Introduction to PYTHON

Python Input/Output

print(value(s), sep = ' ',end = '\n', file = sys.stdout, flush = False)

By: Atul Kumar Uttam

Python Input

input(prompt = None)

Read a string from standard input. The trailing newline is stripped.

Python Input

input(prompt = None)

**prompt string is optional,
if given, is printed to standard output before reading input.**

```
>>>a = input()
Hi how are you
>>>a
'Hi how are you'
>>>a = input("Enter a value: ")
Enter a value:
```

Example of input()

```
>>>a = input()
Hi how are you
>>>a
‘Hi how are you’
>>>a = input(“Enter a value: ”)
Enter a value: Hello world
>>>a
‘Hello world’
```

Python Output

print(value(s), sep = ' ',end = '\n', file = sys.stdout, flush = False)

Optional Keyword arguments

It can take any number of values

Python Output

print(value(s), sep = ' ', end = '\n', file = sys.stdout, flush = False **)**

```
>>>a, b, c = 10, 20.5, "Hello"
>>>print(a, b, c)
10   20.5  Hello
```

Python Output

print(value(s), **sep = ' '**, end = '\n', file = sys.stdout, flush = False **)**

Separator is used between the values.
It defaults into a space character.

Python Output

print(value(s), **sep = ' '**, end = '\n', file = sys.stdout, flush = False **)**

>>>a, b, c = 10**,** 20.5**,** “Hello”

>>>print(a, b, c, sep = “@”)

10@20.5@Hello

Python Output

print(value(s), sep = ' ',**end = '\n'**, file = sys.stdout, flush = False **)**

After all values are printed, end is printed.
It defaults into a new line.

Python Output

print(value(s), sep = ' ', **end = '\n'**, file = sys.stdout, flush = False **)**

Hello.py

```
print("hello")
print("world")
print("of python")
```

print("hello", end ="\n")

print("world", end ="\n")

print("of python", end ="\n")

Output:

hello
world
of python

Python Output

print(value(s), sep = ' ', **end = ‘ '**, file = sys.stdout, flush = False **)**

Hello.py

```
print(“hello”, end = “  ”)
print(“world”, end = “  ”)
print(“of python”)
```

Output:

hello world of python

Python Output

print(value(s), sep = ' ',end = '\n', **file = sys.stdout**, flush = False **)**

The **file is the object** where the values are printed and its default value is **sys.stdout** i.e. **console screen**

Python Output

print(value(s), sep = ' ',end = '\n', **file = sys.stdout**, flush = False **)**

>>>print(“Hello World!!**”)**

Hello World!!

Python Output

print(value(s), sep = ' ',end = '\n', **file = sys.stdout**, flush = False **)**

```
>>>A = open(“Data.txt", "w")
>>>print(“Hello World", file = A)
>>>A.close()
```

Data.txt

Python Output

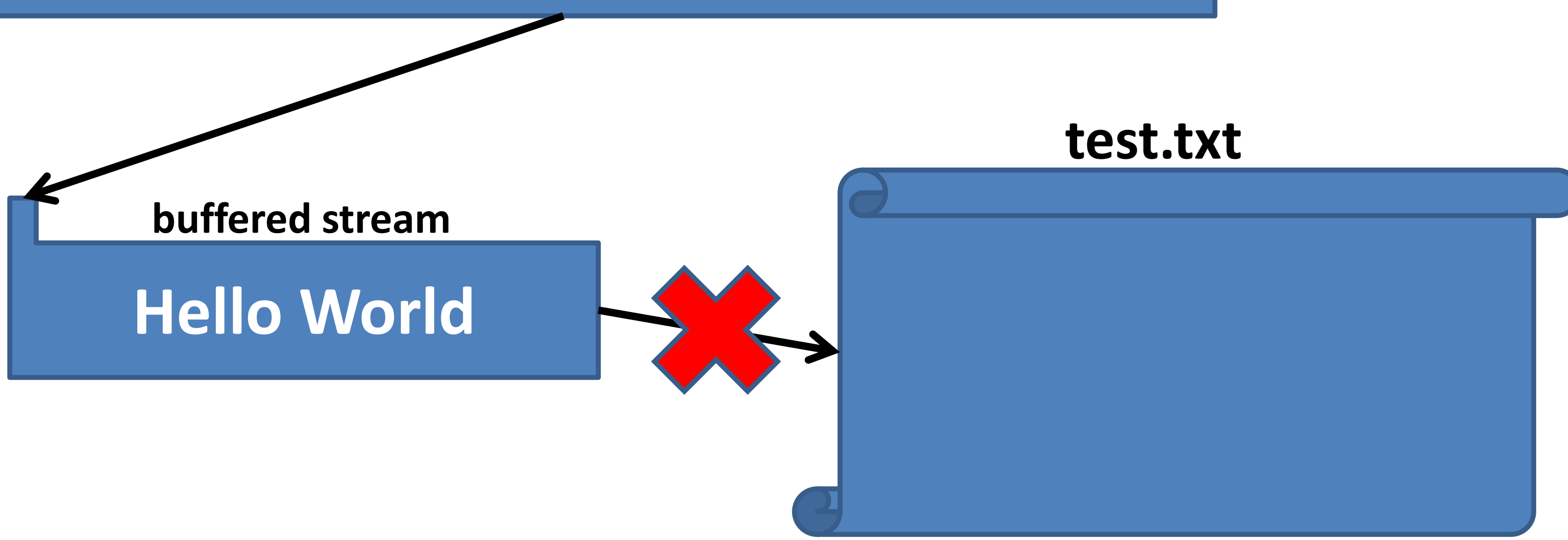
print(value(s), sep = ' ',end = '\n', file = sys.stdout, **flush = False)**

The OUTPUT stream is not forcibly flushed.

