Topic 7

Files and Directories

Murdoch
UNIVERSITY

Last Topic

Murdoch UNIVERSITY

- Collections
- Sets
- Dictionaries
- Use of sets and dictionaries to solve problems
- Sequences
- Common characteristics of sequences

This Topic

Murdoch UNIVERSITY

- The concept of files and directories
- Read and write text files
- Read and write CSV files
- Get operating system information
- Access the underlying file system

Files and Directories

Murdoch UNIVERSITY

- A file is a sequence of bytes, usually stored in a media such as a hard disk, a solid-state drive, a flash drive, etc
- Files are "stored" inside a directory. The directory is also known as "folder". Each file under a directory is known as a directory entry.
- These so-called directories or folders are really just a special files the operating system created and used to organise all files in a hierarchical manner in a file system.
- A file has a name in a directory. However, several different files may have the same name as long as each of them resides inside a different directory.
- This means when referring to a file, just the file name is not enough. In most operating systems such as Windows, macOS and Linux, users use the path of a file to uniquely identify the file.

Files and Directories

Murdoch UNIVERSITY

- A path is simply a sequence of directory names leading to the file name.
- These directory names are separated with slashes: backward slash "\" on Windows and forward slash "/" on all Unix-like systems such as Linux and macOS.
- The name of the root directory is simply a slash. On Windows, you also need to specify the drive letter.
- Eg:

  `/Users/hong/ict582/foo` (macOS)

  `C:\Users\hong\ict582\foo` (Windows)

Absolute Path

Murdoch UNIVERSITY

- The two paths given in the previous slide is known as the *absolute path* – a path starting from the root of the file system:
  - The root is "/" on Unix-like operating systems (eg, macOS and Linux)
  - On Windows, it starts from the disk drive letter and "\" such as "C:\".
- It is not a good idea to use an absolute path to refer to a file in a program, because it would make the programs *unportable*.

Absolute Path

Murdoch UNIVERSITY

- Take the file `C:\Users\hong\ict582\foo` as an example, if my program refers to it using this absolute path, and the program and the file is copied to a different computer, the path of the file in the new computer is very likely different from the path in the original computer.
  - Eg, the file `foo` could be at `C:\Users\john\python\foo`.
- However, the program still refer to it using the path `C:\Users\hong\ict582\foo` in the original computer. When you run the program on the new computer, the program would not be able to locate the file on the new computer.

Relative Path

Murdoch UNIVERSITY

- A good practice is that we refer to a file using *the relative path* of the file in our program.
- As the name implies, a relative path is relative to something known as "current working directory", or CWD, of your program. Depending on the CWD of the program, the relative path for file foo would be different. As an example, for the file `/Users/hong/ict582/foo`:

CWD	Relative path for `foo`
`ict582`	`foo`
`hong`	`ict582/foo`
`Users`	`hong/ict582/foo`
`/`	`Users/hong/ict582/foo`

Relative Path

Murdoch UNIVERSITY

- Normally, we keep the program in a dedicated directory (the program directory) and keep the files the program needs either in the program directory or in one of its subdirectories.
- In the program, we refer to the files by their relative paths – relative to the program directory.
- We run the program from the program directory.
- In this way, we can copy the entire program directory to a different location and the program would still work!

