Dictionaries

सीबीएसई पाठ्यक्रम पर आधारित

कक्षा -11

अध्याय - 9

द्वारा:

संजीव भदौरिया

स्नातकोत्तर शिक्षक (संगणक विज्ञान)

के० वि० बाराबंकी (लखनऊ संभाग)

परिचय

- Python हमें collections को संगठित करने के लिए कई ऐसी सुविधाएँ प्रदान करता है जिनमे हम एक variable के अंतर्गत values का ढेर store कर सकते हैं |
- Dictionaries भी एक string, list और tuple जैसा ही collection होता है |
- यह एक बहुउपयोगी (versatile) प्रकार का data type होता है |
- इसमें एक key तथा उस key की value होती है (key:value)
- Dictionaries एक प्रकार का mutable data type है और ये key:value के जोड़े के रूप में एक अव्यवस्थित संग्रह(unordered collection) होता है|
- List में आपको value का index ध्यान रखना होता था जबकि dictionary में value की key का ध्यान रखना होता है|

Dictionary बनाना

- Dictionary बनाने के लिए आपको " { } " के अंतर्गत key:value के pairs को संग्रहीत करना होता है |
  - <dictionary-name>={ <key1>:<value1>,<key2>:<value2>,<key3>:<value3>. . . }

उदाहरण:

teachers={"Rajeev":"Math", "APA":"Physics","APS":"Chemistry:"SB":"CS"}

उपरोक्त उदाहरण में :

Key-value pair	Key	Value
"Rajeev":"Math"	"Rajeev"	"Math"
"APA":"Physics"	"APA"	"Physics"
"APS":"Chemistry"	"APA"	"Chemistry"
"SB":"CS"	"SB"	"CS"

Dictionary बनाना

- Dictionary के कुछ सामान्य उदाहरण निम्न हैं –

Dict1= { } # यह एक empty dictionary है जिसमे कोई भी element नहीं है |

DayofMonth= { January”:31, ”February”:28, ”March”:31, ”April”:30, ”May”:31, ”June”:30,
”July”:31, ”August”:31, ”September”:30, ”October”:31, ”November”:30,
”December”:31}

FurnitureCount = { “Table”:10, “Chair”:13, “Desk”:16, “Stool”:15, “Rack”:15 }

– उपरोक्त उदाहरणों से आप समझ सकते हैं कि keys और उनकी सम्बंधित values कौन कौन सी हैं |
– बस यहाँ एक ध्यान रखने वाली बात ये है कि keys सदैव immutable type की रहनी चाहिए |

Note: Dictionary को associative array या mapping या hashes भी कहते हैं |

Dictionary बनाना

- बस यहाँ एक ध्यान रखने वाली बात ये है कि keys सदैव immutable type की रहनी चाहिए |

- यदि आप mutable type की key बनाने की कोशिश करते हैं तो पाइथन error दिखायेगा | उदहारण के लिए -

```
>>> dict = {[2,3]:"MyRoom"}
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    dict = {[2,3]:"MyRoom"}
TypeError: unhashable type: 'list'
```

यहाँ key एक list है जो कि mutable type की है |

यहाँ इस error का मतलब है कि अपने एक ऐसी key ली है जिसका type mutable है और पाइथन इसको समर्थन नहीं करता है |

Dictionary को Access करना

- जब भी हमें dictionary से value access करना होता है तो हम key का प्रयोग करते हैं, ठीक वैसे ही जैसे list से वैल्यू को access करने के लिए index का प्रयोग करते हैं |
- Key: value के जोड़े से हमें key का पता चलता है |

  teachers={“Rajeev”:”Math”, “APA”:”Physics”,”APS”:”Chemistry:”SB”:”CS”}

- उपरोक्त उदाहरण से यदि हम निम्न statement को क्रियान्वित करें तो -

```
>>> teachers={"Rajeev":"Math", "APA":"Physics", "APS":"Chemistry", "SB":"CS"}
>>> teachers["Rajeev"]
'Math'
>>> print("Rajeev Teaches ", teachers["Rajeev"])
Rajeev Teaches  Math
```

