Getting Started with Python Part 2

Dr. Mohammad Abdel-Rahman

CISA4358 Senior Project and Seminar

Tuple

- A Tuple is an immutable list. A tuple can not be changed in any way once it is created
- A Tuple is defined in the same way as a list except that the whole set of elements are enclosed in parentheses instead of square brackets

Tuple operations

- Tuples are immutable – you cannot do something like tup[1] = 5
- Adding elements to a tuple is not possible. Tuples have no append or extend
- However, you can do create a new tuple: tup3 = tup1 + tup2
- Similarly removing elements from a tuple is not allowed. Tuples have no remove or pop method
- Tuple supports Boolean expressions like ‘IN’ like LIST

Tuple method

- len(atuple)
- max(atuple)
- min(atuple)
- tuple(alist)
- list(atuple)

range ()

- range() generates lists containing arithmetic progression
- 3 variations of range() function:
- range(stop) –Starts from 0 till (stop -1)
- range(start,stop) –Ends at (stop -1)
- range(start,stop,step) –Step can not be 0, default is 1

In python 3, range() will not really create a list. Hence xrange is no longer needed.

Sets

- Sets are neither mappings nor sequences; rather, they are ordered collections of unique and immutable objects
- Sets are created by calling the built-in set function
- **Set is useful for sorting and removing repeated items in a list.**

Set - Methods

- Union
- Intersection
- Difference

Set does not support indexing

Get user input

- input()
- You can store the results from them into a variable

print

- “,” linking two strings – a space between them

```
print "Hello","SO!"
```

- “+” means we want to do a string concatenation

```
print “Hello” + “SO!”
```

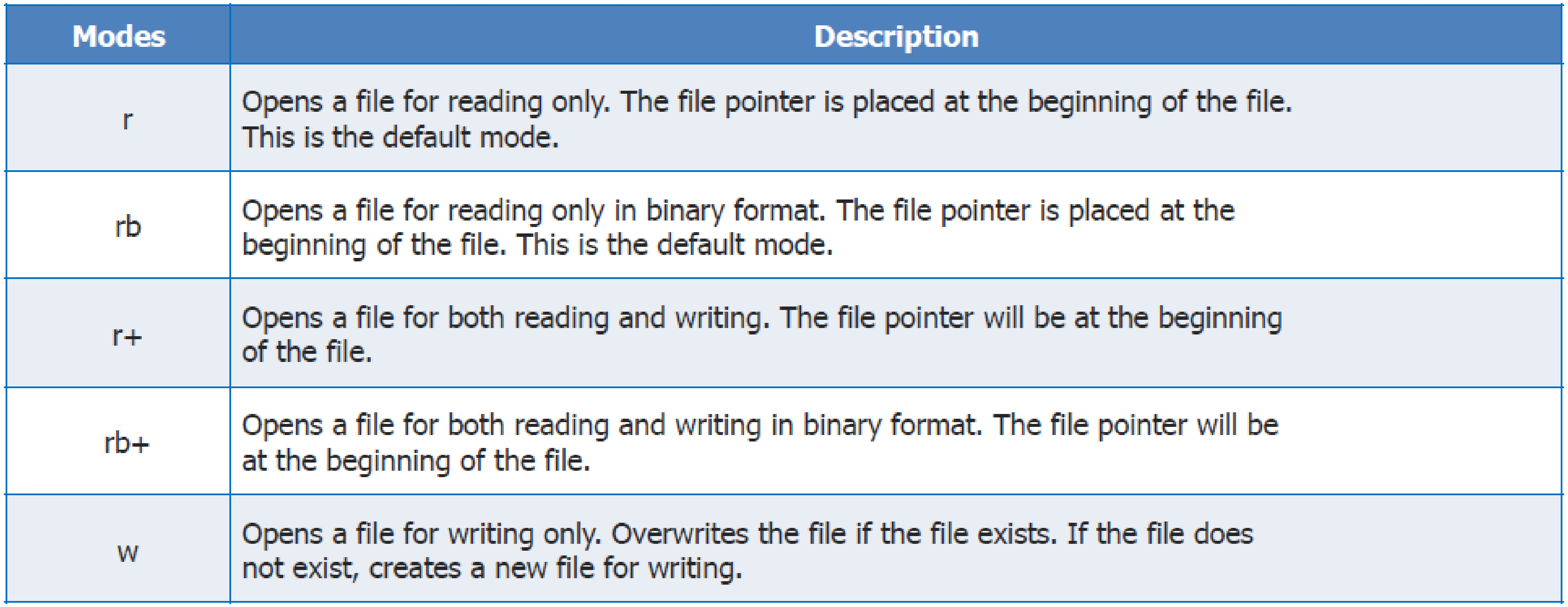
- We can also use % operator.

```
print "%d little pigs come out or I'll %s and %s
and %s" % (3, 'huff', 'puff', 'blow down')
```

