

PYTHON PROGRAMMING LANGUAGE

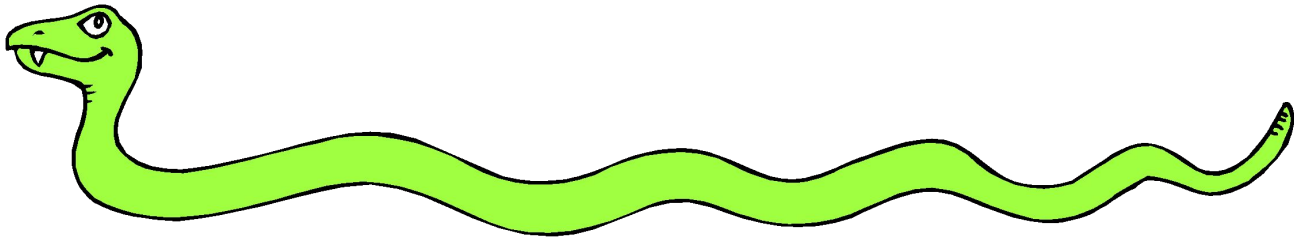
Python Programming Language

PREPARED FOR
Engineering Students
All Engineering College

(PP)
PREPARED BY: MS. SHWETA TIWARI
Published On: April, 2022

FALL SEMESTER, YEAR (IVth, 2nd)
FALL SESSION (2021-22)

Python dicts and sets



Overview

- Python doesn't have traditional vectors and arrays!
- Instead, Python makes heavy use of the **dict** datatype (a hashtable) which can serve as a *sparse array*
 - Efficient traditional arrays are available as modules that interface to C
- A Python **set** is derived from a dict

Dictionaries: A *Mapping* type

- Dictionaries store a *mapping* between a set of keys and a set of values
 - Keys can be any *immutable* type.
 - Values can be any type
 - A single dictionary can store values of different types
- You can define, modify, view, lookup or delete the key-value pairs in the dictionary
- Python's dictionaries are also known as *hash tables* and *associative arrays*

Creating & accessing dictionaries

```
>>> d = { 'user' : 'bozo', 'pswd' : 1234 }
```

```
>>> d[ 'user' ]
```

```
'bozo'
```

```
>>> d[ 'pswd' ]
```

```
1234
```

```
>>> d[ 'bozo' ]
```

```
Traceback (innermost last):
```

```
  File '<interactive input>' line 1, in  
    ?
```

```
KeyError: bozo
```

Updating Dictionaries

```
>>> d = { 'user': 'bozo', 'pswd': 1234 }
>>> d[ 'user' ] = 'clown'
>>> d
{ 'user': 'clown', 'pswd': 1234 }
```

- Keys must be unique
- Assigning to an existing key replaces its value

```
>>> d[ 'id' ] = 45
>>> d
{ 'user': 'clown', 'id': 45, 'pswd': 1234 }
```

- Dictionaries are unordered
 - New entries can appear anywhere in output
- Dictionaries work by *hashing*

Removing dictionary entries

```
>>> d = { 'user': 'bozo', 'p': 1234, 'i': 34 }
>>> del d[ 'user' ]    # Remove one.
>>> d
{ 'p': 1234, 'i': 34 }
>>> d.clear()          # Remove all.
>>> d
{}

>>> a = [1, 2]
>>> del a[1]           # del works on lists, too
>>> a
[1]
```

Useful Accessor Methods

```
>>> d = { 'user': 'bozo', 'p': 1234, 'i': 34 }
```

```
>>> d.keys()    # List of keys, VERY useful  
['user', 'p', 'i']
```

```
>>> d.values()  # List of values  
['bozo', 1234, 34]
```

```
>>> d.items()   # List of item tuples  
[('user', 'bozo'), ('p', 1234), ('i', 34)]
```


A Dictionary Example

Problem: count the frequency of each word in text read from the standard input, print results