- यदि हमने key “Rajeev” को चुना और print किया तो उसकी वैल्यू math आयी | एक अन्य उदाहरण देखें तो -

```
>>> d={"Vowel1":'a', "Vowel2":'e', "Vowel3":'i', "Vowel4":'o', "Vowel5":'u'}
>>> print(d["Vowel2"])
e
>>> print(d["Vowel5"])
u
```

यदि आप किसी नॉन key को access के लिए देते हैं तो error आजायेगी |

Dictionary को Traverse करना

- Dictionary को traverse करने के लिए for loop का प्रयोग करते हैं जिसका प्रारूप निम्न है -

```
for <item> in <dictionary>:
        प्रत्येक item को यहाँ प्रोसेस करें |
```

```
>>> d={5:"number","a":"String",(1,2):"tuple"}
>>> for k in d:
        print(k," : ",d[k])
```

```
5   :   number
a   :   String
(1, 2)   :   tuple
```

यहाँ गौर करने वाली बात ये है कि dictionary ***d*** के प्रत्येक जोड़े की key, loop के ***k variable*** में आ रही है | उसके बाद print statement के साथ दिए गए फॉर्मेट में हम output ले सकते हैं |

Assignment : आप अपने मित्रों की एक फ़ोन dictionary बनाइये जिसमे key, मित्र का नाम हो और वैल्यू, उसका फ़ोन नंबर हो |

Dictionary को Traverse करना

- यदि हमें key और values को access करना हो तो keys( ) और values( ) फंक्शन का प्रयोग करते हैं | इस हेतु निम्न statement लिखते हैं -

```
>>> d={"Vowel1":'a', "Vowel2":'e', "Vowel3":'i', "Vowel4":'o', "Vowel5":'u'}
>>> d.keys()
dict_keys(['Vowel1', 'Vowel2', 'Vowel3', 'Vowel4', 'Vowel5'])
>>> d.values()
dict_values(['a', 'e', 'i', 'o', 'u'])
```

- d.keys( ) फंक्शन सिर्फ key को प्रदर्शित करेगा |
- d.values ( ) फंक्शन सिर्फ values को प्रदर्शित करेगा |

Dictionary के गुण

1. **Unordered set:** dictionary एक प्रकार का अव्यवस्थित key:value जोड़े का समूह होता है
2. **ये sequence नहीं होता है:** list, string और tuple की तरह यह sequence नहीं होता है क्योंकि यह एक प्रकार का elements का अव्यवस्थित समूह होता है | जबकि sequence को संख्याओं से क्रम में indexed किया जाता है जिसके कारण वे क्रमित (Ordered) होती हैं |
3. इनकी indexing करने के लिए keys का प्रयोग किया जाता है तथा पाइथन के अनुसार key कोई भी immutable type की हो सकती है | चूँकि string और number immutable होते हैं अतः आप इन्हें key के रूप में ले सकते हैं| विभिन्न keys के साथ एक उदहारण निम्न है -

```
>>> d={0:"Key0",1:"Key1","3":"KeyAsString",(4,5):"KeyAsTuple","Hello":6}
>>> d[0]
'Key0'
>>> d[1]
'Key1'
>>> d["3"]
'KeyAsString'
>>> d[(4,5)]
'KeyAsTuple'
>>> d["Hello"]
6
```

Dictionary की keys हमेशा immutable type की होनी चाहिए, जैसे number, string या tuple. जबकि dictionary की value किसी भी प्रकार की हो सकती है |

