SWINBURNE UNIVERSITY OF TECHNOLOGY
SARAWAK CAMPUS

# Lecture 9

## Reading and writing files in Python

KNOWING

# Python handles text files

- Python is a great tool for processing data.
- It is likely that any program you write will involve reading, writing, or manipulating data.
- For this reason, it is especially useful to know how to handle different file formats, which store different type of data.

# File processing

- The process of *opening* a file involves associating a file on disk with an object in memory.
- We can manipulate the file by manipulating this object.
  - Read from the file
  - Write to the file

# Overview

- The first thing you'll need to do is use Python's built-in ***open*** function to get a ***file object***.
- The ***open*** function opens a file. It's simple.
- When you use the ***open*** function, it returns something called a ***file object***. ***File objects*** contain methods and attributes that can be used to collect information about the file you opened. They can also be used to manipulate said file.

# Overview

- For example, the ***mode*** attribute of a ***file object*** tells you which mode a file was opened in. And the ***name*** attribute tells you the name of the file that the ***file object*** has opened.
- You must understand that a ***file*** and ***file object*** are two wholly separate – yet related – things.

# Opening a file in Python

- In order to open a file for writing or use in Python, you must rely on the built-in open() function.
- Open() will return a file object, so it is most commonly used with two arguments.
- An argument is nothing more than a value that has been provided to a function which is relayed when you call it.

# Example: Opening a file in Python

File Edit Format Run Options Window Help

```
def read_file(fname):
    txt = open(fname)
    print(txt.read())

read_file('test.txt')
```

File Edit Format Run Options Window Help

```
f = open('test.txt', 'r')
print (f.read(20)) #read the first line, first 20 characters including spacing
```

File Edit Format Run Options Window Help

```
f = open('test.txt', 'r')
print (f.readline()) #read the first paragraph
```

KNOW ING

# 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.

# Mode

- The modes are:
  - ‘r’ – read mode which is used when 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 amended to the end
  - ‘r+’ – special read and write mode, which is used to handle both actions when working with a file

# Example: Opening a file in Python

```
['An interpreted language, Python has a design philosophy which emphasizes code
readability (notably using whitespace indentation to delimit code blocks rather
than curly brackets or keywords), and a syntax which allows programmers to expre
ss concepts in fewer lines of code than possible in languages such as C++ or Jav
a.\n', '\n', 'Python is a widely used high-level programming language for genera
l-purpose programming, created by Guido van Rossum and first released in 1991. A
n interpreted language, Python has a design philosophy which emphasizes code rea
dability (notably using whitespace indentation to delimit code blocks rather tha
n curly brackets or keywords), and a syntax which allows programmers to express
concepts in fewer lines of code than possible in languages such as C++ or Java.
The language provides constructs intended to enable writing clear programs on bo
th a small and large scale.\n', '\n', 'Python features a dynamic type system and
 automatic memory management and supports multiple programming paradigms, includ
ing object-oriented, imperative, functional programming, and procedural styles.
It has a large and comprehensive standard library.\n', '\n', 'Python interpreter
s are available for many operating systems, allowing Python code to run on a wid
e variety of systems. CPython, the reference implementation of Python, is open s
ource software and has a community-based development model, as do nearly all of
its variant implementations. CPython is managed by the non-profit Python Softwar
e Foundation.']
>>>
```

File Edit Format Run Options Window Help

```
f = open('test.txt', 'r')
myList = []

for line in f:
    myList.append(line)

print(myList)
```

KNOW ING

# Don’t forget to close the file

File Edit Format Run Options Window Help

```
f = open('test.txt', 'r')
myList = []

for line in f:
    myList.append(line)

print(myList)

f.close()
```

