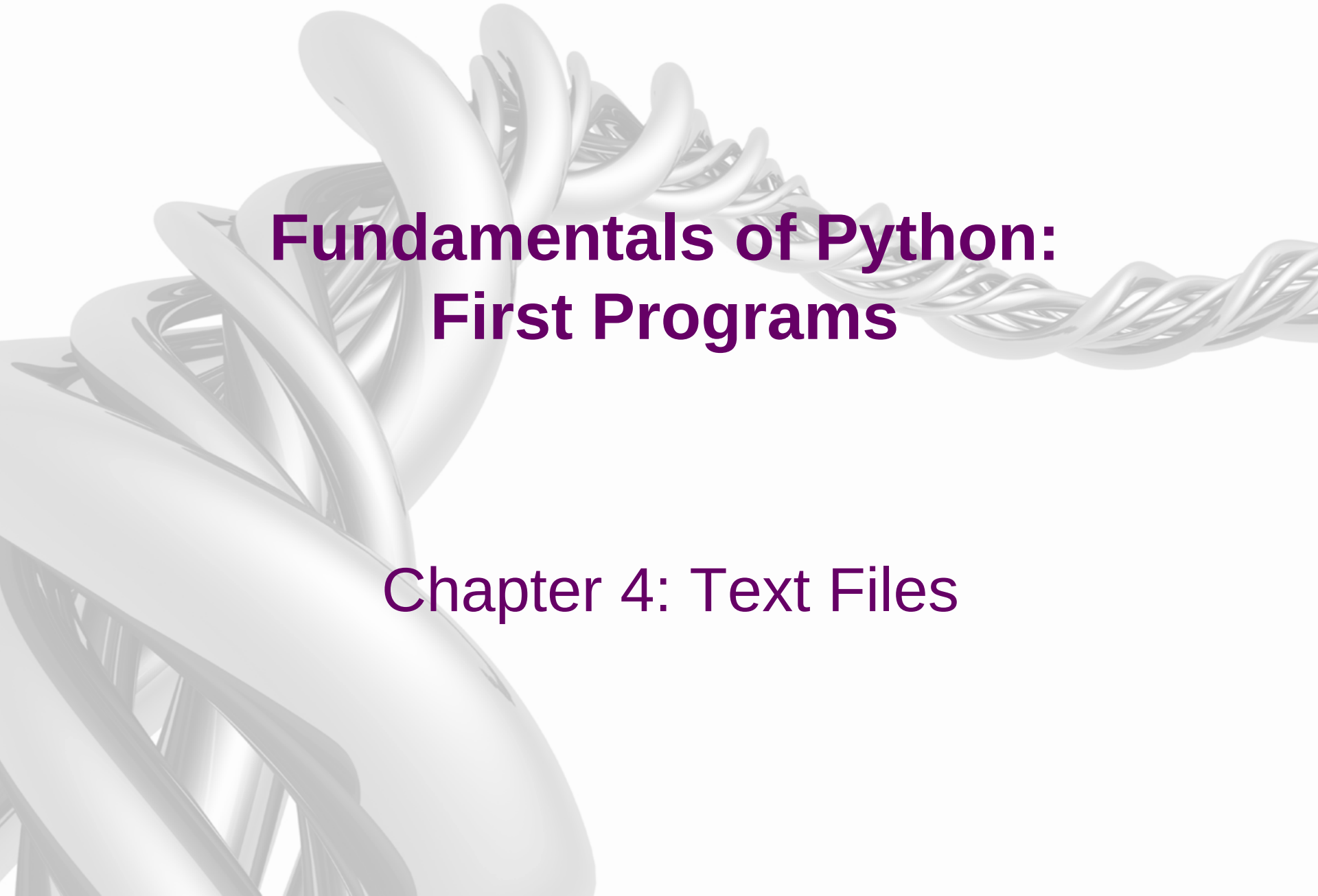

Fundamentals of Python: First Programs

Chapter 4: Text Files

Objectives

After completing this section, you will be able to

- Open a text file for output and write strings or numbers to the file
- Open a text file for input and read strings or numbers from the file

Problem Statement

- Read a text file from the current working directory that contains a “list” of words. Select one word at random and scramble the letters to form a “word jumble”. The user tries to guess the word and is limited to as many guesses as there are letters in the word.
  - What do you know about opening & saving files now?
- Write a BRIEF algorithm in pseudo code indicating how you would accomplish this.

Algorithm Discussion

- BRIEFLY discuss the essentials of the algorithm with your row partner.
  - Start with a brief private thinking time
  - We will use “Listen & Compare”
  - One group share your algorithm.

Today's Lesson "The How"

- Reading a "Technical Text"
- Determine the central ideas of the text, summarizing the complex concepts, processes, and/or information presented in the text by paraphrasing them in simpler but still accurate terms.
- Determine the meaning of symbols, key terms, and Python commands.

Today's Lesson "The What"

- Text Files: Reading and writing numeric and string data.
- Read Section 4.5 (Pages 141 – 147)

Today's Lesson "The How Part 2"

- Start with private thinking time.
- We will use "Listen & Compare" Structured Discussion with your partner.
- Groups will share (**Explain to your partner**)
  - What you learned including:

See Next Slide

Questions

- What is the definition of a text file?
- What format does data need to have as we read and write it to a text file?
- What does Python need to write data to a text file?
- What happens if you fail to close a file after writing data to it?
- What happens in Python if the file name you are trying to read is not found in the path or current working directory?
- After input is finished, what does the **read command** return?