Dictionary के गुण

4. **Keys को unique होना चाहिए:** चूँकि keys का प्रयोग values को identify करने के लिए किया जाता है अतः keys का unique होना आवश्यक है |
5. जबकि दो unique keys की values एक सामान हो सकती हैं |
6. Dictionary परिवर्तनीय (mutable) है इसमें हम किसी निश्चित key की value को बदल सकते हैं | इसके लिए निम्न syntax का प्रयोग कर सकते हैं |

<dictionary>[<key>] = <value>

```
>>> d={0:"Key0",1:"Key1","3":"KeyAsString",(4,5):"KeyAsTuple","Hello":6}
>>> d["3"]="This is String"
>>> d
{0: 'Key0', 1: 'Key1', '3': 'This is String', (4, 5): 'KeyAsTuple', 'Hello': 6}
```

7. आन्तरिक रूप से यह mapping के रूप में संग्रहीत(store) होती है| इसके key:value जोड़े एक दूसरे से एक आतंरिक फंक्शन के द्वारा सम्बंधित रहते हैं ये फंक्शन ***hash-function*** कहलाते हैं | link करने की इस विधा को mapping कहते हैं|

**Hash-function, key और value को map और link करने की एक आतंरिक(internal) algorithm होती है |

Dictionary के साथ काम करना

- यहाँ पर हम dictionary पर होने वाले विभिन्न operation के बारे में चर्चा करेंगे, जैसे dictionary में element को जोड़ना (adding), सुधार करना (Update), delete करना इत्यादि | लेकिन उसके पहले dictionary बनाना सीखते हैं |

1. **dictionary को initialize करना:** इसके लिये हम key:value जोड़े को कोमा (,) से पृथक (separate) करते हुए उनका एक समूह बनाते हैं और उस समूह को मझले कोष्ठक " { }" में रख देते हैं | जैसे -

```
>>> Employee={'name':'suresh','salary':15000,'age':34}
>>> Employee
{'name': 'suresh', 'salary': 15000, 'age': 34}
```

2. **Empty dictionary में key:value जोड़े को add करना:** empty dictionary बनाने के लिए दो तरीके हैं इनमे से कोई एक तरीका अपनाये -
   1. Employee = { }
   2. Employee = dict( )

   उसके बाद उसमे निम्न syntax लगायें

   <dictionary>[<key>] = <value>

```
>>> Employee = {}
>>> Employee['name']='Pankaj'
>>> Employee['salary']=20000
>>> Employee
{'name': 'Pankaj', 'salary': 20000}
```

Dictionary के साथ काम करना

3. Dictionary को name और value जोड़े के साथ बनाना : इसके अंतर्गत dict( ) constructor का प्रयोग करके key और value के जोड़े बनाकर dictionary बनाई जा सकती है | ऐसा करने के कई तरीके हैं

I. Key:value pair को argument के रूप में पास करके :

```
>>> Employee=dict(name='Ramesh',salary=10000,age=24)
>>> Employee
{'name': 'Ramesh', 'salary': 10000, 'age': 24}
```

इसमें गौर करिए की dictionary में name, salary एवं age पर अपने आप single inverted comma आगया है जबकि argument में नहीं लगाया गया है|

II. Comma-separated key:value जोड़े को specify करके :

```
>>> Employee=dict({'name':'Rahul','age':24})
>>> Employee
{'name': 'Rahul', 'age': 24}
```

Dictionary के साथ काम करना

III. Keys को अलग और values को अलग specify करके:

इसके लिए dict ( ) constructor के अन्दर zip ( ) फंक्शन का प्रयोग करेंगे -

```
>>> Employee=dict(zip(('name','salary','age'),('Mukesh',12000,26)))
>>> Employee
{'name': 'Mukesh', 'salary': 12000, 'age': 26}
```

IV. Key:value pair को अलग अलग sequence के रूप में देकर:

```
>>> Employee=dict([['name','anand'],['salary',14000],['age',23]])
>>> Employee
{'name': 'anand', 'salary': 14000, 'age': 23}
```

