11 Dictionaries

"Versatility, thy names is dictionary..."

Contents

KanNotes

What are Dictionaries?

- Dictionary is a collection of key-value pairs.
- Dictionaries are also known as maps or associative arrays.
- A dictionary contains comma separated key : value pairs enclosed within { }.

```
d1 = { }       # empty dictionary
d2 = {'A101' : 'Amol', 'A102' : 'Anil', 'B103' : 'Ravi'}
```

Here, 'A101', 'A102', 'B103' are keys, whereas, 'Amol', 'Anil', 'Ravi' are values.

- Different keys may have same values.

```
d = {10 : 'A', 20 : 'A', 30 : 'Z'}     # ok
```

- Keys must be unique. If keys are same, but values are different, latest key value pair gets stored.

```
d = {10 : 'A', 20 : 'B', 10 : 'Z'}     # will store {10 : 'Z', 20 : 'B'}
```

- If key value pairs are repeated, then only one pair gets stored.

```
d = {10 : 'A', 20 : 'B', 10 : 'A'}     # will store {10 : 'A', 20 : 'B'}
```

Accessing Dictionary Elements

- Entire dictionary can be printed by just using the name of the dictionary.

```
d = {'A101' : 'Amol', 'A102' : 'Anil', 'B103' : 'Ravi'}
print(d)
```

- Unlike sets, dictionaries preserve insertion order. However, elements are not accessed using the position, but using the key.

```
d = {'A101' : 'Dinesh', 'A102' : 'Shrikant', 'B103' : 'Sudhir'}
print(d['A102'])      # prints value for key 'A102'
```

Thus, elements are not position indexed, but key indexed.

- Dictionaries cannot be sliced using [ ].

Looping in Dictionaries

- Like strings, lists, tuples and sets, dictionaries too can be iterated over using a **for** loop. There are three ways to do so:

```
courses = {'DAA' : 'CS', 'AOA' : 'ME', 'SVY' : 'CE' }

# iterate over key-value pairs
for k, v in courses.items( ) :
    print(k, v)

# iterate over keys
for k in courses.keys( ) :
    print(k)

# iterate over keys - shorter way
for k in courses :
    print(k)

# iterate over values
for v in courses.values( ) :
    print(v)
```

- While iterating through a dictionary using a for loop, if we wish to keep track of index of the key-value pairs that is being referred to, we can do so using the built-in **enumerate( )** function.

```
courses = {'DAA' : 'CS', 'AOA' : 'ME', 'SVY' : 'CE' }
for i, (k, v) in enumerate(courses.items( )) :
    print(i,k)
```

Note that ( ) around **k**, **v** are necessary.

Basic Dictionary Operations

- Dictionaries are mutable. So we can perform add/delete/modify operations on a dictionary.

```
courses = { 'CS101' : 'CPP', 'CS102' : 'DS', 'CS201' : 'OOP',
            'CS226' : 'DAA', 'CS601' : 'Crypt', 'CS442' : 'Web'}
# add, modify, delete
courses['CS444'] = 'Web Services'       # add new key-value pair
```

```
courses['CS201'] = 'OOP Using java'    # modify value for a key
del(courses['CS102'])                  # delete a key-value pair
del(courses)                           # delete dictionary object
```

- Note that any new addition will take place at the end of the existing dictionary, since dictionary preserves the insertion order.
- Dictionary keys cannot be changed in place.
- Given below are the operations that work on lists and tuples. These operations are discussed in detail in Chapter 8. Try these operations on dictionaries as an exercise.

Concatenation - doesn't work	Merging - doesn't work
Conversion - works	Aliasing - works
Cloning - works	Searching - works
Identity - works	Comparison - doesn't work
Emptiness - works	

- Two dictionaries cannot be concatenated using +.
- Two dictionaries cannot be merged using the form **z = s + t**.
- Two dictionary objects cannot be compared using <, >.