Working with Files

- Files are storage compartments on your computer that are managed by operating system
- Python built in open() function creates a Python file object, which serves as a link to a file residing on your machine
- fileobject=open(file_name[,access_mode])

Mode of Opening a File



Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
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.
r+	Opens a file for both reading and writing. The file pointer will be at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer will be at the beginning of the file.
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.

Most of time, we use rU – open as a text file with universal newline interpretation

Mode of Opening a File (cont.)

Modes	Description
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.
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.
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.

How to read file

Read file line by line

```
#Read file line by line
f = open('readfiletest.txt', 'r')
for line in f:## iterates over the lines of the file
    print (line.rstrip())## trailing , so print does not add an end-of-line char
    ## since 'line' already includes the end-of line.
f.close()

#Read file line by line into a list
f = open('readfiletest.txt', 'r')
content = f.readlines()
```

Move the cursor to the beginning of a file

```
f.seek(0)
```

How to write to a file

```
f = open('readfiletest2.txt', 'w')
thelist = [1,2,'this is CISA4358 class',4]
for item in thelist:
    f.write("%s\n" % item)
f.close()
```

Type conversion

function	**Description**
int(x)	Converts x to an integer.
long(x)	Converts x to a long integer.
float(x)	Converts x to a floating-point number.
str(x)	Converts object x to a string representation.
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of(key,value) tuples.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

Exercise: how to define a function

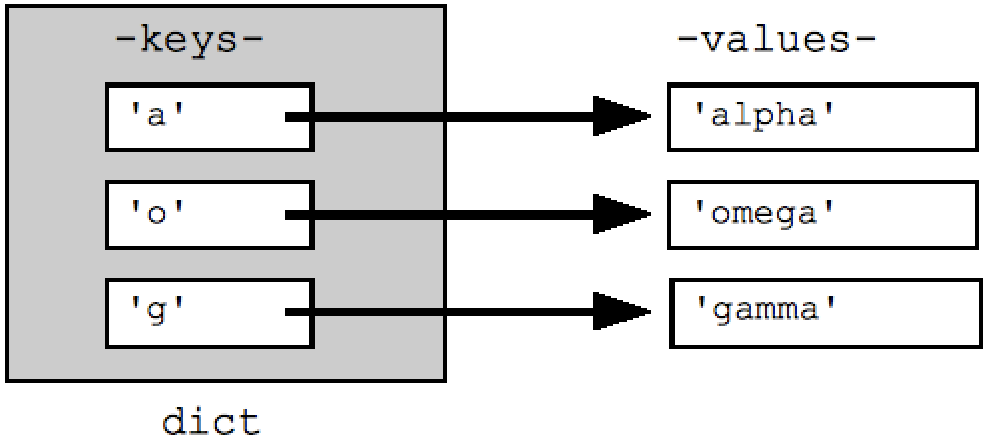
- Write a function named length that receive a string and return its length

```
def length(s):
  return len(s)
print (length("Python"))
```

- Now try to implement the function but without using len method

Dictionary

- Dictionaries are perhaps the most flexible built-in data type in Python
- The chief distinction is that in dictionaries, items are stored and fetched by key, instead of by positional offset



Dictionaries - creation

- Dictionary is written as a series of key:value pairs, separated by commas, enclosed in curly braces { }
- An empty dictionary is an empty set of braces, and dictionaries can be nested by writing one as a value inside another dictionary, or within a list or tuple

```
dic = {}
dic['a'] = 'alpha'
dic['g'] = 'gamma'
dic['o'] = 'omega'
print (dic)  ## {'a': 'alpha', 'o': 'gamma', 'g': 'omega'}

print (dic['a'])     ## Simple lookup, returns 'alpha'
dic['a'] = 6       ## Put new key/value into dict
print('a' in dic)        ## True
##print (dic['z'])                 ## Throws KeyError
if 'z' in dic: print (dic['z'])     ## Avoid KeyError
print (dic.get('z'))  ## None (instead of KeyError)
```