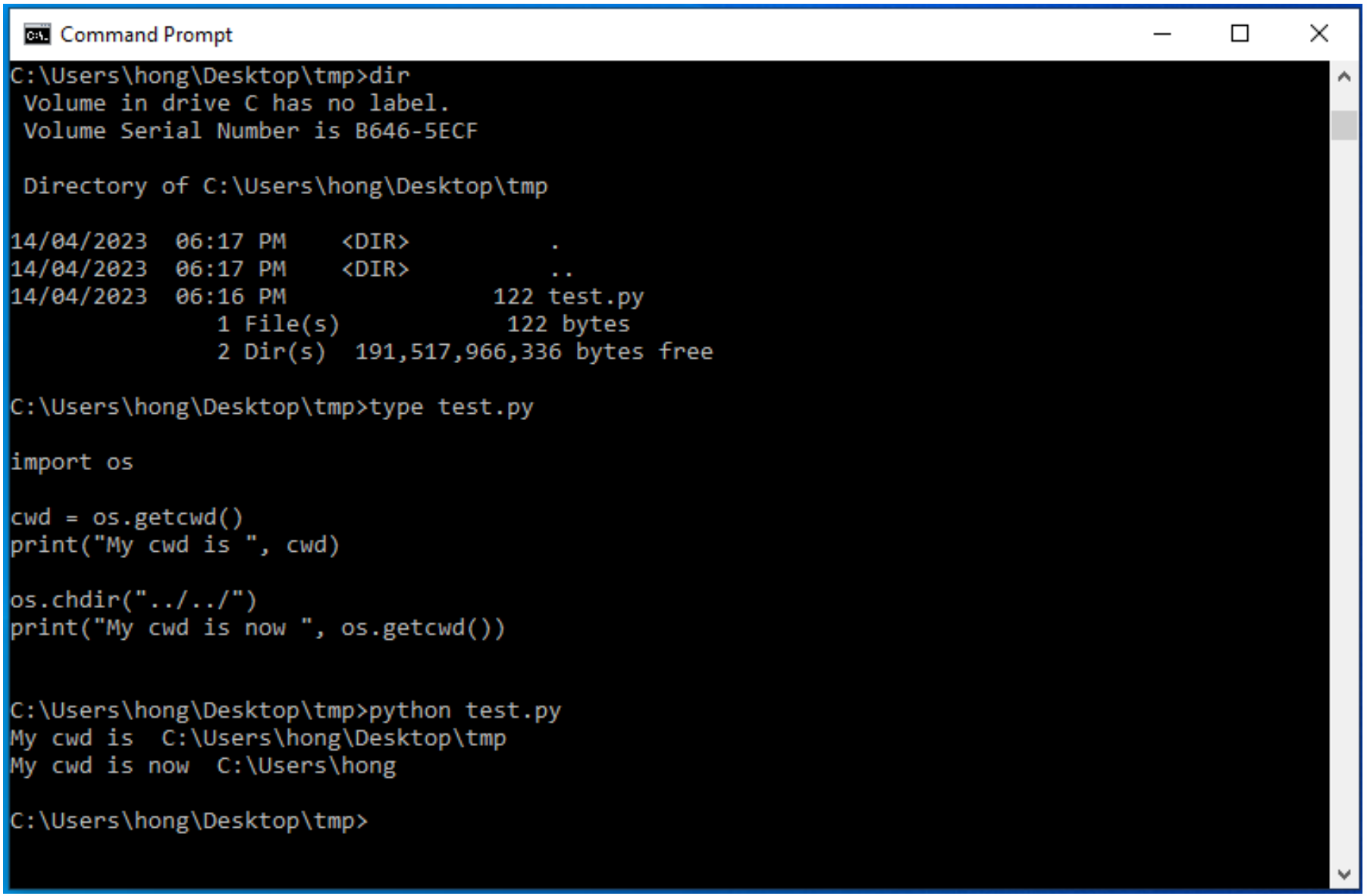
The Current Working Directory

Murdoch UNIVERSITY

- For the relative paths to work correctly, it is essential you know what your program's CWD is.
- Your program's CWD depends on how you start your program.
- If you start your Python program from a directory in a terminal, that directory would be your program's CWD.
- The module `os` provides two functions for us to get and set the program's CWD
  - `getcwd()`
  - `chdir(`*`the_new_directory`*`)`

The Current Working Directory

Murdoch UNIVERSITY



Opening a File

Murdoch UNIVERSITY

- In Python, central to all file operations is a built-in function `open`.
- Before reading from a file or writing to a file, you must firstly open the file using `open` function.
- This function would locate the file in the file system (usually stored in the disk) and create the necessary data structure in the memory for the subsequent operations on the file.
- The function would return an object representing the file.
- All subsequent access and manipulation must be done via this file object!