Python Output

print(value(s), sep = ' ',end = '\n', file = sys.stdout, **flush = False)**

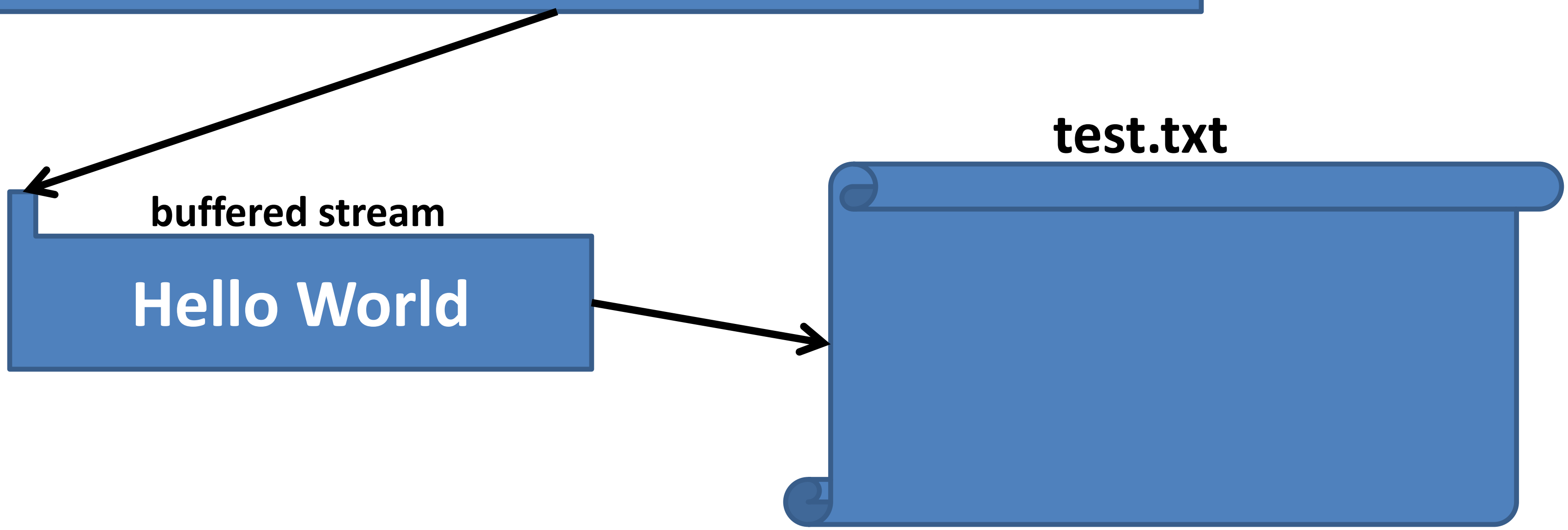
```
>>>A=open("test.txt","w")
>>>print("Hello World", file=A, flush=False)
```



Python Output

print(value(s), sep = ' ',end = '\n', file = sys.stdout, **flush = False)**

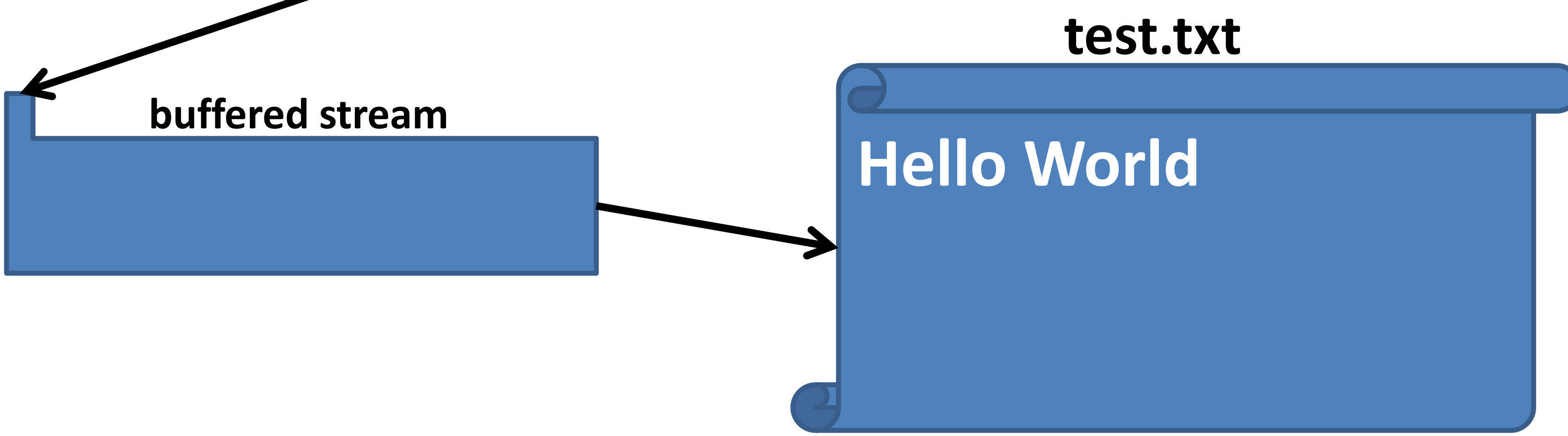
```
>>>A=open("test.txt","w")
>>>print("Hello World", file=A, flush=True)
```



Python Output

print(value(s), sep = ' ',end = '\n', file = sys.stdout, **flush = False)**

```
>>>A=open("test.txt","w")
>>>print("Hello World", file=A, flush=True)
```



python

Introduction to PYTHON

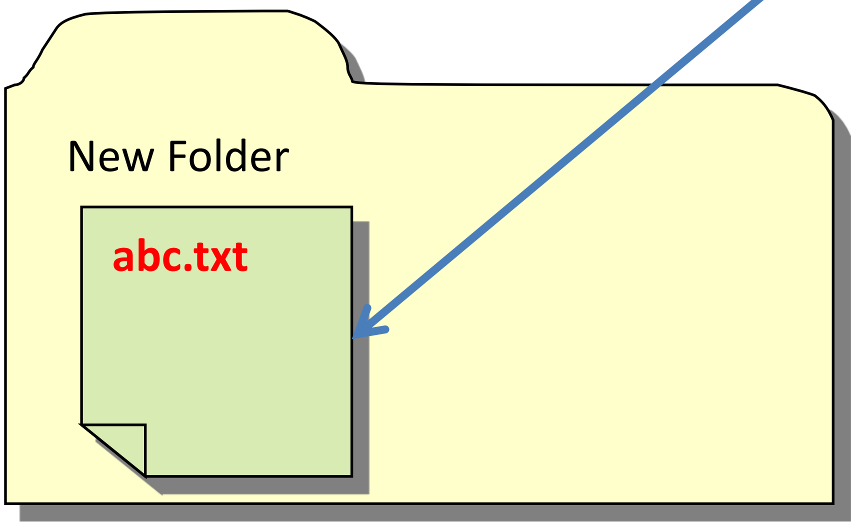
File Handling
Part-1

(Steps in file handling, Access Modes, File Object's attribute)