List पास करके

```
>>> Employee=dict((['name','Angad'],['salary',11000],['age',29]))
>>> Employee
{'name': 'Angad', 'salary': 11000, 'age': 29}
```

List की tuple पास करके

```
>>> Employee=dict((('name','Suman'),('salary',17000),('age',21)))
>>> Employee
{'name': 'Suman', 'salary': 17000, 'age': 21}
```

Tuple की tuple पास करके

Dictionary में element को जोड़ना

Dictionary में element को जोड़ने के लिए निम्न syntax का प्रयोग करते हैं |

<dictionary>[<key>]=<value>

```
>>> Employee=dict((('name','Suman'),('salary',17000),('age',21)))
>>> Employee
{'name': 'Suman', 'salary': 17000, 'age': 21}
>>> Employee['Dept']='Sales'
>>> Employee
{'name': 'Suman', 'salary': 17000, 'age': 21, 'Dept': 'Sales'}
```

Dictionary में Nesting

निम्न उदाहरण पर ध्यान दें जिसमे dictionary के elements में dictionary ही है |

```
Employee={'mukesh':{'age':23,'salary':34000},'Meena':{'age':27,'salary':24000}}
for key in Employee:
    print("Employee",key,':')
    print('Age: ',str(Employee[key]['age']))
    print('Salary: ',str(Employee[key]['salary']))
```

```
Employee mukesh :
Age:  23
Salary:  34000
Employee Meena :
Age:  27
Salary:  24000
```

Dictionary में element को सुधारना (Updation)

Dictionary में element को सुधरने के लिए निम्न syntax का प्रयोग करते हैं |

<dictionary>[<ExistingKey>]=<value>

```
>>> Employee={'name':'Sudha','Salary':10000,'age':24}
>>> Employee['Salary']=15000
>>> Employee
{'name': 'Sudha', 'Salary': 15000, 'age': 24}
```

WAP to create a dictionary containing names of employee as key and their salary as value.

```
File  Edit  Format  Run  Options  Window  Help
n=int(input("Enter the Number of Employees"))
Employee={}    #Empty Dictionary
for a in range(n):
    key=input("Enter Name of the Employee")
    value=int(input("Enter Salary of Employee"))
    Employee[key]=value
print("The Dictionary is now is :")
print(Employee)
```

Output

```
Enter the Number of Employees3
Enter Name of the EmployeePavan
Enter Salary of Employee30000
Enter Name of the EmployeeRakesh
Enter Salary of Employee12000
Enter Name of the EmployeeMukesh
Enter Salary of Employee20000
The Dictionary is now is :
{'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
```

Dictionary से element को Delete करना

Dictionary से element को delete करने के लिए निम्न दो syntaxes का प्रयोग करते हैं | delete करने के लिए key का उपस्थित होना ज़रूरी है अन्यथा पाइथन error दे देगा|

1. del <dictionary>[<key>] ये सीधे delete करता है बिना deleted value को return नहीं करता है |

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> del emp['Pavan']
>>> emp
{'Rakesh': 12000, 'Mukesh': 20000}
```

Delete करने के बाद value return नहीं हुई

2. <dictionary>.pop(<key>) ये element delete करने के साथ साथ deleted value को return भी करता है |

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> emp.pop('Rakesh')
12000
>>> emp.pop('Aman',"Not Found")
'Not Found'
```

Delete करने के बाद value return हुई

ऐसा करने से यदि key नहीं match होती है तो साथ में दिया गया सन्देश print होजाता है |

Dictionary में element की उपस्थिति पता करना

Dictionary में element की उपस्थिति पता करने के लिए membership operator का प्रयोग करते हैं |

1. <key> in <dictionary> यह key उपस्थित होने पर True देता है अन्यथा False
2. <key> not in <dictionary> यह key के न उपस्थित होने पर True देता है अन्यथा False