KNOW ING

# Write file

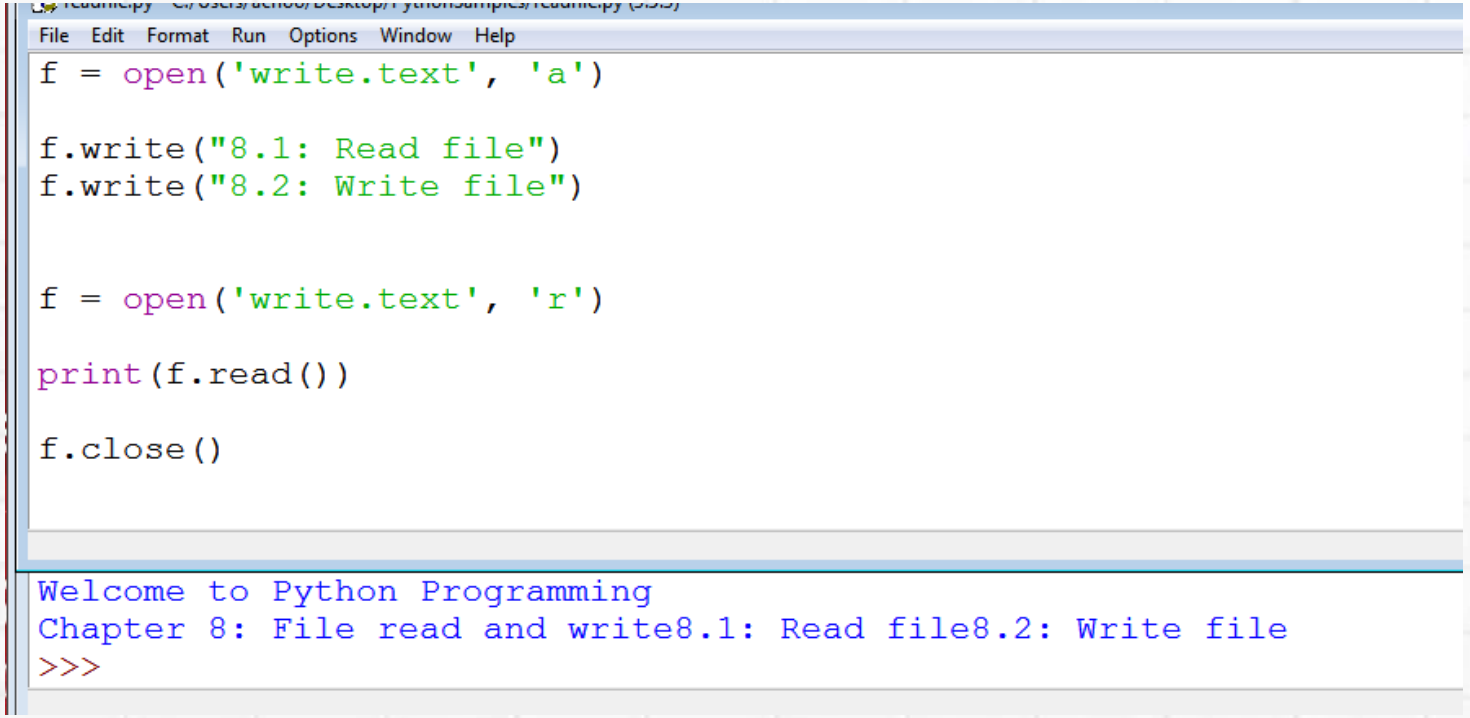
File Edit Format Run Options Window Help

```
f = open('write.text', 'w')

f.write("Welcome to Python Programming\n")
f.write("Chapter 8: File read and write")


f = open('write.text', 'r')

print(f.read())

f.close()
```

```
Welcome to Python Programming
Chapter 8: File read and write
>>>
```

KNOW ING

# Appending to a file




KNOW ING

# With statement

- You can work with file objects using the with statement.
- It is designed to provide much cleaner syntax and exceptions handling when you are working with code.
- That explains why it is good practice to use the with statement where applicable.