Using Built-in Functions on Dictionaries

- Many built-in functions can be used with dictionaries.

```
d = { 'CS101' : 'CPP', 'CS102' : 'DS', 'CS201' : 'OOP'}
len(d)          # return number of key-value pairs
max(d)          # return maximum key in dictionary d
min(d)          # return minimum key in dictionary d
sorted(d)       # return sorted list of keys
sum(d)          # return sum of all keys if keys are numbers
any(d)          # return True if any key of dictionary d is True
all(d)          # return True if all keys of dictionary d are True
reversed(d)     # can be used for reversing dict/keys/values
```

- Use of reversed function to reverse a dictionary by keys is shown below:

```
courses = { 'CS101' : 'CPP', 'CS102' : 'DS', 'CS201' : 'OOP'}
for k, v in reversed(courses.items( )) :
    print(k, v)
```

Dictionary Methods

- There are many dictionary methods. Many of the operations performed by them can also be performed using built-in functions. The useful dictionary methods are shown below:

```
c = { 'CS101' : 'CPP', 'CS102' : 'DS', 'CS201' : 'OOP'}
d = { 'ME126' : 'HPE', 'ME102' : 'TOM', 'ME234' : 'AEM'}

print(c.get('CS102', 'Absent'))      # prints DS
print(c.get('EE102', 'Absent'))      # prints Absent
print(c['EE102'])                    # raises keyerror

c.update(d)              # updates c with items in d
print(c.popitem( ))      # removes and returns item in LIFO order
print(c.pop('CS102')     # removes key and returns its value
c.clear( )               # clears all dictionary entries
```

Note that while updating a dictionary if keys are same, values are overwritten.

popitem( ) is useful in destructively iterate through a dictionary.

Dictionary Varieties

- Keys in a dictionary must be unique and immutable. Numbers, strings or tuples can be used as keys. If tuple is used as a key it should not contain any mutable element like list.

```
d = { (1, 5) : 'ME126', (3, 2)  : 'ME102', (5, 4) : 'ME234'}
```

- Dictionaries can be nested.

```
contacts = {
                'Anil': {'DOB' : '17/11/98', 'Favorite' : 'Igloo'},
                'Amol': {'DOB' : '14/10/99', 'Favorite' : 'Tundra'},
                'Ravi': {'DOB' : '19/11/97', 'Favorite' : 'Artic'}
            }
```

- Two dictionaries can be merged to create a third dictionary by unpacking the two dictionaries using **. If we use * only keys will be unpacked.

```
animals = {'Tiger' : 141, 'Lion' : 152, 'Leopard' : 110}
birds = {'Eagle' : 38, 'Crow': 3, 'Parrot' : 2}
```

```
combined = {** animals, ** birds }
```

- A dictionary containing different keys but same values can be created using a **fromkeys( )** function as shown below:

```
lst = [12, 13, 14, 15, 16]
d = dict.fromkeys(lst, 25)  # keys - list items, all values set to 25
```

Problem 11.1

Create a dictionary called **students** containing names and ages. Copy the dictionary into **stud**. Empty the **students** dictionary, as **stud** continues to hold the data.

Program

```
students = {'Anil' : 23, 'Sanjay' : 28, 'Ajay' : 25}
stud = students    # shallow copy, stud starts referring to same dictionary
students = { }     # students now refers to an empty dictionary
print(stud)
```

Output

```
{'Anil': 23, 'Sanjay': 28, 'Ajay': 25}
```

Tips

- By making a shallow copy, a new dictionary is not created. **stud** just starts referring (pointing) to the same data to which **students** was referring (pointing).
- Had we used **students.clear( )** it would have cleared all the data, so **students** and **stud** both would have referred to an empty dictionary.

Problem 11.2

Create a list of cricketers. Use this list to create a dictionary in which the list values become keys of the dictionary. Set the values of all keys to 50 in the dictionary created.

Program

```
lst = ['Sunil', 'Sachin', 'Rahul', 'Kapil', 'Sunil', 'Rahul']
d = dict.fromkeys(lst, 50)
print(len(lst))
print(len(d))
print(d)
```

Output

```
6
4
{'Sunil': 50, 'Sachin': 50, 'Rahul': 50, 'Kapil': 50}
```

Tips

- The list may contain duplicate items, whereas a dictionary always contains unique keys. Hence when the dictionary is created from list, duplicates are eliminated, as seen in the output.

Problem 11.3

Write a program to sort a dictionary in ascending/descending order by key and ascending/descending order by value.