By: Atul Kumar Uttam

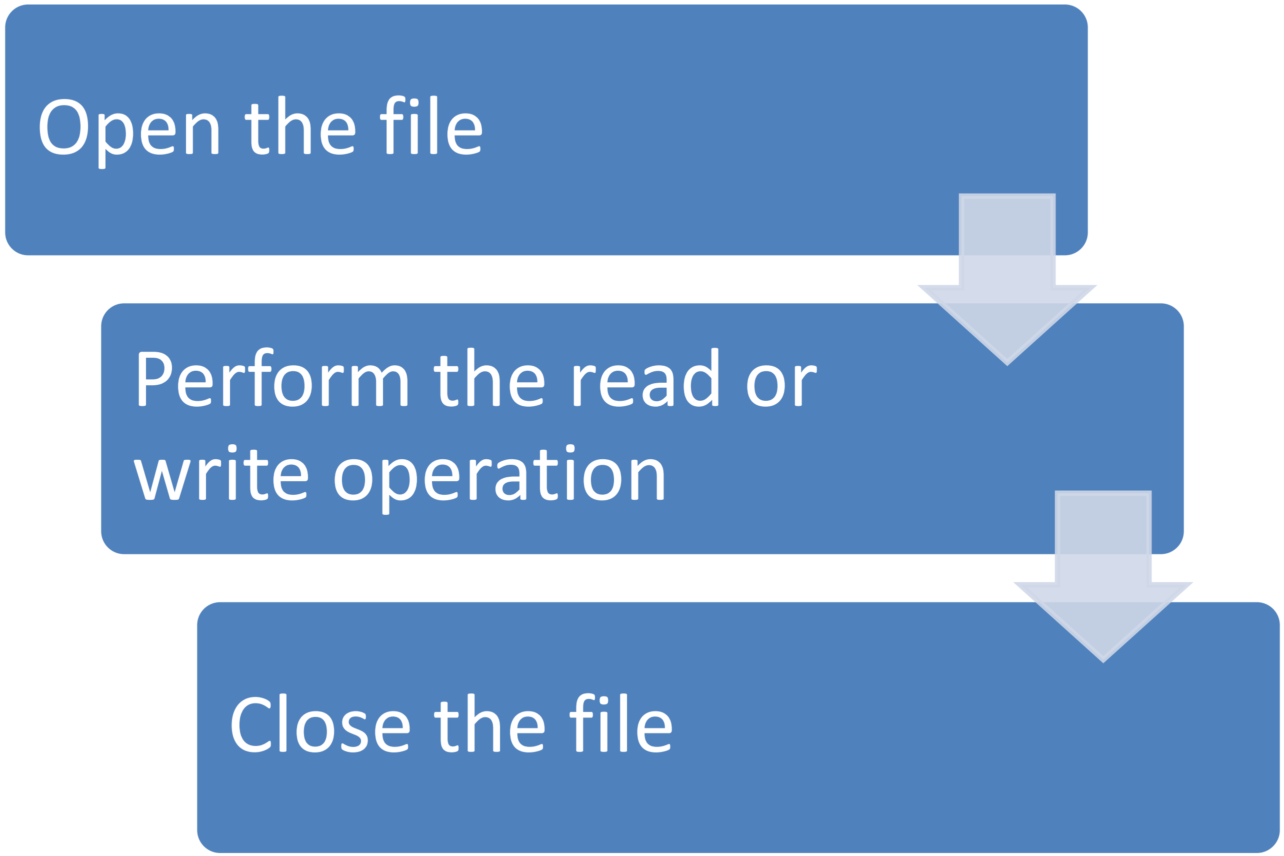
File Handling

File is a named location on disk to store related information permanently



D:\New folder\abc.txt

Steps in File Handling



Step 1. open the file

- Before you can read or write a file, you have to open it using Python's built-in open() function.
- This function returns a **file** object
- If the file cannot be opened, an Error is raised

File_object = **open(** file_name [, access_mode] **)**

A string value that contains the absolute / relative path the file

Step 1. open the file

- Before you can read or write a file, you have to open it using Python's built-in open() function.
- This function returns a **file** object
- If the file cannot be opened, an Error is raised

File_object = **open(** file_name [, access_mode] **)**

the mode in which the file has to be opened, i.e., read, write, append

Access modes of a file

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open for updating (reading and writing)

Access modes of a file

<table>
<tr><th>Character</th><th>Meaning</th></tr>
<tr><td>'r'</td><td>open for reading (default)</td></tr>
<tr><td colspan="2">Opens a file for writing only.

Throws FileNotFoundError if file does not exists.

Puts the file pointer at the beginning of the file</td></tr>
</table>

Access modes of a file

Character	Meaning
'w'	**open for writing, truncating the file first**
	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.

Access modes of a file

Character	Meaning
'x'	open for exclusive creation, failing if the file already exists

Open a new file for writing

If the file exist already then it will give error.

Access modes of a file

Character	Meaning
'a'	**open for writing, appending to the end of the file if it exists**

Opens a file for appending.

The file pointer is at the end of the file if the file exists.

