Dictionary & Files in Python

Online One Week Short Term Training Programme on Python Programming-Day 5

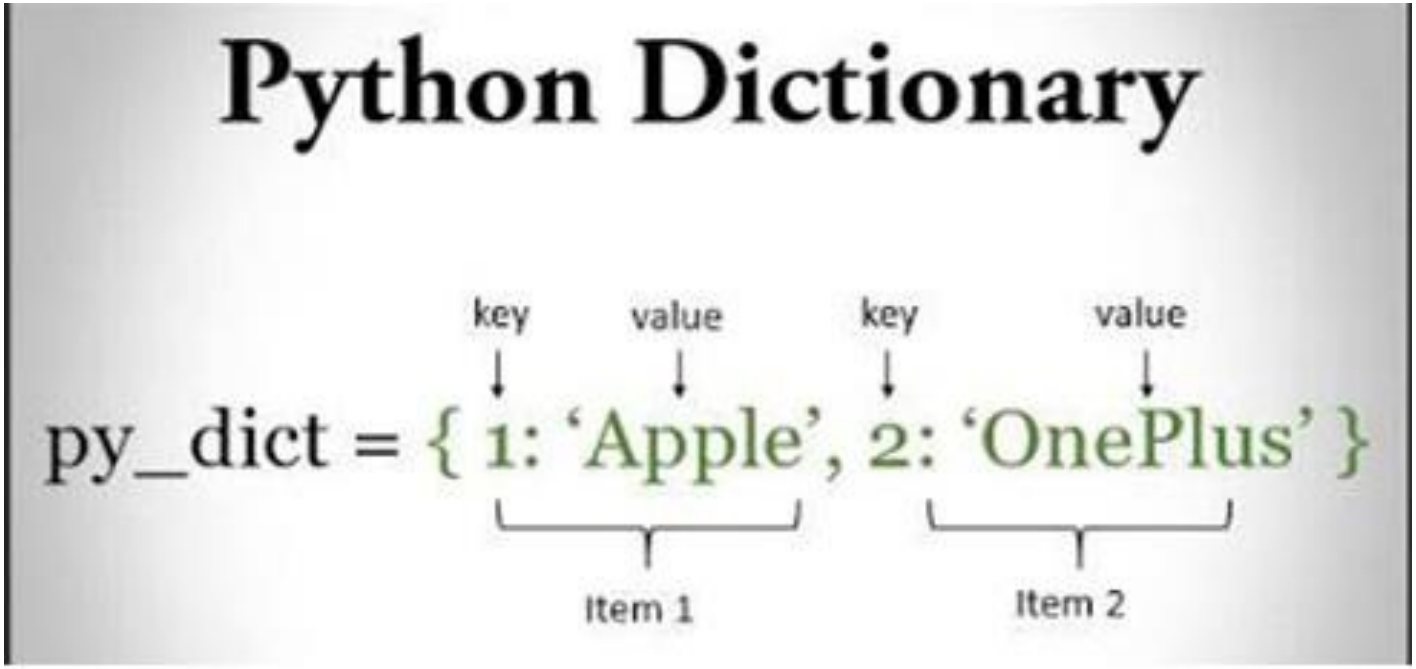
Ms.M.KAMALA MALAR

Assistant Professor, Department of IT,
Dr.Sivanthi Aditanar College of Engineering,
Tiruchendur

Dictionary

- Need : When there is not so much interested in the position of the item or element in the structure but in association of that element with some other element in the structure.
- Definition :Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces.
- Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

Example



Python Dictionary

```
py_dict = { 1: ‘Apple’, 2: ‘OnePlus’ }
```

Creating a Dictionary

```
>>> D1={} #Create empty Dict
>>> D1['Name']='Dr.SACOE‘  #to add 1 field at a time
>>> D1['Year']=1999
>>> D1
{'Name': 'Dr.SACOE', 'Year': 1999}
>>> D2={'Dept':'IT','Year':2001} #all contents
>>> D2
{'Dept': 'IT', 'Year': 2001}
```

Accessing Values in Dictionary

- square brackets along with the key to obtain its value. example −

```
>>>dict1 = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
>>>print ("dict1['Name']: ", dict1['Name'])
>>>print ("dict1['Age']: ", dict1['Age‘])
```

Result:

```
dict['Name']: Zara
 dict['Age']: 7
```

Updating Dictionary

```
>>>dict1 = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
# update existing entry
>>>dict1['Age'] = 8;
# Add new entry
>>>dict1['School'] = "DPS School";
>>>print ("dict1['Age']: ", dict1['Age'] )
>>>print ("dict1['School']: ", dict1['School’])
```