Opening a File

```
file = open("foo.txt")
filecontent = file.read()
print(filecontent)
```

- The function `open` opens a text file `foo.txt` using its relative path.
- It returns a file object (in variable `file`), which has many properties and methods for various file operations.
- The method `read` from object `file` reads a number of bytes from the file. The above statement reads the entire content of the text file `foo.txt` and returns it as a string.

Function `open`

Murdoch UNIVERSITY

- The function `open` takes many arguments, the two most important are the *file* path and *mode*.

```
open(file, mode='r', buffering=- 1, encoding=None, errors=None,
newline=None, closefd=True, opener=None)
```

mode	
`r`	Open for reading (default)
`r+`	Like `r`, but also allow writing to the file
`w`	Open for writing, truncate the file first. Create a new file if the file doesn't exist
`w+`	Like `w`, but also allow reading from the file
`a`	Open for appending at the end of the file without truncating it. Create a new file if the file doesn't exist
`a+`	Like a, but also allow reading from the file
`x`	Create a new file for writing, failing if the file already exists

Text File or Binary File

Murdoch UNIVERSITY

- A file contains a sequence of bytes representing any content, such a photo image, a video, a word file, and excel file, etc.
- However, there is a special type of files, known as text file, which represents a sequence of characters
  - these characters are not limited to those normally used by the English speakers – they include characters used by all human-kind.
- Apart from *text files*, all other types of files are called *binary files*.

Text File or Binary File

Murdoch UNIVERSITY

- Python doesn't care what it contains in a binary file
  - application programs need to work out what the content in the file means.
- However, Python treats text files specially.
- When we open a file, we need to tell Python whether the file is a text file or a binary file.
- This is achieved via the mode value `t` for text file (default) or `b`.

Text File or Binary File

Murdoch UNIVERSITY

- Examples

```
# mode = "rt", default
file = open("foo")  # mode = "rt", default

# read and write binary file
file = open("foo", "w+b")

# read and write text file
file = open("foo", "r+t")

# truncate and write to binary file
file = open("foo", "wb")

# write to the end of text file, and read from the file
file = open("foo", "a+t")

# create a new binary file for write
file = open("foo", "xb")
```

Read and Write a Text File

Murdoch UNIVERSITY

- The text `file` object returned from an `open` function provides several methods for reading and writing the file
  - `file.read(`*`n`*`)`: read the next *`n`* characters from the file, default to all characters in the file
  - `file.readline()`: read the next line from the file
  - `file.write(`*`str`*`)`: write string *`str`* to the file
- Examples:

```
str = file.read(10)     # read the next 10 characters
str = file.read()       # read the entire file
line = file.readline()  # read the next line from the file
file.write("Hi, there!") # write the string to the file
```

Read and Write a Text File

Murdoch UNIVERSITY

- To read lines from a text file one by one, you can loop over a text file:

```
file = open("foo.txt")
i = 0
for line in file:
    i += 1
    print("%3d:%s"%(i,line), end='')
```

- Each line read in includes the newline character at the end of the string.
  - This is why we remove the newline that would be added by the `print` statement with `end=''`, otherwise each line printed would be followed by a blank line.

Read and Write Position

Murdoch UNIVERSITY

- An open file has a current position, ie, where the next read or write should start.
  - The position is set to 0 when the file is open, unless you use append mode `a`
  - Each read or write operation would advance this position by the number of characters read or written.
- The method `file.tell()` would return the current position.
- The method `file.seek(`*`offset`*`)` can set the position to a new position *`offset`*.