If the file does not exist, it creates a new file for writing

To open a file

open file in current directory

```
>>> f = open("test.txt")
```

Relative Path

Absolute Path

specifying full path

```
>>> f = open("C:/Python3/README.txt")
```

The file Object’s Attributes

- Provide various information related to that file.

>>>file = open(“……………………”, “..”)

- **file.closed** Returns true if file is closed, false otherwise.
- **file.mode** Returns access mode with which file was opened.
- **file.name** Returns name of the file.

The file Object's Attributes Example

>>>**f** = **open(“**test.txt**", **"w**")**

>>>print ("Name of the file: ", **f.name**)

Name of the file: **test.txt**

>>>print ("Closed or not : ", **f.closed**)

Closed or not : **False**

>>>print ("Opening mode : ", **f.mode**)

Opening mode : **w**

Introduction to PYTHON

File Handling
Part-2

(Reading & Writing into Files write(), read(), readline(), readlines())

By: Atul Kumar Uttam

Step 2. Writing in Files

write()

- The **write()** method writes any string to an open file.
- returns the number of characters written.

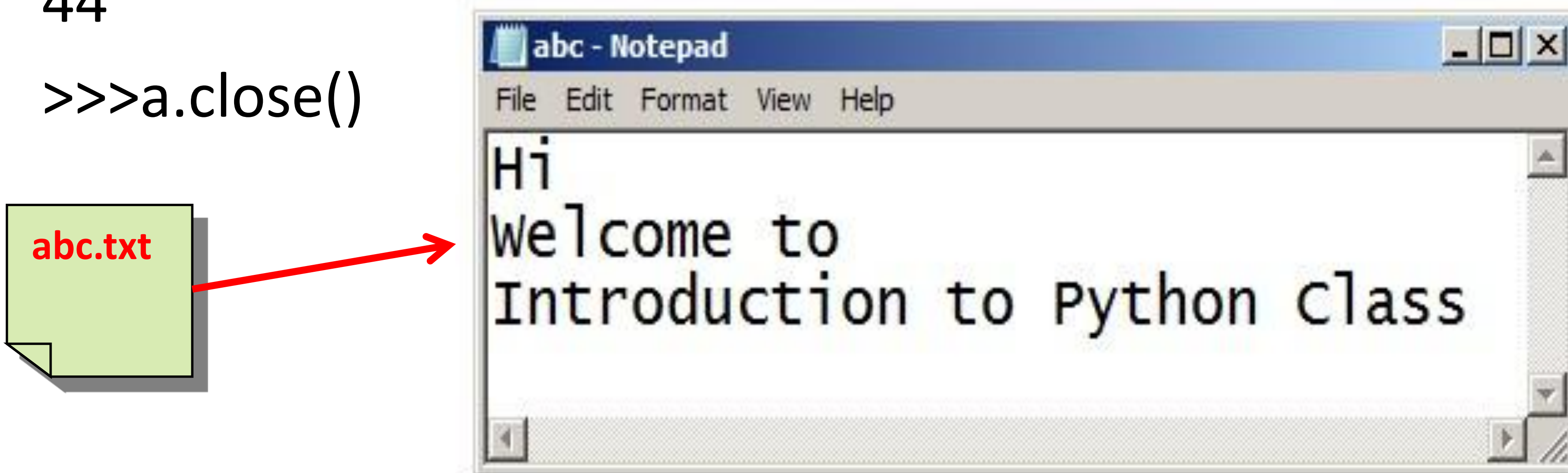
Syntax

fileObj**.write(**string**)**

string to be written into the opened file

write()

```
>>>a=open("abc.txt","w")
>>>a.write("""Hi \nWelcome to
   Introduction to Python Class """)
44
>>>a.close()
```



Reading from files

read() returns the read string

Syntax

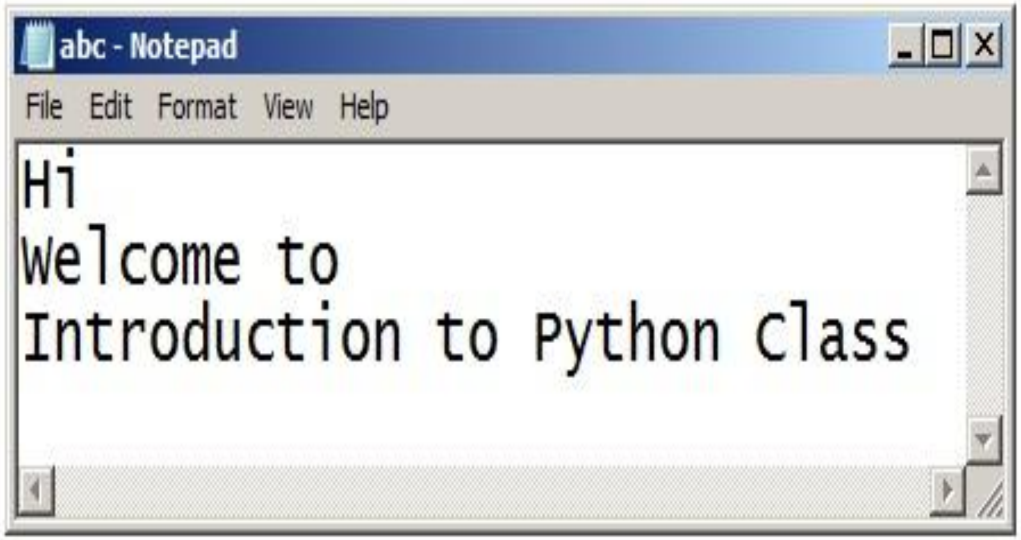
>>>fileObj.read([count])