# With statement

- To use the with statement to open a file.

  *with open('filename') as file:*

- Some more example:

  *with open('text.txt') as file:*
  *data = file.read()*
  *....*

- You can also call upon other methods while using this statement. For example, you can do something like loop over a file object.

# Using the With statement in the real world

- To better understand the with statement, let's take a look at some real world examples just like we did with the file handling functions.
  - To write the file using the with statement.

    Example:

    *with open('test.txt', 'w') as f:*

    *f.write("Hello World")*

# Using the With statement in the real world

- To read a file line by line, output into a list:

  *with open('test.txt') as f:*

  *data = f.readlines()*

- This will take all of the text or content from the 'test.txt' file and store it into a string called 'data'.

# Splitting Lines in a Text File

- Let's explore a unique function that allows you to split the lines taken from a text file.
- What this is designed to do, is split the string contained in variable data whenever the interpreter encounters a space character.
- You can actually split your text using any character you wish - such as a colon, for instance.

# Example: Splitting Lines in a Text File

File Edit Format Run Options Window Help

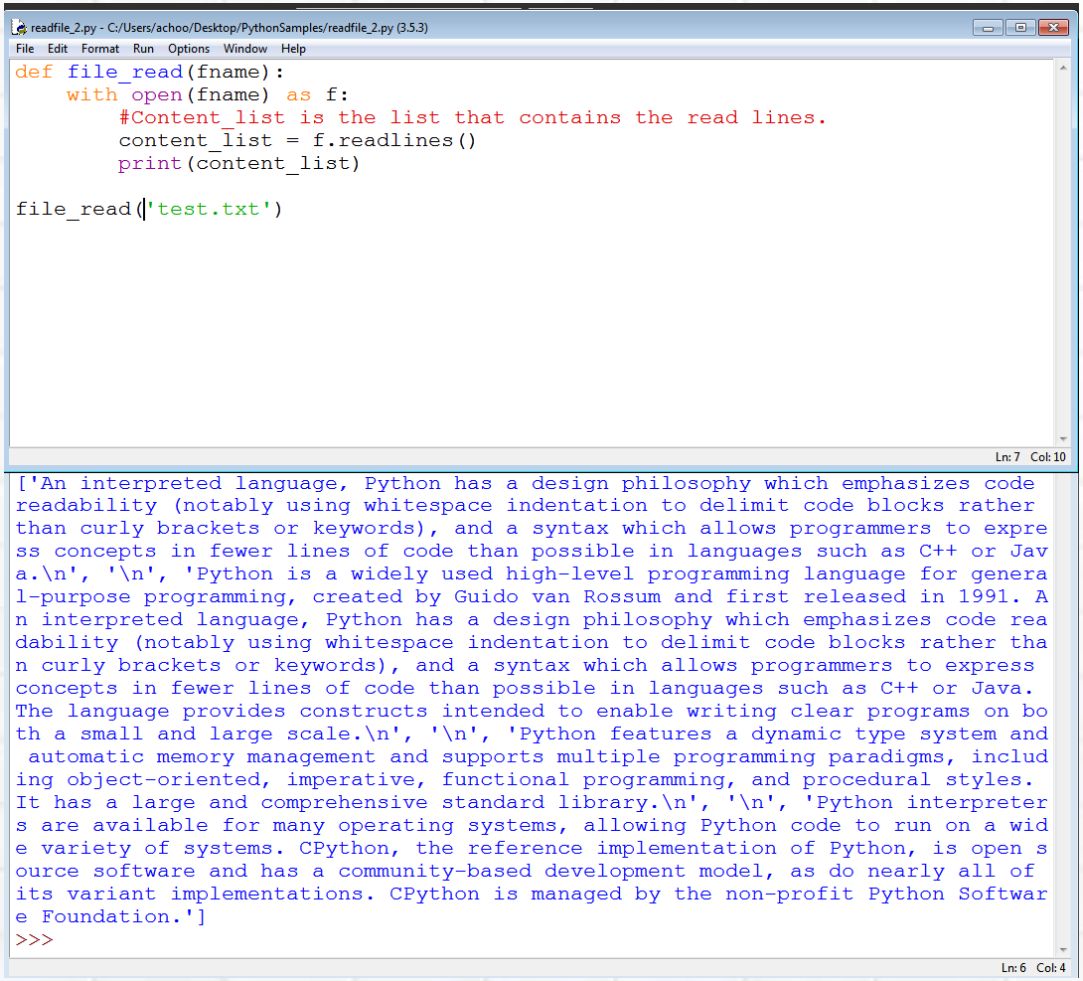
```
with open('write.text', 'r') as f:
    data = f.readlines()

for line in data:
    words = line.split()

print(words)
```