- **Result:**

```
dict1['Age']: 8
dict1['School']: DPS School
```

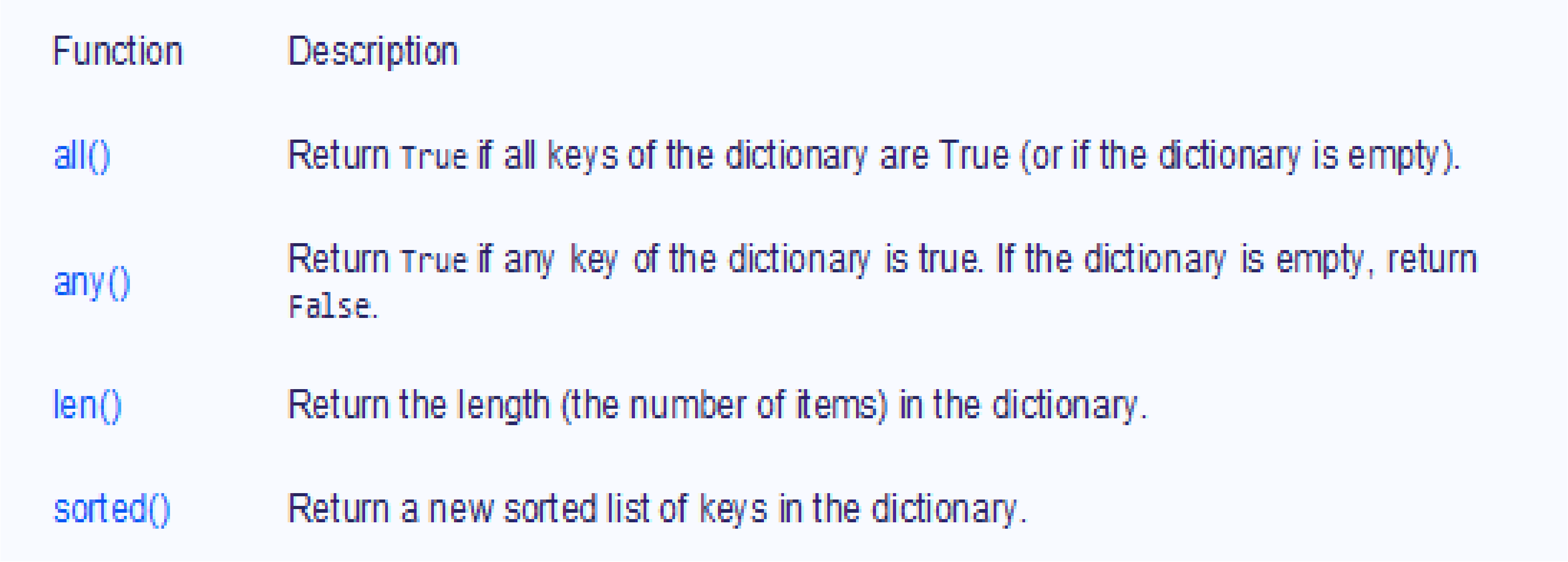
Delete Dictionary Elements

```
dict1 = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
>>>del dict1['Name']
>>>dict1.clear()
>>>del dict1
```

Built-in Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Built-in Dictionary Functions



Function	Description
all()	Return `True` if all keys of the dictionary are True (or if the dictionary is empty).
any()	Return `True` if any key of the dictionary is true. If the dictionary is empty, return `False`.
len()	Return the length (the number of items) in the dictionary.
sorted()	Return a new sorted list of keys in the dictionary.

Length and Sort

```
>>>dict1 = {'Name': 'Zara', 'Age': 7};
>>>print ("Length : %d" % len (dict))
```

- OUTPUT

```
Length : 2
```

```
>>> sorted(dict1.keys())
```

- OUTPUT

```
['Age', 'Name']
```

Copy

```
>>>dict1 = {'Name': 'Zara', 'Age': 7};
>>>dict2 = dict1.copy()
>>>print ("New Dictionary : " ,str(dict2))
```

- OUTPUT

```
New Dictionary : {'Age': 7, 'Name': 'Zara'}
```

Items

```
>>>dict1 = {'Name': 'Zara', 'Age': 7}
>>>print ("Value : " ,dict1.items())
```

OUTPUT

- Value : dict_items([('Name', 'Zara'), ('Age', 7)])