starts reading from the beginning of the file

the number of bytes to be read / if not given then reads the complete file

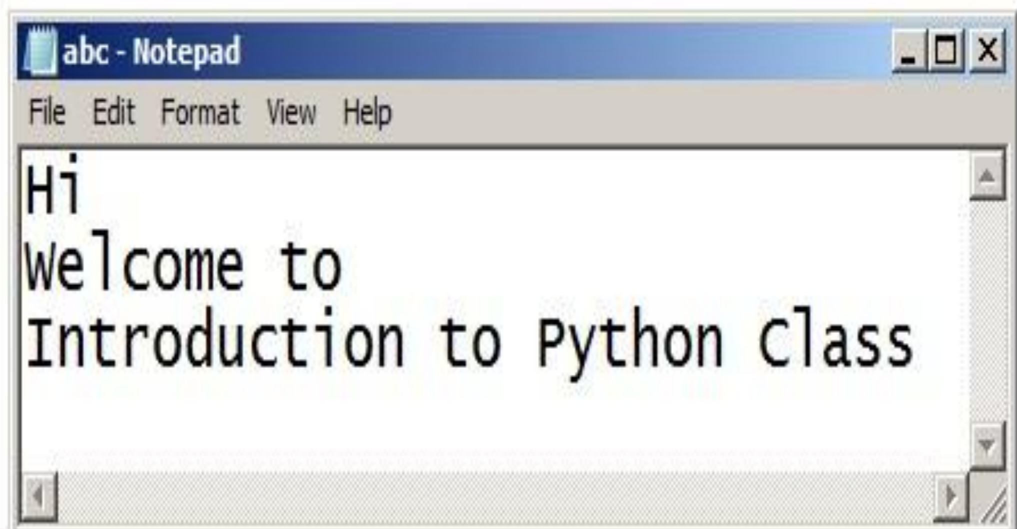
Reading from files

```
>>>a=open("abc.txt","r")
>>>a.read(10)
'Hi\nWelcome'
>>>a.close()
```



Reading from files

```
>>>a=open("abc.txt","r")
>>>a.read()
'Hi\nWelcome to\nIntroduction to Python Class ‘
>>>a.close()
```



Reading from files

readline() read one line each time from the file, including the newline character.

>>>a = open("abc.txt","r"**)**

>>>a.readline()

'Hi\n‘

>>>a.readline()

'Welcome to\n‘

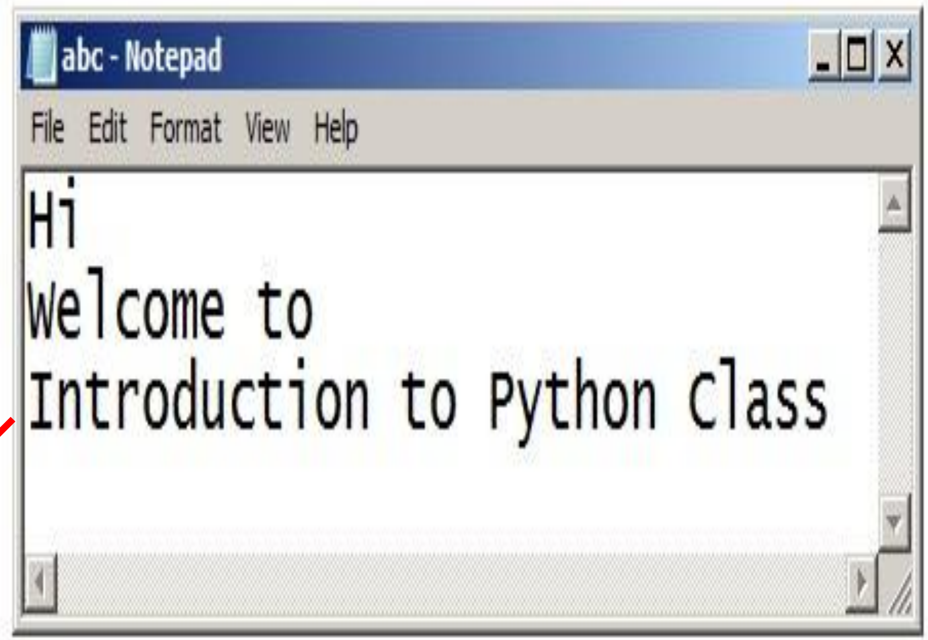
>>>a.readline()

'Introduction to Python Class‘

>>>a.readline()

“

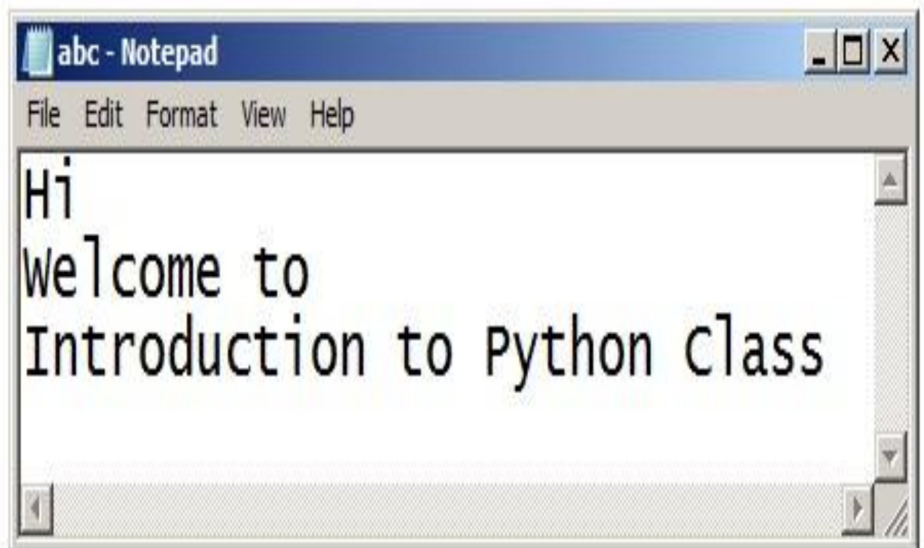
>>>a.close()