Ln: 8 Col: 0

```
['Chapter', '8:', 'File', 'read', 'and', 'write8.1:', 'Read', 'file8.2:', 'Write
', 'file']
>>>
```

KNOW ING

# Example: Read line by line and store into a list

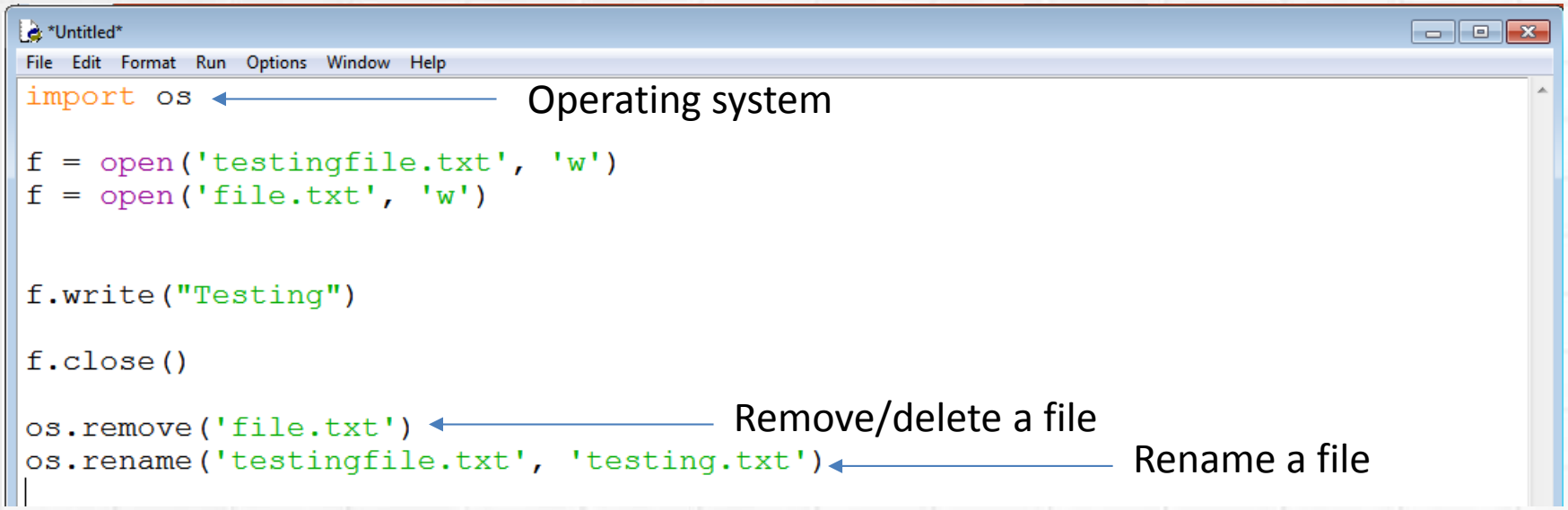



KNOW ING

# Example: Read random line from file


KNOW ING

# Rename and Remove a file




KNOW ING