Read and Write Position

Murdoch UNIVERSITY

```
file = open("foo.txt", "w+")

# write three lines
file.write("Line 1\n")
file.write("Line 2\n")
file.write("Line 3\n")

# now read the first line
file.seek(0)
print(file.readline(), end='')
```

Buffered Input/Output

Murdoch UNIVERSITY

- An open file maintains a buffer in the memory.
  - When you read from the file, you are actually reading from the buffer, rather than directly from the disk.
    - When the buffer is empty, a disk read operation is performed to fill the buffer
  - When you write to the file, you are actually writing to the buffer, rather than writing directly to the disk.
    - When the buffer is full, a disk write operation is performed to empty the buffer to the disk
- Buffered I/O reduces the frequency of actual disk access because actual disk I/O is always done block by block, rather than a few bytes each time.
  - Disk I/O is extremely slow compared to memory operations, therefore reducing the frequency of disk I/O can substantially increase the I/O efficiency, thus the overall system performance.

Buffered Input/Output

Murdoch UNIVERSITY

- Due to buffering, the stuff you write to a file may not appear in the actual disk file immediately.
- Usually this is not a problem, because when the program finishes, the content in the buffer will be flushed out to the disk.
- However, in some situations, this is not ideal. Eg, a server writing a log to the log file. Due to buffering, the system administrator would not see the log immediately.
- To resolve this issue, you can force a buffer flush using the method `file.flush()`.

Buffered Input/Output

Murdoch UNIVERSITY

```
import time

file = open("foo.txt", "w+")
file.write("line 1\n")
file.write("line 2\n")
file.write("line 3\n")

# wait for 10 seconds
time.sleep(10)

# check the file foo.txt on
# the disk within 10 seconds,
# you will see nothing
```

```
import time

file = open("foo.txt", "w+")
file.write("line 1\n")
file.write("line 2\n")
file.write("line 3\n")

file.flush()

# wait for 10 seconds
time.sleep(10)

# check the file foo.txt,
# you will see the three
# lines immediately
```

Close File

Murdoch UNIVERSITY

- Each open file would take up a substantial amount of system resources (eg, memory space), therefore you should close the file as soon as the file is no longer needed to free up the system resources.
- The method `file.close()` would close the file.
- Once a file is closed, you cannot use the file object to invoke the methods such as `read`, `readline`, `write`, `tell` and `seek`, to perform operations on the file.
- If you need to do something on the file again after it is closed, you must re-open it.

`with` Statement

- A better way to open a file is by using `with` statement

```
with open(path, mode) as file:
    # perform operations on file
```

- The `with` statement would automatically close the file at the end of the `with` statement. No need to invoke `file.close()` if you use `with` statement.

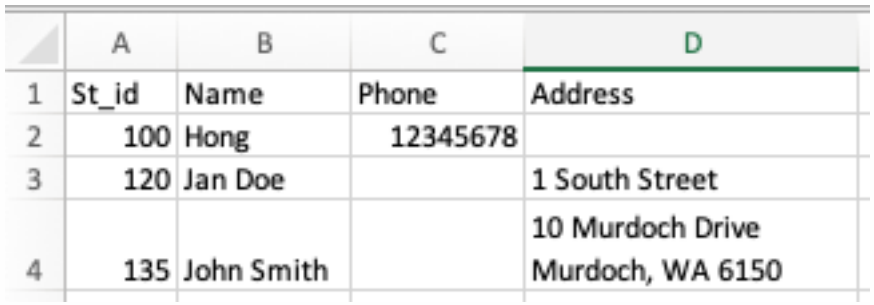
```
with open("foo.txt", "r+") as myfile:
    for line in myfile:
        print(line, end='')
    myfile.write("this is the last line")

# the file myfile is now closed!
```

CSV File

- CSV (Comma Separated Values) file is a popular text file format for storing tabularized data for data exchange. Many software tools use this format, including Microsoft Excel.
- A CSV file is a text file. Each line contains multiple field values separated by commas. An optional first line contains field names (the header).
- Example `staff.csv`

```
St_id,Name,Phone,Address↩
100,Hong,12345678,↩
120,Jan Doe,,1 South Street↩
135,John Smith,,"10 Murdoch Drive↩
Murdoch, WA 6150"
```



	A	B	C	D
1	St_id	Name	Phone	Address
2	100	Hong	12345678	
3	120	Jan Doe		1 South Street
4	135	John Smith		10 Murdoch Drive Murdoch, WA 6150

Note 1) the symbol ↩ represents the invisible newline characters, 2) the address in the last line contains a newline character!

