age', None)
print "Value : %s" %  dict.setdefault('Sex', None)
```

Output

```
Value : 7
Value : None
```

update( ) Method

- The method update() adds dictionary dict2's key-values pairs in to dict. This function does not return anything.

Syntax: dict.update(dict2)

Example

```
dict = {'Name': 'Zara', 'Age': 7}
dict2 = {'Sex': 'female' }
dict.update(dict2)
print ("Value : %s" %  dict)
```

Output

```
Value : {'Age': 7, 'Name': 'Zara', 'Sex': 'female'}
```

values( ) Method

- The method values() returns a list of all the values available in a given dictionary.

Syntax: dict.values()

Example

```
dict = {'Name': 'Zara', 'Age': 7}
print "Value : %s" %  dict.values()
```

Output

Value : [7, 'Zara']

Sets

- An un-ordered collection of unique elements
- Are lists with no index value and no duplicate entries
- Can be used to identify unique words used in a paragraph
- Operations like intersection,difference,union etc can be performed on sets

Syntax:

```
set1={ }   # creation of empty set
set2={"Python"} # set with an element
```

Ex:

```
s1=set{"my name is Python and Python is my
  name".split()}
s1={'is','and','my','name','Python'}
```

Operations on Sets

Operation	Equivalent	Operation
len(s)		Length of set 's'
x in s		Membership of 'x' in 's'
x not in s		Membership of 'x' not in 's'
s.issubset(t)	s <= t	Check whether 's' is subset 't'
s.issuperset(t)	s >= t	Check whether 't' is superset of 's'
s.union(t)	s | t	Union of sets 's' and 't'
s.Intersection	s & t	Intersection of sets 's' and 't'
s.difference(t)	s- t	Returns elements in 's' but not in 't'
s.symmetric_difference(t)	s ^ t	Returns elements in either 's' or 't' but not both
s.copy(_)		A new copy of 's'

Windows

If you're using Windows, then here's your list of commands:

- Pwd — print working directory
- hostname — my computer's network name
- mkdir — make directory
- cd — change directory
- ls — list directory
- rmdir — remove directory
- pushd — push directory
- popd — pop directory
- cp — copy a file or directory
- robocopy — robust copy

Windows

If you're using Windows, then here's your list of commands:

- mv — move a fi le or directory
- more — page through a fi le
- type — print the whole fi le
- forfiles — run a command on lots of fi les
- dir - r — find fi les
- select- string — find things inside fi les
- help — read a manual page
- helpctr — find what man page is appropriate
- echo — print some arguments
- set — export/set a new environment variable
- exit — exit the shell
- runas — DANGER! become super user root DANGER!

IIIT BASARA

Infosys | Campus Connect

For Details Contact Me @ :

9247448766

ravikanth27787@gmail.com

python

Programming Language