Module-11

(Hash table for given sentence)

In the previous module you have learned how to give value to the character. Now we will see how to give the value to word.

A hash table is a list of strings in which each item is in the form Name=Value. It can be illustrated as follows:

KEY	Value
Name1	Value1
Name2	Value2
Name3	Value3

There is no strict rule as to when, where, why, or how to use a hash table. Everything depends on the programmer. For example, it can be used to create a list that would replace a 2-dimensional array.

Example for referring a value to a string:

```
>>> string="word"
>>> value=ord(string[0])+ord(string[1])+ord(string[2])+ord(string[3])
>>> print value
444
```

In the above example **word** can refer the value **444**

Another useful data type built into Python is the ***dictionary.***

One of Python's built-in datatypes is the dictionary, which defines one-to-one relationships between keys and values.

Dictionaries:

A dictionary is mutable and is another container type that can store any number of Python objects, including other container types.

Dictionaries consist of pairs (called items) of **keys** and their corresponding **values.**

Python dictionaries are also known as associative arrays or **hash tables**. The general syntax of a dictionary is as follows:

It is best to think of a dictionary as an unordered set of *key: value* pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}. Placing a comma-separated list of key: value pairs within the braces adds initial key: value pairs to the dictionary; this is also the way dictionaries are written on output.

Here is a small example using a dictionary:

Example defining a dictionary

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```

Keys are unique within a dictionary while values may not be.

```
>>> dictionary = {'apple': 1, 'apple': 2, 'apple': 3, 'ball': 4, 'cat': 5}
>>> print dictionary
{'ball': 4, 'apple': 3, 'cat': 5}
>>> dictionary['apple']
3
>>> dictionary.keys()
['ball', 'apple', 'cat']
>>> dictionary.values()
[4, 3, 5]
```

Properties of Dictionary Keys:

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys:

1) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment will prints.
2) Keys must be immutable. Which means you can use strings, numbers, or tuples as dictionary keys but something like ['key'] is not allowed.

Worked out example:

Converting a sentence into dictionary:

```
count = {}
sen=raw_input("Enter a sentence  :")
st=sen.split()
print count
j=0
for s in st:
        count[s]=j
        j=j+1
print count
```

OUTPUT:

Enter a sentence : hi this is dictionary program

{'this': 1, 'program': 4, 'is': 2, 'hi': 0, 'dictionary': 3}

Finding the given word in the given sentence using dictionary:

```
count = {}
sen=raw_input("Enter a sentence  :")
word=raw_input("Enter a word:")
st=sen.split()
j=0
for s in st:
        count[s]=j
        j=j+1
if count.has_key(word):
        print "True"
else: print "False"
```

OUTPUT: Enter a sentence: hi this is dictionary program

Enter a word: hi

True

Problem set:

1) Print all the keys and values from the given dictionary.
2) Take the user input key and its value. Change the first key and value from the given dictionary. Z={‘abc’:1,’bcd’:2, ‘cde’:3}
3) Change dictionary keys to values, values to keys.
   Ex: Z={‘abc’:1,’bcd’:2, ‘cde’:3} changed z={1:’abc’, 2:’bcd’,3:’cde’}
4) Print the true if both dictionaries having same key and values. Take two input dictionaries.
5) Sort the given dictionary. Z={‘apple;1, ‘ant’:2, ‘bat’:3, ‘ball’:4, ‘cat’:5}