Program

```
import operator
d = {'Oil' : 230, 'Clip' : 150, 'Stud' : 175, 'Nut' : 35}
print('Original dictionary : ', d)

# sorting by key
d1 = sorted(d.items( ))
print('Asc. order by key : ', d1)
d2 = sorted(d.items( ), reverse = True)
print('Des. order by key : ', d2)

# sorting by value
d1 = sorted(d.items( ), key = operator.itemgetter(1))
print('Asc. order by value : ', d1)
d2 = sorted(d.items( ), key = operator.itemgetter(1), reverse = True)
```

```
print('Des. order by value : ', d2)
```

Output

```
Original dictionary :  {'Oil': 230, 'Clip': 150, 'Stud': 175, 'Nut': 35}
Asc. order by key :  [('Clip', 150), ('Nut', 35), ('Oil', 230), ('Stud', 175)]
Des. order by key :  [('Stud', 175), ('Oil', 230), ('Nut', 35), ('Clip', 150)]
Asc. order by value :  [('Nut', 35), ('Clip', 150), ('Stud', 175), ('Oil', 230)]
Des. order by value :  [('Oil', 230), ('Stud', 175), ('Clip', 150), ('Nut', 35)]
```

Tips

- By default items in a dictionary would be sorted as per the key.
- To sort by values we need to use **operator.itemgetter(1)**.
- The **key** parameter of **sorted( )** requires a key function (to be applied to be objects to be sorted) rather than a single key value.
- **operator.itemgetter(1)** gives a function that grabs the first item from a list-like object.
- In general, **operator.itemgetter(n)** constructs a callable that assumes an iterable object (e.g. list, tuple, set) as input, and fetches the n^{th} element out of it.

Problem 11.4

Write a program to create three dictionaries and concatenate them to create fourth dictionary.

Program

```
d1 = {'Mango' : 30, 'Guava': 20}
d2 = {'Apple' : 70, 'Pineapple' : 50}
d3 = {'Kiwi' : 90, 'Banana' : 35}
d4 = { }
for d in (d1, d2, d3) :
    d4.update(d)
print(d4)

# one more way
d5 = { **d1, **d2, **d3}
```

```
print(d5)

# will unpack only the keys into the list
d6 = list({*d1, *d2, *d3})
print(d6)
```

Output

```
{'Mango': 30, 'Guava': 20, 'Apple': 70, 'Pineapple': 50, 'Kiwi': 90, 'Banana': 35}
{'Mango': 30, 'Guava': 20, 'Apple': 70, 'Pineapple': 50, 'Kiwi': 90, 'Banana': 35}
[Apple', 'Guava', 'Kiwi', 'Mango', 'Banana', 'Pineapple']
```

Tips

- From the output it can be observed that the dictionaries are merged in the order listed in the expression.
- Note that list of keys is constructed from a dictionary they are not stored in the order listed in the expression.

Problem 11.5

Write a program to check whether a dictionary is empty or not.

Program

```
d1 = {'Anil' : 45, 'Amol' : 32}
if bool(d1) :
   print('Dictionary is not empty')
d2 = { }
if not bool(d2) :
   print('Dictionary is empty')
```

Output

```
Dictionary is not empty
Dictionary is empty
```

Problem 11.6

Suppose there are two dictionaries called **boys** and **girls** containing names as keys and ages as values. Write a program to merge the two dictionaries into a third dictionary.

Program

```
boys = {'Nilesh' : 41, 'Soumitra' : 42, 'Nadeem' : 47}
girls = {'Rasika' : 38, 'Rajashree': 43, 'Rasika' : 45}
combined = {**boys, **girls}
print(combined)
combined = {**girls, **boys}
print(combined)
```

Output

```
{'Nilesh': 41, 'Soumitra': 42, 'Nadeem': 47, 'Rasika': 45, 'Rajashree': 43}
{'Rasika': 45, 'Rajashree': 43, 'Nilesh': 41, 'Soumitra': 42, 'Nadeem': 47}
```

Tips

- From the output it can be observed that the dictionaries are merged in the order listed in the expression.
- As the merging takes place, duplicates get overwritten from left to right. So Rasika : 38 got overwritten with Rasika : 45.

Problem 11.7

For the following dictionary, write a program to report the maximum and minimum salary.

Program

```
d = {
        'anuj' : {'salary' : 10000, 'age' : 20, 'height' : 6},
        'aditya' : {'salary' : 6000, 'age'  : 26, 'height' : 5.6},
        'rahul' :  {'salary' : 7000, 'age'  : 26, 'height' : 5.9}
    }
lst = [ ]
for v in d.values( ) :
```

```
    lst.append(v['salary'])
print(max(lst))
print(min(lst))
```

Output

```
10000
6000
```

Problem 11.8

Suppose a dictionary contains roll numbers and names of students. Write a program to receive the roll number, extract the name corresponding to the roll number and display a message congratulating the student by his name. If the roll number doesn't exist in the dictionary, the message should be 'Congratulations Student!'.

