BCSE101E

Computer Programming: Python

Course Faculty: Rajesh M – SCOPE, VIT Chennai

Module – 4 Collections in Python (List, Tuple, Dictionary, Set)

Dictionary in Python

Problem

Write a kids play program that prints the capital of a country given the name of the country.

PAC For Quiz Problem

Input	Processing	Output
A set of question/ answer pairs and a question	Map each question to the corresponding answer. Find the answer for the given question	Answer for the question

Pseudocode

```
READ num_of_countries
FOR i=0 to num_of_countries
        READ name_of_country
        READ capital_of_country
        MAP name_of_country to capital_of_country
END FOR
READ country_asked
GET capital for country_asked
PRINT capital
```

Dictionary

Imagine that you **want to find the definition** of the word "**python**". **What do you do?**
You go to a **dictionary** and look up "python".
Now that you have found "python", do you know what a python is?

Dictionary - Introduction

- **Pair of items.**
- Each pair has **key** and **value.**
- **Keys** should be **unique**
- **Key** and **value** are **separated by :**
- **Each pair** is **separated by ,**

Example:

```
dict = {‘Alice’ : 1234, ‘Bob’ : 1235}
```

Dictionary

A **dictionary** has two parts,

a **key** ("python")

a **value** ("the best darn programming language ever invented")

Properties of Dictionaries

- unordered **mutable collections**
- items are **stored** and **fetched by key**
- **Accessed by key**, **not by offset position**
- **Unordered collections** of **arbitrary objects**
- **Variable-length**, **heterogeneous**, and **arbitrarily nestable.**

Creating a Dictionary

- **Creating** an **EMPTY dictionary**

  dictname = {}

Example:

```
Dict1 = {}
MyDict = {}
Books = {}
```

- **Creating** a **dictionary with items**

  dict = {key1:val1, key2:val2,….}

Example:

```
MyDict  = { 1 : ‘Chocolate’,
            2 : ‘Icecream’}
MyCourse = {‘MS’ : ‘Python’, ‘IT’ : ‘C’,
            ‘CSE’ : ‘C++’, ‘MCA’ : ‘Java’}
MyCircle = {‘Hubby’:9486028245,
‘Mom’:9486301601}
```

Accessing Values

- **Using keys within square brackets**

```
>>> print (MyDict[1])
    ‘Chocholate’
>>> print (MyCourse[‘CSE’])
     ‘C++’
```

Updating Elements

- **update by adding a new item (key-value)** pair
- **modify an existing entry**

```
>>>MyDict[1] = ‘Pizza’
>>>MyCourse[‘MCA’] = ‘UML’
```

Deleting Elements

- **remove an element** in a dictionary **using** the **key**

```
>>>del MyCourse['IT']
```

- **remove all** the elements

```
>>>MyCourse.clear()
```

- **delete** the dictionary

```
>>>del MyCourse
```

Basic Operations

```
>>> len(D)
# Number of entries in dictionary
3
>>> 'ham' in D
# Key membership test alternative
True
>>> list(D.keys())
# Create a new list of D's keys
['eggs', 'spam', 'ham']
```

Basic Operations

```
>>> D = {'spam': 2, 'ham': 1, 'eggs': 3}
>>> list(D.values())
[3, 2, 1]
>>> list(D.items())
[('eggs', 3), ('spam', 2), ('ham', 1)]
>>> D.get('spam')                  # A key that is there
2
>>> print(D.get('toast'))      # A key that is missing
None
>>> D.get('toast', 88) # Key is assigned a value if given
88
```

Update Method

```
>>> D {'eggs': 3, 'spam': 2, 'ham':1}
>>> D2 = {'toast':4, 'muffin':5}
# Lots  of delicious scrambled order here
>>> D.update(D2)
>>> D
{'eggs': 3, 'muffin': 5, 'toast': 4, 'spam': 2, 'ham':1}
```

Pop Method

```
# pop a dictionary by key
>>> D {'eggs': 3, 'muffin': 5, 'toast': 4, 'spam': 2, 'ham':1}
>>> D.pop('muffin')
5
>>> D.pop('toast')     # Delete and return from a key
4
>>> D {'eggs': 3, 'spam': 2, 'ham':1}
```

