Online Workshop on ‘How to develop Pythonic coding rather than Python coding – Logic Perspective’

22.7.20 Day 2 session 2

Dr. S.Mohideen Badhusha

Sr.Professor/ CSE department

Alva’s Institute Engineering and Technology

Mijar, Moodbidri, Mangalore

Dictionary and File

Objectives of the Day 2 session 2

To acquire knowledge of Dictionary and File

To comprehend the in-built functions and operations in Dictionary and File

To practice the simple problems in Dictionary and File

Dictionaries: A *Mapping* type

A dictionary in Python is a **collection of items** accessed by a **specific key rather than by index**.

- Dictionaries store a *mapping* between a set of keys and a set of values.
- Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair.

Dictionaries are optimized to retrieve values when the key is known.

  – 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, and delete the key-value pairs in the dictionary.

Creating and 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 entry might appear anywhere in the 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
{}
```

Useful Accessor Methods

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

>>> d.keys()                    # List of keys.
[‘user’, ‘p’, ‘i’]

>>> d.values()                  # List of values.
[‘bozo’, 1234, 34]

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

How to create an empty dictionary?

Creating a dictionary is as simple as placing items inside curly braces {} separated by comma.

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

#list of tuples can be changed as dict using dict()
l=[(1,'apple'), (2,'ball')
my_dict = dict(l)
print(my_dict)

#o/p : {1:'apple', 2:'ball'}
```

```
my_dict = {'name':'Jack', 'age': 26}

print(my_dict['name'])
# Output: Jack

print(my_dict.get('age'))
# Output: 26
```

How to change or add elements in a dictionary?

```
my_dict = {'name':'Jack', 'age': 26}

my_dict['age'] = 27
# update value of key age

print(my_dict)
#Output: {'age': 27, 'name': 'Jack'}

# add an item
my_dict['address'] = 'Downtown'

print(my_dict)
# Output: {'address': 'Downtown', 'age': 27, 'name': 'Jack'}
# create a dictionary
squares = {1:1, 2:4, 3:9, 4:16, 5:25}

# remove a particular item of the key 4
print(squares.pop(4))
# Output: 16

print(squares)
# Output: {1: 1, 2: 4, 3: 9, 5: 25}

# remove last item
print(squares.popitem())
# Output: (5, 25)

print(squares)
# Output: {1: 1, 2: 4, 3: 9}

# delete a particular item
del squares[2]
```

```
print(squares)
# Output: {1: 1, 3: 9}

# remove all items
squares.clear()

print(squares)
# Output: {}

# delete the dictionary itself
del squares

# Throws Error
 print(squares)
```

```
#Initializing the values in dict
marks = {}.fromkeys(['Math','English','Science'], 0)

print(marks)
# Output: {'English': 0, 'Math': 0, 'Science': 0}

squares = {x: x*x for x in range(6)} # using list comprehension
print(squares)
# Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

This code is equivalent to

```
squares = {}
for x in range(6):
   squares[x] = x*x

odd_squares = {x: x*x for x in range(11) if x%2 == 1}

 print(odd_squares)
# Output: {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
```

Iterating Through a Dictionary

Using a for loop we can iterate though each key in a dictionary.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
for i in squares:
    print(squares[i])
```

Dictionary comprehension

```
dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
 # Check for values greater than 2
 d = {k:v for (k,v) in dict1.items() if v>2}
 print(d)
# output
{'e': 5, 'c': 3, 'd': 4}
# multiple conditions
d1 = {k:v for (k,v) in dict1.items() if v>2 if v%2 == 0}
print(d1)
#output
{'d': 4}
```

Files in Python

Open ( ) Function

In order to open a file for writing or use in Python, you must rely on the built-in open () function.

As explained above, open ( ) will return a file object, so it is most commonly used with two arguments. Syntax is

file_object = open("filename", "mode") where file_object is the variable to add the file object.

Mode

Including a mode argument is optional because a default value of 'r' will be assumed if it is omitted. The 'r' value stands for read mode, which is just one of many.

The modes are:

‘r’ – Read mode which is used when the file is only being read

‘w’ – Write mode which is used to edit and write new information to the file (any existing files with the same name will be erased when this mode is activated)

‘a’ – Appending mode, which is used to add new data to the end of the file; that is new information is automatically appended to the end

The modes are:

‘r+’ – Opens a file for both reading and writing.
The file pointer placed at the beginning of the file.

‘w+’ – Opens a file for both writing and reading.
Overwrites the existing file if the file exists.
If the file does not exist, creates a new file for reading and writing.

‘a+’ – Opens a file for both appending and reading.
The file pointer is at the end of the file if the file exists.
The file opens in the append mode.
If the file does not exist,
it creates a new file for reading and writing.

Create a Text file

```
#(filewrite and read.py)
fo = open("foo2.txt", "w")
print ("Name of the file: ", fo.name)
fo.write("Python is a great language Yeah its great!")
# Close opened file
fo.close()

fo = open("foo2.txt", "r")
str1=fo.readline()
print ("Read String is :",str1)

# Close opend file
fo.close()
```

o/p

```
Name of the file:  foo2.txt
Read String is : Python is a wonderful language
Yeah its great!
```

```
# filemultiread.py

fo = open("foo2.txt", "a")
fo.write("Python is very  wonderful language!")
# Close opened file
fo.close()
print ("Entire file is  :")
fo = open("foo2.txt", "r")

for line in fo:
        print(line)

# Close opend file
fo.close()
```

’’’file reading the file line by line by opening and creating object at a stretch’’’

```
# Display line by line information from a file
filename=input("Enter file name: ")
with open (filename,'r') as f:
    for line in f:
        print(line)

# Display word by word  information from a file
filename=input("Enter file name: ")
with open (filename,'r') as f:
    for line in f:
        l=line.split()
        for word in l:
            print(word)
```

```
# Display only first line using readline() from a file
filename=input("Enter file name: ")
with open (filename,'r') as f:
    l=f.readline() # stores 1st line from the file
    print(l)
```

```
# Display line by line  info using readlines() from a file
filename=input("Enter file name: ")
with open (filename,'r') as f:
    l=f.readlines() # stores list of lines in l
    for i in l:
        print(i)
```

```
# Open a file
fo = open("foo.txt", "r+")
str = fo.read(4)
print ("Read String is : ", str)
fo.write(" I am new sentence")
# Check current position
position = fo.tell()
print ("Current file position : ", position)

# Reposition pointer at the beginning once again
position = fo.seek(0, 0)
str = fo.read(30)
print ("Again read String is : ", str)

# Close opened file
fo.close()
```

```
Read String is :  I am
Current file position :  52
Again read String is :  I am new sentenceI am new sent
```