Program

```
students = {554 : 'Ajay', 350: 'Ramesh', 395: 'Rakesh'}
rollno = int(input('Enter roll number: '))
name = students.get(rollno, 'Student')
print(f'Congratulations {name}!')
rollno = int(input('Enter roll number: '))
name = students.get(rollno, 'Student')
print(f'Congratulations {name}!')
```

Output

```
Enter roll number: 350
Congratulations Ramesh!
Enter roll number: 450
Congratulations Student!
```

[A] State whether the following statements are True or False:

(a) Dictionary elements can be accessed using position-based index.

(b) Dictionaries are immutable.

(c) Insertion order is preserved by a dictionary.

(d) The very first key - value pair in a dictionary **d** can be accessed using the expression **d[0]**.

(e) **courses.clear( )** will delete the dictionary object called **courses**.

(f) It is possible to nest dictionaries.

(g) It is possible to hold multiple values against a key in a dictionary.

[B] Attempt the following questions:

(a) Write a program that reads a string from the keyboard and creates dictionary containing frequency of each character occurring in the string. Also print these occurrences in the form of a histogram.

(b) Create a dictionary containing names of students and marks obtained by them in three subjects. Write a program to replace the marks in three subjects with the total in three subjects, and average marks. Also report the topper of the class.

(c) Given the following dictionary:

```
portfolio = {
                'accounts' :  ['SBI', 'IOB'],
                'shares' : '  [HDFC, 'ICICI', 'TM', 'TCS'],
                'ornaments' : ['10 gm gold', '1 kg silver']
            }
```

Write a program to perform the following operations:

- Add a key to portfolio called 'MF' with values 'Relaince' and 'ABSL'.
- Set the value of 'accounts' to a list containing 'Axis' and 'BOB'.
- Sort the items in the list stored under the 'shares' key.
- Delete the list stored under 'ornaments' key.

(d) Create two dictionaries—one containing grocery items and their prices and another containing grocery items and quantity purchased. By using the values from these two dictionaries compute the total bill.

(e) Which functions will you use to fetch all keys, all values and key value pairs from a given dictionary?

(f) Create a dictionary of 10 user names and passwords. Receive the user name and password from keyboard and search for them in the dictionary. Print appropriate message on the screen based on whether a match is found or not.

(g) Given the following dictionary

```
marks = {
            'Subu' : {'Maths' : 88, 'Eng' : 60, 'SSt' : 95},
            'Amol' : {'Maths' : 78, 'Eng' : 68, 'SSt' : 89},
            'Raka' : {'Maths' : 56, 'Eng' : 66, 'SSt' : 77}
        }
```

Write a program to perform the following operations:

- Print marks obtained by Amol in English.
- Set marks obtained by Raka in Maths to 77.
- Sort the dictionary by name.

(h) Create a dictionary which stores the following data:

Interface	**IP Address**	**status**
eth0	1.1.1.1	up
eth1	2.2.2.2	up
wlan0	3.3.3.3	down
wlan1	4.4.4.4	up

Write a program to perform the following operations:

- Find the status of a given interface.
- Find interface and IP of all interfaces which are up.
- Find the total number of interfaces.
- Add two new entries to the dictionary.

(i) Suppose a dictionary contains 5 key-value pairs of name and marks. Write a program to print them from last pair to first pair. Keep deleting every pair printed, such that the end of printing the dictionary falls empty.

[C] Answer the following questions:

(a) What will be the output of the following code snippet?

```
d = { 'Milk' : 1, 'Soap' : 2, 'Towel' : 3, 'Shampoo' : 4, 'Milk' : 7}
print(d[0], d[1], d[2])
```

(b) Which of the following statements are CORRECT?

i. A dictionary will always contain unique keys.
ii. Each key in a dictionary may have multiple values.
iii. If same key is assigned a different value, latest value will prevail.

(c) How will you create an empty list, empty tuple, empty set and empty dictionary?

(d) How will you create a list, tuple, set and dictionary, each containing one element?

(e) Given the following dictionary:

```
d = { 'd1': {'Fruitname' : 'Mango', 'Season' : 'Summer'},
      'd2': {'Fruitname' : 'Orange', 'Season' : 'Winter'}}
```

How will you access and print Mango and Winter?

(f) In the following table check the box if a property is enjoyed by the data types mentioned in columns?

Property	**str**	**list**	**tuple**	**set**	**dict**
Object					
Collection					
Mutable					
Ordered					
Indexed by position					
Indexed by key					
Iterable					
Slicing allowed					
Nesting allowed					
Homogeneous elements					
Heterogeneous elements					

(g) What is the most common usage of the data types mentioned below?

str
list
tuple
set
dict