Keys and get(key)

keys()-returns a sequence of keys

```
>>>dict1= {'Name': 'Zara', 'Age': 7}
>>>print ("Value : " ,dict1.keys())
```

- OUTPUT

```
Value :  dict_keys(['Name', 'Age'])
```

get(key)-returns value for a key

```
>>> dict1.get('Name')
```

- OUTPUT

```
'Zara'
```

Update

```
>>>dict1 = {'Name': 'Zara', 'Age': 7}
>>>dict2 = {'Sex': 'female' }
>>>dict1.update(dict2)
>>>print ("Value :" ,dict1)
```

OUTPUT

- Value : {'Age': 7, 'Name': 'Zara', 'Sex': 'female'}

Values

```
>>>dict1 = {'Name': 'Zara', 'Age': 7}
>>>print("Value : " ,dict1.values())
```

- OUTPUT

```
Value : dict_values(['Zara', 7, 'female'])
```

Any

If all keys (not values) are false or the dictionary is empty, any() returns False. If at least one key is true, any() returns True.

```
>>> d={0:1}
>>> any(d)
False
>>> d={'0':1}
>>> any(d)
True
>>> d = {0: 'False', False: 0}
>>> any(d)
False
```

All

In case of dictionaries, if all keys (not values) are true or the dictionary is empty, all() returns True. Else, it returns false for all other cases.

```
>>> s = {1: 'True', False: 0}
>>> all(s)
False
>>> all(s)
True
>>> s = {'0': 'True'}
>>> all(s)
True
```

Sum all the items in a dictionary

-
  ```
  my_dict = {'data1':100,'data2':-54,'data3':247}
  print(sum(my_dict.values()))
  ```

- 293

Get the maximum and minimum value

```
my_dict = {'x':500, 'y':5874, 'z': 560}
>>>key_max=max(my_dict.keys(),key=(lambda k: my_dict[k]))
>>>key_min = min(my_dict.keys(), key=(lambda k: my_dict[k]))
        >>>print('Maximum Value: ',my_dict[key_max])
           >>>print('MinimumValue:',my_dict[key_min])
```

o/p

- Maximum Value: 5874
- Minimum Value: 500

Dictionary Comprehension

- Syntactic construct which creates a dictionary based on existing dictionary.

D={expression for variable in sequence [if condition]}

Example:

```
>>>dict1={x:x**3 for x in range(10) if x%2==1}
```

OUTPUT:

```
{1: 1, 3: 27, 5: 125, 7: 343, 9: 729}
```

Traversing Dictionaries

```
>>>D2={'Dept':'IT','Year':2001}
>>> for k in D2:                # Key
        print(k)
Dept  Year
>>> for k,v in D2.items():      # Key and Value
        print(k,v,sep=':',end=' ')
Dept:IT Year:2001
>>> for v in D2.values():    #Value
        print(v,end=' ')
IT 2001
```

Nested Dictionaries

```
>>> student={'Nila':{'DS':99,'DAA':96,'Java':95},
                 'Sibi':{'DS':95,'DAA':99,'Java':96}}
>>> for k,v in student.items():
        print(k,v)
```

- OUTPUT:

```
Nila {'DS': 99, 'DAA': 96, 'Java': 95}
Sibi {'DS': 95, 'DAA': 99, 'Java': 96}
```

Handling Files in Python

Files - Motivations

- Data stored in the program are temporary; they are lost when the program terminates.
- To permanently store the data created in a program, you need to save them in a file on a disk or other permanent storage.
- The file can be transported and can be read later by other programs.
- There are two types of files:
  - Text
  - Binary

File Modes

S.no	Modes	Description
1	r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
2	rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
3	r+	Opens a file for both reading and writing. The file pointer is placed at the beginning of the file.
4	rb+	Opens a file for both reading and writing in binary format. The file pointer is placed at the beginning of the file.
5	w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
6	wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
7	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.

File Modes –cont…

S.no	Modes	Description
8	wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
9	a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
10	ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
11	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.
12	ab+	Opens a file for both appending and reading in binary format. 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.