Reading from files

readlines() read all the lines of a file in a list

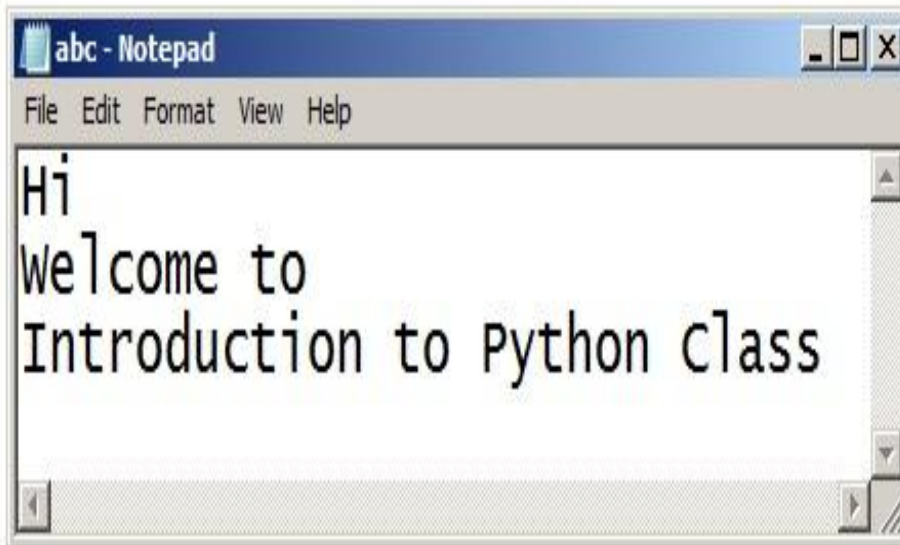
```
>>>a=open("abc.txt","r")
>>>a.readlines()
['Hi\n', 'Welcome to\n', 'Introduction to Python Class']
>>>a.close()
```



Reading from files

- We can read a file line-by-line using a for loop.
- This is both efficient and fast.

```
>>>a=open("abc.txt","r")
>>>for line in a:
        print(line)
```



Hi

Welcome to

Introduction to Python Class

Step 3. Closing the file

- It flushes any unwritten information and closes the file object, after which no more writing can be done.
- Python automatically closes a file when the reference object of a file is reassigned to another file.
- It is a good practice to use the close() method to close a file.

```
>>>fileObject.close()
```

python

Introduction to PYTHON

File Handling
Part-3

(File Position tell() & seek())

By: Atul Kumar Uttam

File Positions

tell() returns an integer

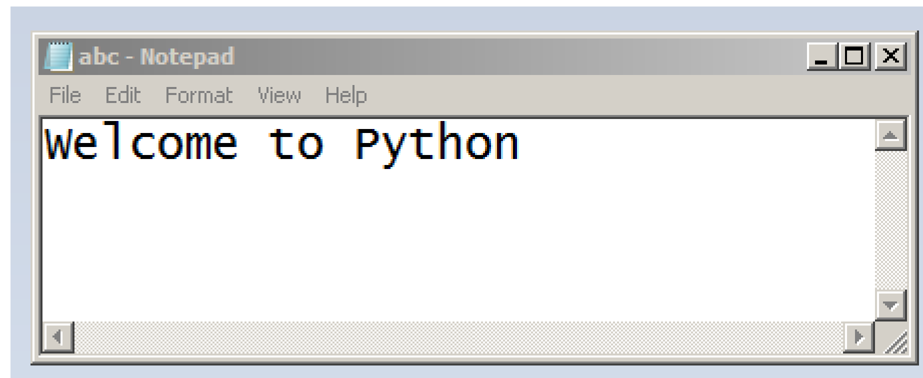
- Gives the file object's current position in the file as a number

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

W e l c o m e   t o   P y t h o n

- a=open("abc.txt","r")
- a.tell()

0



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	W	e	l	c	o	m	e		t	o		P	y	t	h	o	n

- a=open("abc.txt","r")
- a.tell()

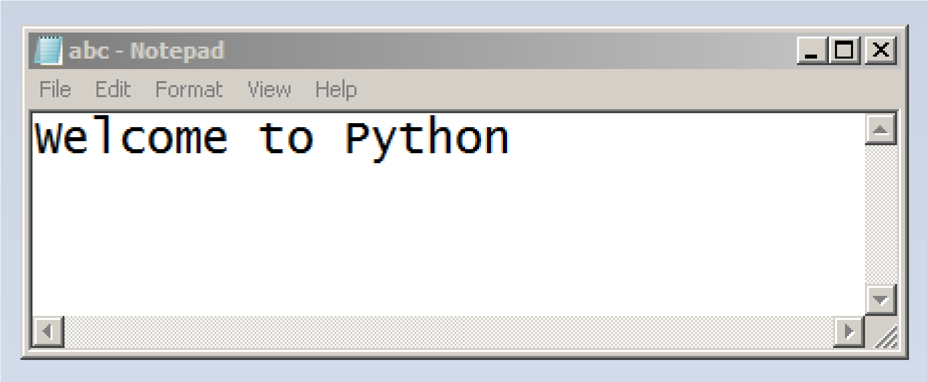
0

- a.read(3)

‘wel’

- a.tell()

3



File Positions

seek(offset[, from])

File Positions

- To change the file object’s position

seek(offset**[,** from**])**

The number of positions, file pointer will move

File Positions

- To change the file object’s position

seek(offset**[,** from**])**

Optional, defines the point of reference

File Positions

- To change the file object's position

seek(offset**[,** from**])**

<u>**Three possible values**</u>

0: Beginning of the file

1: Current position

2: End of the file

File Positions

- To change the file object's position

seek(offset**[,** from**])**

1: Current position
2: End of the file

These two works only in binary mode files

File Positions

- To change the file object's position

seek(offset**[,** from**])**

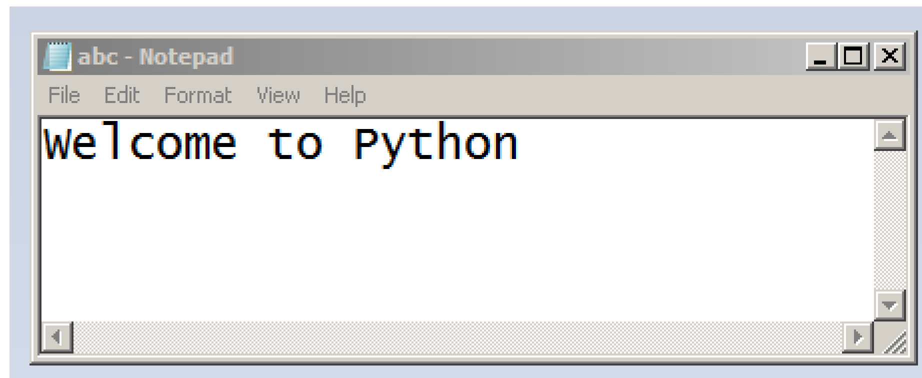
fileobj.seek(0, 0)
go to the beginning of the file.

