Python - Dictionaries

Dictionaries are the structures which holds the data in key - value pair format. For example, key can be student id and value can be student object corresponding to that student id. It allows faster access to the object as compared to the elements stored as part of list.

Dictionaries are more general form of the lists. They are more readable and one does not to keep the index in mind while accessing it. For example, a list of months is available, in order to access the October month, one need to keep its index in mind. Also if days of those months also needs to be referred, then another list needs to be maintained.

In [ ]:

```
month_list = ['Jan', 'Feb', 'Mar', 'Apr', "May", 'Jun', 'Jul', "Aug", 'Sep', 'Oct', 'Nov',
month_list[9] # accessing Oct
```

In [ ]:

```
days_list = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
days_list[9] # accessing days of Oct
```

Dictionaries simplifies this job by maintaing 'month names' as keys and days of months as values.

List

Index	Element
0	Jan
1	Feb
2	Mar
3	Apr
4	May
.....	...
11	Dec

Dictionary

Key	Value/Element
'Jan'	31
'Feb'	28
'Mar'	31
'Apr'	30
'May'	31
...	...
'Dec'	31

Creation of Dictionaries

It can be created with curly braces.

In [ ]:

```
empty_dict = {}
empty_dict
```

It can have key value pairs within it. Each key is separated from its value by a colon ":". Commas separate the items, and the whole dictionary is enclosed in curly braces.

In [ ]:

```
my_dict = {'Jan': 31, 'Feb':28, 'March': 31, 'Apr':30} # keys are strings
my_dict
```

Key can be string, numbers or tuples, basically it has to be immutable object.

In [ ]:

```
dict2 = {1:'Jan', 2:'Feb', 3:'March', 4:'April'}  # keys are numbers
dict2
```

In [ ]:

```
dict3 = {(1, 2019) : 'Jan', (2, 2019):'Feb'}  #keys are tuples
dict3
```

Type can be checked by type() method.

In [ ]:

```
type(my_dict)
```

Number of items in dictionary can be found out by len() method.

In [ ]:

```
len(my_dict)
```

dict() function can also be used to create dictionary. List of tuples which actually contains the key value pairs needs to be provided as the input.

In [ ]:

```
d = dict([('A',1), ('B',2), ('C',3)])
d
```

Accessing keys and values

All keys can be accessed via keys() method.

In [ ]:

```
my_dict = {'Jan': 31, 'Feb':28, 'March': 31, 'Apr':30} # keys are strings
my_dict
```

In [ ]:

```
my_dict.keys()  # get all the keys of dictionary object
```

All values can be accessed via values() method.

In [ ]:

```
my_dict.values()  # get all the values of dictionary object
```

All key-value pairs can be accessed by items() method.

In [ ]:

```
my_dict.items()
```

Individual value can be found out by using the key associated with it.

In [ ]:

```
my_dict['Jan']  # access number of days associated with 'Jan'
```

In [ ]:

```
my_dict['Apr']  # access number of days associated with 'Apr'
```

In [ ]:

```
my_dict['xyz']  # access number of days associated with 'xyz' , error as key is not present
```

In [ ]:

```
my_dict.get('xyz') # even if key is not present in the dictionary does not throw error
```

In [ ]:

```
my_dict.get('Apr') # access number of days associated with 'Apr'
```

'in' and 'not in' can be used to determine key is present in dictionary or not

In [ ]:

```
'Apr' in my_dict  # check whether entry with key 'Apr' is present or not
```

In [ ]:

```
'XYZ' in my_dict  # check whether entry with key 'XYZ' is present or not
```

Looping

One can iterate over the entries in Dictionaries.

In [ ]:

```
#Dictionary with Roman numerals
letter_dict = { 'I':1, 'V':5, 'X':10, 'L':50, 'C':100, 'D':500, "M":1000}
letter_dict
```

In [ ]:

```
for key in letter_dict.keys():   # iterate over keys
    print("   " , key)
```

In [ ]:

```
for value in letter_dict.values(): # iterate over values
    print("   " , value)
```

In [ ]:

```
for key in letter_dict.keys():  #access keys and values
    print(key , "   " , letter_dict[key])
```

Operations on dictionary

Entry (i.e. key value pair) can be added to dictionary in following manner :

In [ ]:

```
my_dict['May'] = 31
my_dict
```

Key values can be modified using the assignment operator.

In [ ]:

```
my_dict['May'] # current number of days associated with May is 31
```

In [ ]:

```
my_dict['May'] = 25 # change it to 25
my_dict['May']
```

Entry can be removed from dictionary by using the del().

In [ ]:

```
my_dict  # Key 'May' present in dictionary
```

In [ ]:

```
del(my_dict['May'])   #key 'May' is removed from the dictionary
my_dict
```

Dictionary with complex object as values

The value can be a complex object like list itself.

In [ ]:

```
#Cretae a employee database with emp id as key and list of employees where each value in li
emp_data = { 1:['1','Emp A', 34] , 2:['2','Emp B', 35], 3:['3','Emp C', 36]}

print(emp_data)
```

Add new employee in the database

In [ ]:

```
new_emp = ['4', 'Emp D', 45]  #create employee
emp_data[4] = new_emp  # add employee to the dictionary
emp_data
```

Existing employee details can be modified.

In [ ]:

```
emp_data # look at current employees
```

In [ ]:

```
emp_data[2] = ['2', 'Emp B modified', 67]  # alter the second employee
```

In [ ]:

```
emp_data # look at modified employees
```

Individual employee details can also be accessed.

In [ ]:

```
emp_data[2][0]  # access id of second employee
```

In [ ]:

```
emp_data[2][1]  # name of second employee
```

In [ ]:

```
emp_data[2][2]  # access age of second employee
```

Exercise

Q1. Write a program that asks user to enter the 5 product names and prices. Store all of these in a dictionary with keys are product names and prices are values. Then allow the user to input the product name and print the corresponding price of the product.

In [ ]:

```
#Try it here
```

Q2. Create a dictionary where keys are name of months and days are values.
(a) Ask the user to input a month name and use the dictionary to tell how many days are in that month.
(b) Print out all the keys in alphabetical order.

In [ ]:

```
#Try it here
```

Q3. Ask user 5 times to enter a team name and how many times the team own and how many they lost. Store the information in a dictionary where the keys are team names and values are lists of form [wins, lossess].
(a) Using this dictionary, allow the user to enter a team name and print out the teams winning percentage.
(b) Using this dictionary, create a list whose entries are the number of wins of each team.

In [ ]:

```
#Try it here
```