File Methods

SNo.	Methods	Description
1	file.close()	Close the file.
2	file.flush()	Flush the internal buffer.
3	file.next()	Returns the next line from the file each time it is being called.
4	file.read([size])	Reads at most size bytes from the file
5	file.readline([size])	Reads one entire line from the file.
6	file.readlines([sizehint])	Reads until EOF using readline() and return a list containing the lines.
7	file.seek(offset[,whence])	Sets the file's current position file.seek(10,0) – 10 bytes from the beginning file.seek(10,1) – 10 bytes from the current position file.seek(10,2) – 10 bytes from the end of the file
8	file.tell()	Returns the file's current position
9	file.truncate([size])	Truncates the file's size. If the optional size argument is present, the file is truncated to (at most) that size.
10	file.write(str)	Writes a string to the file. There is no return value.
11	file.writelines(sequence)	Writes a sequence of strings to the file. The sequence can be list of strings.

1. Opening Files

Prepares the file for reading:

A. Links the file variable with the physical file (references to the file variable are references to the physical file).

B. Positions the file pointer at the start of the file.

Format:

```
<file variable> = open(<file name>, "r")
```

Example:

(Constant file name)

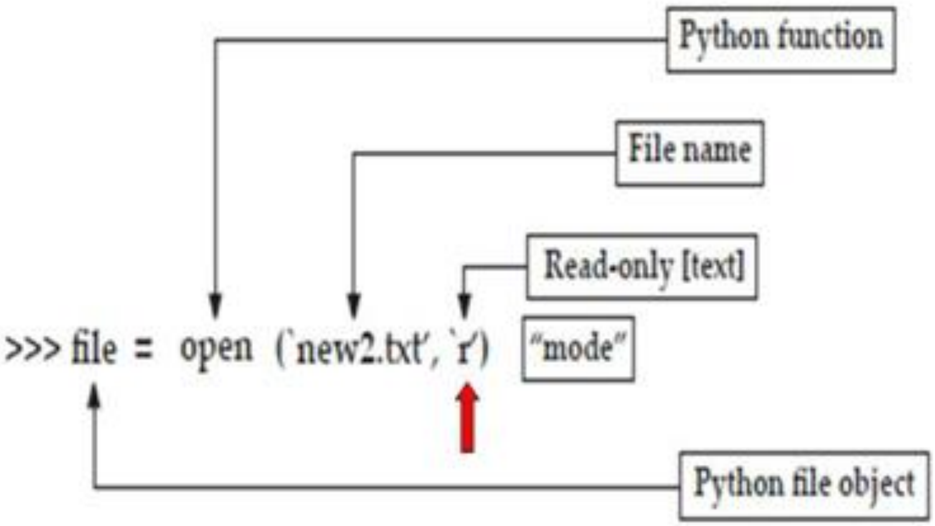
```
inputFile = open("data.txt", "r")
```

OR

(Variable file name: entered by user at runtime)

```
filename = input("Enter name of input file: ")
inputFile = open(filename, "r")
```

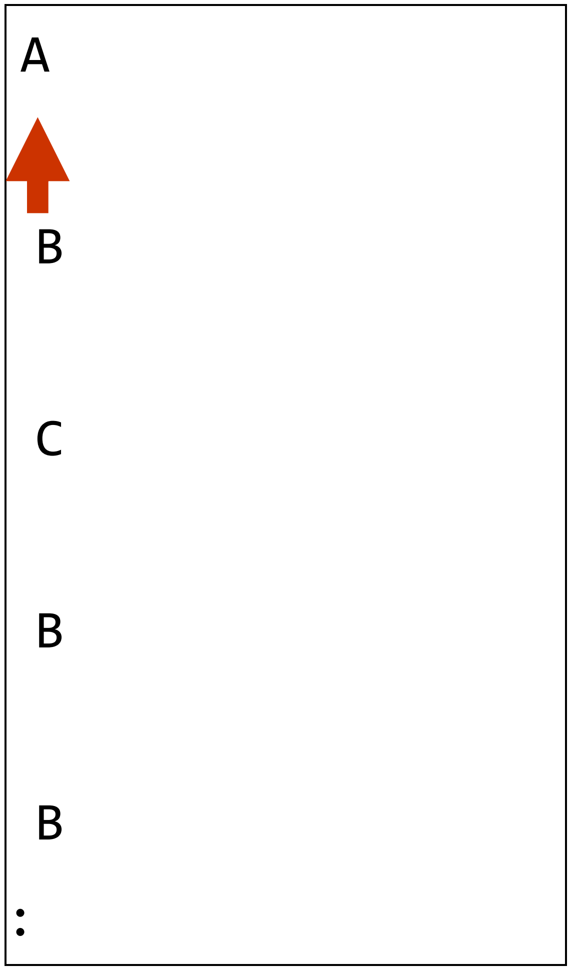
- Reading the entire contents of a file:



```
>>> file = open('new2.txt', 'r')
```