fileobj.seek(0, 2)
go to the end of the file.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	W	e	l	c	o	m	e		t	o		P	y	t	h	o	n

- a=open("abc.txt","rb")
- a.tell()

0

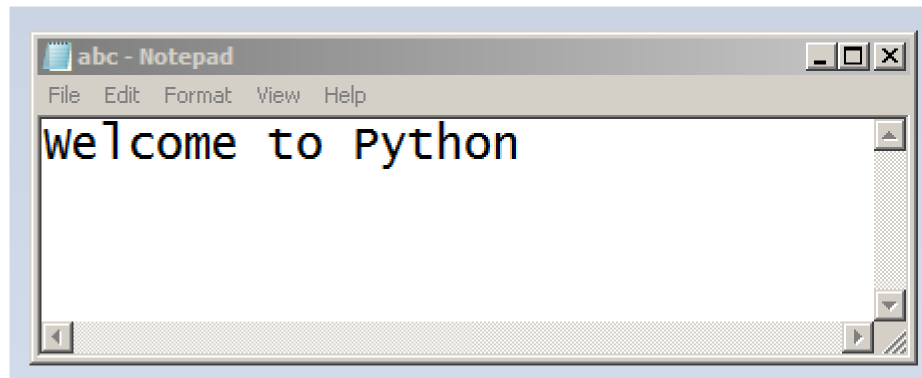


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

W e l c o m e t o P y t h o n

- a.read()
- b'Welcome to Python'
- a.tell()

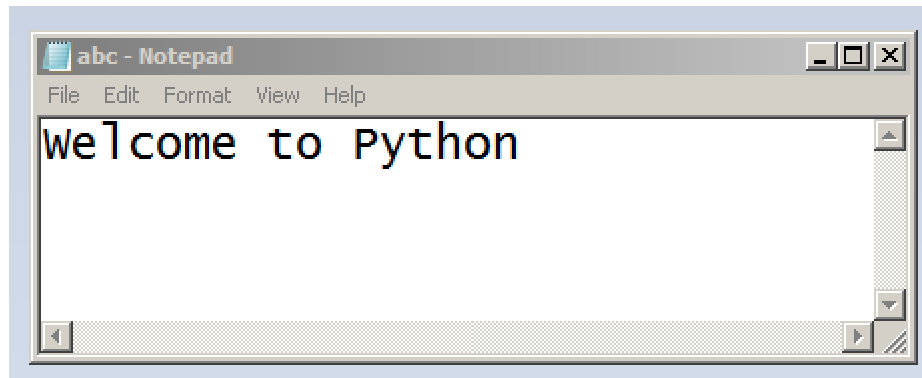
17



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	W	**e**	**l**	**c**	**o**	**m**	**e**		**t**	**o**		**P**	**y**	**t**	**h**	**o**	**n**

- a.seek(-10,2)

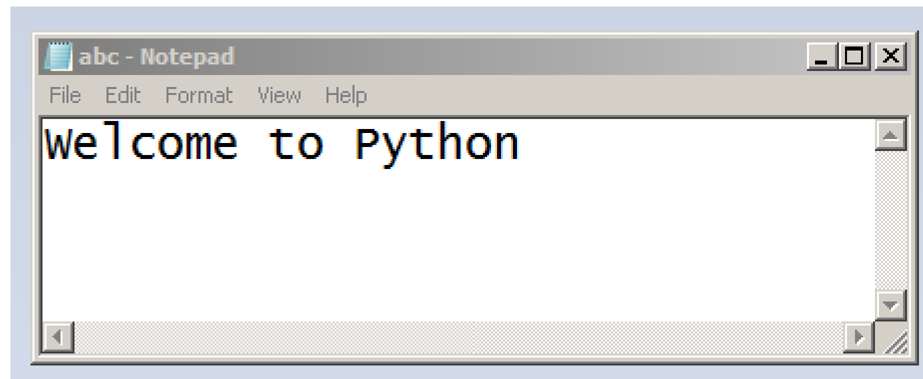
7



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	W	**e**	**l**	**c**	**o**	**m**	**e**		**t**	**o**		**P**	**y**	**t**	**h**	**o**	**n**

- a.read()

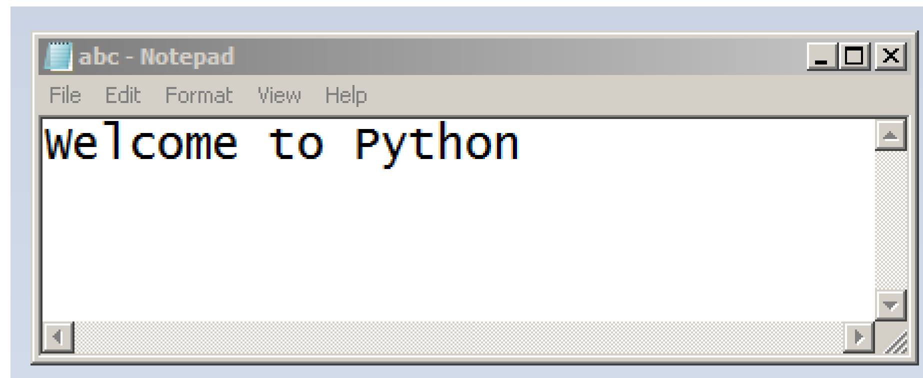
b' to Python‘



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	W	**e**	**l**	**c**	**o**	**m**	**e**		**t**	**o**		**P**	**y**	**t**	**h**	**o**	**n**