List vs Dictionary

```
>>>  L = []
>>> L[99]  = 'spam'
Traceback (most recent call last):  File "<stdin>", line 1, in ?
    IndexError: list assignment index out of range
>>>D = {}
>>> D[99]  = 'spam'
>>> D[99] 'spam'
>>> D {99: 'spam'}
```

Nesting in dictionaries

```
>>> rec = {}
>>> rec['name'] = 'Bob'
>>> rec['age']  = 40.5
>>> rec['job']  = 'developer/manager'
>>> print(rec['name'])
Bob
```

Nesting in dictionaries

```
>>> rec = {'name': 'Bob',
...        'jobs': ['developer', 'manager'],
...        'web':  'www.bobs.org/~Bob',
...        'home': {'state': 'Overworked', 'zip': 12345}}
```

- A list can be within a dictionary and dictionary within dictionary

Nesting in dictionaries

```
>>> rec['name']
'Bob'
>>> rec['jobs']
['developer', 'manager']
>>> rec['jobs'][1]
'manager'
>>> rec['home']['zip']
12345
```

Other Ways to Make Dictionaries

```
{'name': 'Bob', 'age': 40}  # Traditional literal expression
D = {}                           # Assign by keys dynamically
D['name'] = 'Bob'
D['age']  = 40


# dict keyword argument form
dict(name='Bob', age=40)

# dict key/value tuples form
dict([('name', 'Bob'), ('age', 40)])
```

Comprehensions in Dictionaries

```
>>> D = {k: v for (k, v) in zip(['a', 'b', 'c'], [1, 2, 3])}
>>> D {'b': 2, 'c': 3, 'a': 1}
>>> D = {x: x ** 2 for x in [1, 2, 3, 4]}
# Or: range(1, 5)
>>> D
{1: 1, 2: 4, 3: 9, 4: 16}
>>> D = {c: c * 4 for c in 'SPAM'}
# Loop over any iterable
>>> D
{'S': 'SSSS', 'P': 'PPPP', 'A': 'AAAA',  'M': 'MMMM'}
```

Comprehensions in Dictionaries

```
>>> D = {c.lower(): c + '!' for c in ['SPAM', 'EGGS', 'HAM']}

>>> D {'eggs': 'EGGS!', 'spam': 'SPAM!', 'ham': 'HAM!'}
```

Initializing Dictionaries

```
# Initialize  dict from keys

>>> D = dict.fromkeys(['a', 'b', 'c'], 0)

>>> D {'b': 0, 'c': 0, 'a': 0}

# Same, but with a comprehension

>>> D = {k:0 for k in ['a', 'b', 'c']}

>>> D {'b': 0, 'c': 0, 'a': 0}
```

Initializing Dictionaries

```
# Other iterables, default value

>>> D = dict.fromkeys('spam')

>>> D {'s': None, 'p': None, 'a': None, 'm': None}

# Comprehension

>>> D = {k: None for k in 'spam'}

>>> D {'s': None, 'p': None, 'a': None, 'm': None}
```

Dictionary methods

- <dict>.items()
  - displays the items in the dictionary (pair of keys and values)
- <dict>.keys() / <dict>.viewkeys()
  - display the keys in the dictionary
- <dict>.values() / <dict>.viewvalues()
  - displays the values in the dictionary
- <dict>.pop()
  - removes the last item from the dictionary
- <dict2> = <dict1>.copy()
  - copies the items from dict1 to dict2
- <dict>.clear()
  - removes all the items from the dictionary

Other methods

- `str(dict)`
  - produces printable string representation of a dictionary

- `len(dict)`
  - returns the number of items in the dictionary

Dictionaries can replace elif ladder/switch-case

```
choice = 3

print ({1:’one’,2:’two’,3:’three’,4:’four’,5:’five’}
[choice])
```

Prints ‘three’

Exercise 1:

Write a program to maintain a **telephone directory of the employees** of an organization. **If the employee has more than one number** store all the numbers. Write a program to print the mobile numbers given full or part of the name of the employee.

Eg: Given name of the employee as '**John'** the program must print phone numbers of 'John Paul' and 'Michel John'.

Exercise 2:

Write a program to **store the name of the players against each of a 20-20 cricket team**. The program should print the name of the players given the team name.