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> 'Pavan' in emp
True
>>> 'Hari' in emp
False
>>> 'Hari' not in emp
True
>>> 'Pavan' not in emp
False
```

ध्यान रखने वाली यह बात है की *in* और *not in* values पर लागू नहीं होते| ये सिर्फ key के साथ ही कार्य करेंगे |

Dictionary की Pretty Printing

Dictionary को सही से सजा के print करने के लिए ***json module*** को import करना होगा | उसके बाद dumps ( ) के निम्न syntax का प्रयोग करना होगा |

json.dumps(<>,indent=<n>)

```
>>> import json
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> print(json.dumps(emp,indent=2))
{
  "Pavan": 30000,
  "Rakesh": 12000,
  "Mukesh": 20000
}
```

एक program जो किसी वाक्य में शब्दों की गणना करके dictionary बनाएगा

```
import json
statement="His Name is Pankaj. \
His father is a teacher. His \
father is a good person"
w=statement.split()
d={}
for c in w:
    key=c
    if key not in d:
        count=w.count(key)
        d[key]=count
print("Counting frequencies in list\n",w)
print(json.dumps(d,indent=1))
```

```
Counting frequencies in list
 ['His', 'Name', 'is', 'Pankaj.', '
His', 'father', 'is', 'a', 'teacher
.', 'His', 'father', 'is', 'a', 'go
od', 'person']
{
 "His": 3,
 "Name": 1,
 "is": 3,
 "Pankaj.": 1,
 "father": 2,
 "a": 2,
 "teacher.": 1,
 "good": 1,
 "person": 1
}
```

यहाँ शब्द और उनकी बारंबारता (frequency) की एक dictionary बन गयी है ।

Dictionary Function और Method

1. *len( )* Method : यह dictionary की लम्बाई बताता है |

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> len(emp)
3
```

2. *clear( )* Method : यह dictionary को खाली कर empty बना देता है |

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> emp.clear()
>>> emp
{}
```

3. *get( )* Method : यह दिए गए key की value को return करता है |

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> emp.get('Rakesh')
12000
>>> emp.get('Ankit',"Not Found")
'Not Found'
```

यह <dictionary>[<key>] की भांति ही कार्य करता है|

इसमें यदि key न मिले तो default सन्देश भी दिया जा सकता है |

Dictionary Function और Method

4. ***items( )*** Method : यह dictionary के समस्त items को (key:value) के tuple के रूप में return करता है |

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> mylist=emp.items()
>>> mylist
dict_items([('Pavan', 30000), ('Rakesh', 12000), ('Mukesh', 20000)])
```

5. ***keys( )*** Method : यह dictionary के keys की list return करता है |

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> emp.keys()
dict_keys(['Pavan', 'Rakesh', 'Mukesh'])
```

6. ***values( )*** Method : यह dictionary के values की list return करता है

```
>>> emp={'Pavan': 30000, 'Rakesh': 12000, 'Mukesh': 20000}
>>> emp.values()
dict_values([30000, 12000, 20000])
```

Dictionary Function और Method

7. ***Update ( )*** Method: यह फंक्शन किसी dictionary में दूसरी dictionary के key:value जोड़े का विलय (merge) कर देता है | इसमें आवश्यकतानुसार परिवर्तन तथा addition दोनों संभव हैं | उदहारण :

```
>>> emp1={'name':'Suresh','salary':10000,'age':24}
>>> emp2={'name':'siya','salary':45000,'dept':'sales'}
>>> emp1.update(emp2)
>>> emp1
{'name': 'siya', 'salary': 45000, 'age': 24, 'dept': 'sales'}
>>> emp2
{'name': 'siya', 'salary': 45000, 'dept': 'sales'}
```

उपरोक्त उदहारण में यदि आप देखें तो emp1 में सामान key के values में परिवर्तन हुआ जबकि असमान key अपनी value के साथ जुड़ गयी |

और अधिक पाठ्य-सामग्री हेतु निम्न लिंक पर क्लिक करें -

www.pythontrends.wordpress.com