B. Positioning The File Pointer

letters.txt



```
A

B

C

B

B
:
```

2. Reading Information From Files

- Typically reading is done within the body of a loop
- Each execution of the loop will read a line from the file into a string

Format:

```
for <variable to store a string> in <name of file variable>:
    <Do something with the string read from file>
```

Example:

```
for line in inputFile:
    print(line) # Echo file contents back onscreen
```

Closing The File

- Although a file is automatically closed when your program ends it is still a good style to explicitly close your file as soon as the program is done with it.
  - What if the program encounters a runtime error and crashes before it reaches the end? The input file may remain ‘locked’ an inaccessible state because it’s still open.
- **Format:**

```
<name of file variable>.close()
```

- **Example:**

```
inputFile.close()
```

Reading From Files: Putting It All Together

Name of the online example: `readf.py`

Input files: `myfile.txt`

```
inputFileName = input("Enter name of input file: ")
inputFile = open(inputFileName, "r")
print("Opening file", inputFileName, " for reading.")

for line in inputFile:
    print(line)

inputFile.close()
print("Completed reading of file", inputFileName)
```

What you need to write information to a file

1. Open the file and associate the file with a file variable (file is “locked” for writing).
2. A command to write the information.
3. A command to close the file.

1. Opening the file

Format[1]:

```
<name of file variable> = open(<file name>, "w")
```

Example:

(Constant file name)

```
outputFile = open("gpa.txt", "w")
```

(Variable file name: entered by user at runtime)

```
outputFileName = input("Enter the name of the output file
                        to record the GPA's to: ")
outputFile = open(outputFileName, "w")
```

2. Writing to a file

- You can use the ‘`write()`’ function in conjunction with a file variable.
- Note however that this function will ONLY take a string parameter.

Format:

```
outputFile.write(temp)
```

Example:

```
# Assume that temp contains a string of characters.
outputFile.write(temp)
```

Writing To A File: Putting It All Together

- Name of the online example: `grade.py`
- Input file: “`letters.txt`” (sample output file name: `gpa.txt`)

```
inputFileName = input("Enter the name of input file to read the
                       grades from: ")
outputFileName = input("Enter the name of the output file to
                       record the GPA's to: ")

inputFile = open(inputFileName, "r")
outputFile = open(outputFileName, "w")

print("Opening file", inputFileName, " for reading.")
print("Opening file", outputFileName, " for writing.")
gpa = 0
```

Writing To A File: Putting It All Together (2)

```
for line in inputFile:
    if (line[0] == "A"):
        gpa = 4
    elif (line[0] == "B"):
        gpa = 3
    elif (line[0] == "C"):
        gpa = 2
    elif (line[0] == "D"):
        gpa = 1
    elif (line[0] == "F"):
        gpa = 0
    else:
        gpa = -1
    temp = str (gpa)
    temp = temp + '\n'
    print (line[0], '\t', gpa)
    outputFile.write (temp)
```

Writing To A File: Putting It All Together (3)

```
inputFile.close ()
outputFile.close ()
print ("Completed reading of file", inputFileName)
print ("Completed writing to file", outputFileName)
```

Example Problem 1 : Count no of word in a file

```
fw=open(“sample.txt”,”w”)
str=input(“Enter the content to write in to the file sample.txt”)
fw.write(str)
fw.close()
fr=open(“sample.txt”,”r”)
content=fr.read()
words=content.split()
wc=len(words)
print(“totoal no of words”,wc)
```

Example Problem 2 : Copy the content of one file (in.txt) to another (out.txt)

```
fw=open(“in.txt”,”w”)
str=input(“Enter the content to write in to the file in.txt”)
fw.write(str)
fw.close()
with open("in.txt",”r”) as f:
    with open("out.txt", "w") as f1:
        for line in f:
            f1.write(line)
```

Format operator(%)

- **Ex**:

```
print ("My name is %s and weight is %d kg!" % ('Zara', 21))
```

- output: My name is Zara and weight is 21 kg!
- list of complete set of symbols which can be used along with %:
- **Format Symbol**

%c character

%s string

%i signed decimal integer

%d signed decimal integer

%u unsigned decimal integer

%o octal integer

%x hexadecimal integer (lowercase letters)

%X hexadecimal integer (UPPERcase letters)

%e exponential notation (with lowercase 'e')

%E exponential notation (with UPPERcase 'E')

%f floating point real number

%g the shorter of %f and %e

%G the shorter of %f and %E

Thank You