Six versions of increasing complexity

- **wf1.py** is a simple start
- **wf2.py** uses a common idiom for default values
- **wf3.py** sorts the output alphabetically
- **wf4.py** downcase and strip punctuation from words and ignore stop words
- **wf5.py** sort output by frequency
- **wf6.py** add command line options: -n, -t, -h

Dictionary example: wf1.py

```
#!/usr/bin/python
import sys
freq = {}    # frequency of words in text
for line in sys.stdin:
    for word in line.split():
        if word in freq:
            freq[word] = 1 + freq[word]
        else:
            freq[word] = 1
print freq
```

Dictionary example wf1.py

```
#!/usr/bin/python
import sys
freq = {}      # frequency of words in text
for line in sys.stdin:
    for word in line.split():
        if word in freq:
            freq[word] = 1 + freq[word]
        else:
            freq[word] = 1
print freq
```

This is a common pattern

Dictionary example wf2.py

```
#!/usr/bin/python
import sys
freq = {}    # frequency of words in text
for line in sys.stdin:
    for word in line.split():
        freq[word] = 1 + freq.get(word, 0)
print freq
```

key

Default value
if not found

Dictionary example wf3.py

```
#!/usr/bin/python
import sys
freq = {}      # frequency of words in text

for line in sys.stdin:
    for word in line.split():
        freq[word] = freq.get(word, 0)

for w in sorted(freq.keys()):
    print w, freq[w]
```

Dictionary example wf4.py

```
#!/usr/bin/python
```

```
import sys
```

```
punctuation =
```

```
    """, '!', '#', '$', '%', '&', '\'', '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<=', '>?', '@', '[', '\\', ']', '^', '_', '`', '{', '|', '}', '~', '"',  
    "
```

```
freq = {}      # frequency of words in text
```

```
stop_words = set()
```

```
for line in open("stop_words.txt"):
```

```
    stop_words.add(line.strip())
```

Dictionary example wf4.py

```
for line in sys.stdin:
    for word in line.split():
        word = word.strip(punct).lower()
        if word not in stop_words:
            freq[word] = freq.get(word, 0) + 1

# print sorted words and their frequencies
for w in sorted(freq.keys()):
    print w, freq[w]
```

Dictionary example wf5.py

```
#!/usr/bin/python
import sys
from operator import itemgetter

...

words = sorted(freq.items(),
               key=itemgetter(1), reverse=True)

for (w,f) in words:
    print w, f
```


Dictionary example wf6.py

```
from optparse import OptionParser
# read command line arguments and process
parser = OptionParser()
parser.add_option('-n', '--number', type="int",
    default=-1, help='number of words to report')
parser.add_option("-t", "--threshold", type="int",
    default=0, help="print if frequency > threshold")
(options, args) = parser.parse_args()
...
# print the top option.number words but only those
# with freq>option.threshold
for (word, freq) in words[:options.number]:
    if freq > options.threshold:
        print freq, word
```

Why must keys be immutable?

- The keys used in a dictionary must be immutable objects?

```
>>> name1, name2 = 'john', ['bob', 'marley']
```

```
>>> fav = name2
```

```
>>> d = {name1: 'alive', name2: 'dead'}
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
TypeError: list objects are unhashable
```

- Why is this?
- Suppose we could index a value for name2
- and then did `fav[0] = "Bobby"`
- Could we find `d[name2]` or `d[fav]` or ...?

defaultdict

```
>>> from collections import defaultdict
>>> kids = defaultdict(list, {'alice': ['mary',
'nick'], 'bob': ['oscar', 'peggy']})
>>> kids['bob']
['oscar', 'peggy']
>>> kids['carol']
[]
>>> age = defaultdict(int)
>>> age['alice'] = 30
>>> age['bob']
0
>>> age['bob'] += 1
>>> age
defaultdict(<type 'int'>, {'bob': 1, 'alice': 30})
```