Algorithm Discussion Part 2

- What do we need to do in order to read the text file?
  - List all the steps…
- How should we transfer the information from the text file to variable(s) inside the program?
- How can we separate the words so we can select a word at random?
- How can we “scramble” the characters in the word to convert it to a “word jumble”?

Exit Ticket: Short Quiz

- Socrative.com
- Room number: LCHS607

- If time permits, start program 4C

Text Files

- A text file is **software object** that stores data on permanent medium such as disk or CD
- When compared to keyboard input from human user, the main advantages of taking input data from a file are:
  - The data set can be much larger
  - The data can be input much more quickly and with less chance of error
  - The data can be used repeatedly with the same program or with different programs

Text Files and Their Format

- Using a text editor such as Notepad or TextEdit, you can create, view, and save data in a text file

```
34.6 22.33 66.75
77.12 21.44 99.01
```

- All data output to or input from a text file must be strings

Writing Text to a File

- Data can be output to a text file using a **`file` object**
- To **`open`** a file for output:

```
>>> f = open("myfile.txt", 'w')
```

  - If file does not exist, it is created
  - If it already exists, Python opens it; when data are written to the file and the file is closed, any data previously existing in the file are erased

```
>>> f.write("First line.\nSecond line.\n")
```

```
>>> f.close()
```

←Failure to close output file can result in data being lost

Writing Numbers to a File

- The **`file`** method **`write`** expects a string as an argument
  - Other types of data must first be converted to strings before being written to output file (e.g., using `str`)

```
import random
f = open("integers.txt", 'w')
for count in range(500):
    number = random.randint(1, 500)
    f.write(str(number) + "\n")
f.close()
```

Reading Text from a File

- You open a file for input in a manner similar to opening a file for output

```
>>> f = open("myfile.txt", 'r')
```

  - If the path name is not accessible from the current working directory, Python raises an error
- There are several ways to read data from a file
  - Example: the **`read`** method

```
>>> text = f.read()
>>> text
'First line.\nSecond line.\n'
>>> print(text)
First line.
Second line.
```

Reading Text from a File (continued)

- After input is finished, **`read`** returns an empty string

```
>>> f = open("myfile.txt", 'r')
>>> for line in f:
        print(line)

First line.

Second line.
>>> f = open("myfile.txt", 'r')
>>> while True:
        line = f.readline()
        if line == "":
           break
        print(line)

First line.

Second line.
```

Reading Numbers from a File

- Examples:

```
f = open("integers.txt", 'r')
sum = 0
for line in f:
    line = line.strip()
    number = int(line)
    sum += number
print("The sum is", sum)
```

```
f = open("integers.txt", 'r')
sum = 0
for line in f:
    wordlist = line.split()
    for word in wordlist:
        number = int(word)
        sum += number
print("The sum is", sum)
```

Reading Numbers from a File (continued)

METHOD	WHAT IT DOES
`open(pathname, mode)`	Opens a file at the given pathname and returns a **`file`** object. The **`mode`** can be `'r'`, `'w'`, `'rw'`, or `'a'`. The last two values, `'rw'` and `'a'`, mean read/write and append, respectively.
`f.close()`	Closes an output file. Not needed for input files.
`f.write(aString)`	Outputs **`aString`** to a file.
`f.read()`	Inputs the contents of a file and returns them as a single string. Returns `''` if the end of file is reached.
`f.readline()`	Inputs a line of text and returns it as a string, including the newline. Returns `''` if the end of file is reached.

[TABLE 4.3] Some **`file`** operations

Accessing and Manipulating Files and Directories on Disk

- When designing Python programs that interact with files, it's a good idea to include error recovery
- For example, before attempting to open a file for input, you should check to see if file exists
  - Function **`os.path.exists`** supports this checking
- Example: To print all of the names of files in the current working directory with a **`.py`** extension:

```
import os
currentDirectoryPath = os.getcwd()
listOfFileNames = os.listdir(currentDirectoryPath)
for name in listOfFileNames:
    if ".py" in name:
        print(name)
```

Accessing and Manipulating Files and Directories on Disk (continued)

os MODULE FUNCTION	WHAT IT DOES
`chdir(path)`	Changes the current working directory to **path**.
`getcwd()`	Returns the path of the current working directory.
`listdir(path)`	Returns a list of the names in directory named **path**.
`mkdir(path)`	Creates a new directory named **path** and places it in the current working directory.
`remove(path)`	Removes the file named **path** from the current working directory.
`rename(old, new)`	Renames the file or directory named **old** to **new**.
`rmdir(path)`	Removes the directory named **path** from the current working directory.

[TABLE 4.4] Some file system functions

Accessing and Manipulating Files and Directories on Disk (continued)

os.path MODULE FUNCTION	WHAT IT DOES
`exists(path)`	Returns **`True`** if **`path`** exists and **`False`** otherwise.
`isdir(path)`	Returns **`True`** if **`path`** names a directory and **`False`** otherwise.
`isfile(path)`	Returns **`True`** if **`path`** names a file and **`False`** otherwise.
`getsize(path)`	Returns the size of the object names by **`path`** in bytes.

[TABLE 4.5] More file system functions

Summary

- A text file is a software object that allows a program to transfer data to and from permanent storage
- A **`file`** object is used to open a connection to a text file for input or output
  - Some useful methods: **`read`**, **`write`**, **`readline`**
- **`for`** loop treats an input file as a sequence of lines
  - On each pass through the loop, the loop's variable is bound to a line of text read from the file