- a.seek(0,2)

17

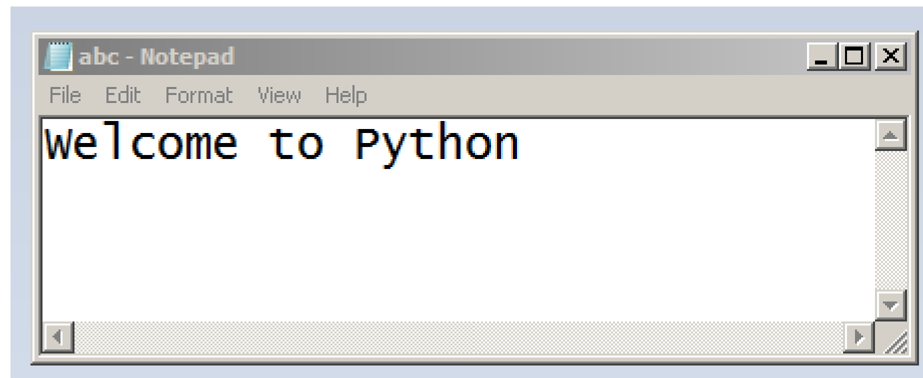


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

W e l c o m e   t o   P y t h o n

- a.seek(0,0)

0

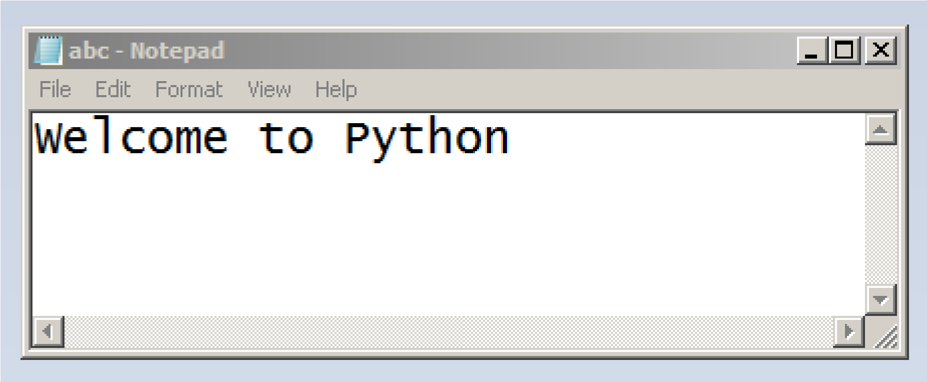


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

W e l c o m e t o P y t h o n

- a.read(2)

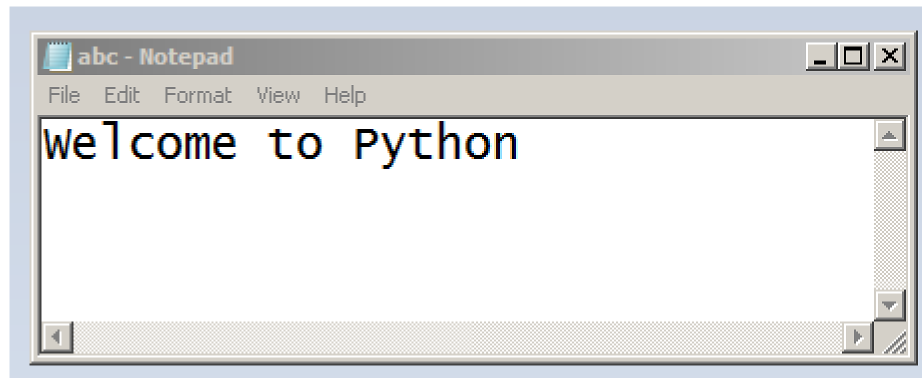
b'We'



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	W	**e**	**l**	**c**	**o**	**m**	**e**		**t**	**o**		**P**	**y**	**t**	**h**	**o**	**n**

- a.seek(5,1)

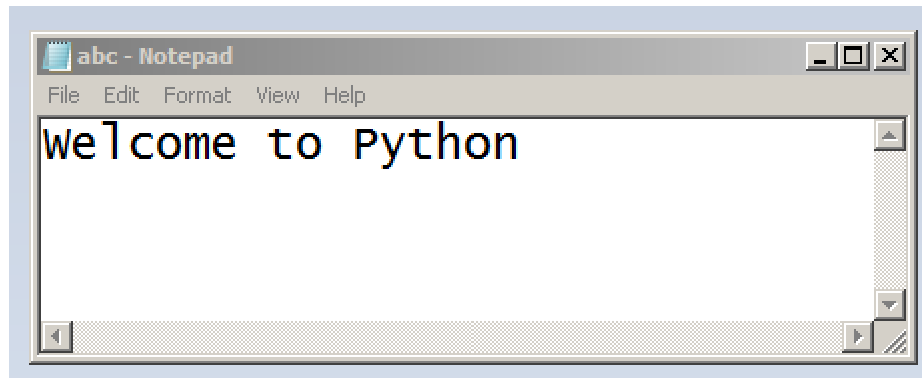
7



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	W	e	l	c	o	m	e		t	o		P	y	t	h	o	n

- a.read()

b' to Python'



Introduction to PYTHON

File Handling Part-4

(with statement)

By: Atul Kumar Uttam

Open a file using ***with*** statement

- No need to close the file explicitly.
- File will be closed as control come out of ***with*** statement block

```
>>> with open(‘filename.txt') as fobj:
        for line in fobj:
            print (line)
```

```
>>>with open("test.txt",'w') as f:
        f.write("my first file\n")
        f.write("This file\n\n")
        f.write("contains three lines\n")
```

File Handling example

```
with open("name.txt","r") as namefile:
    with open("Message.txt","r") as messagefile:
        text = messagefile.read()
        for name in namefile:
            mail="Dear Sir/Madam" + name + text
            with open(name.strip() + ".txt", 'w') as mailfile:
                mailfile.write(mail)
```