Open a CSV File

- To open a csv file, you might think the following example is good enough:

```
file = open("staff.csv", "rt")
```

- However, two issues are special to a CSV file:
  - Some field values may contain new line characters, to avoid processing errors, we should add `newline=''` to the open call
  - Many CSV files are generated by Microsoft Excel, this type of files have a file signature `0xefff` at the beginning of the file. To handle this type of file properly, we need to add `encoding='utf-8-sig'`

```
file = open("staff.csv", "rt",
            newline='', encoding='utf-8-sig')
```

Read a CSV File: reader object

Murdoch UNIVERSITY

- Python has a built-in module, `csv`, for handling CSV files.
  - We have introduced modules in Topic 4.
  - There are two types of modules, those that come with Python 3 package (built-in modules) and those that were written by the third parties.
  - To use a module, you need to import it into your program, as

    ```
    import csv
    ```

  - The csv module provides function `reader` that turns a csv file into an iterable object (similar to string, list, tuple etc). You can loop through the `reader` object to get each and every row from the csv file.

    ```
    reader = csv.reader(file)
    ```

Read a CSV File: reader object

```
import csv
filepath = "staff.csv"
csvfile = open(filepath, newline='', encoding='utf-8-sig')

# create a reader object for reading lines from the csv file
# reader is an iterable object
reader = csv.reader(csvfile)

# read and print header
header = next(reader))  # return the next row
print(header[1],header[2])

# loop through the reader object
for row in reader:
    print("name: %s, phone: %s" % (row[1], row[2]))
```

- The above program would read each row and print out the 2nd and 3rd field values in the row.
- Each row is actually a list

Read a CSV File: DictReader object

Murdoch UNIVERSITY

- The `csv` module also provides function `DictReader`. Unlike `reader` object returned from `reader` function, where each row is a list, with `DictReader`, each row is a dictionary.

```
import csv
filepath = "staff.csv"

with open(filepath, newline='', encoding='utf-8-sig') as csvfile:
    # create a DictReader object for reading rows
    # from the csv file
    reader = csv.DictReader(csvfile)

    # loop through the reader object
    for row in reader:
        print("name: %s, phone: %s" % (row["Name"], row["Phone"]))
```

Write a CSV File: writer object

- The module `csv` also provides function `writer`, which returns a `writer` object. The object has a method `writerow` (write a row) for us to write a row to the csv file.

```
import csv
filepath = "newstaff.csv"

with open(filepath, "w+", newline='', encoding='utf-8-sig')as csvfile:
    writer = csv.writer(csvfile)

    header = ["Name", "Phone", "Address"]
    writer.writerow(header)

    record = ["Hong", 12345678, "1 South Street"]
    writer.writerow(record)

    record = ["John Smith", '', "100 Murdoch Drive\nMurdoch WA 6150"]
    writer.writerow(record)
```

Access Operating System

Murdoch UNIVERSITY

- We often need to do something on the underlying operating system. Python provides builtin modules `os` and `platform` for us to interact with the underlying operating system.
- The details of this module can be found in the Python documentation:
  https://docs.python.org/3/library/os.html
  https://docs.python.org/3/library/platform.html
- In this topic, we will only highlight a few functions related to manipulating the files and directories on the underlying file system.

Getting Information about the Operating System

Murdoch UNIVERSITY

```
import platform

print(platform.system())    # os name
print(platform.release())   # os version
print(platform.machine())   # machine type
print(platform.node())      # machine's network name
print(platform.python_version())  # Python version
```

- Running the above code on my computer output the following details:

```
Darwin
19.6.0
x86_64
Mac2021
3.10.2

1
```

Get and Set Current Directory


```
import os

# assume the program's cwd is in /Users/hong/ict582/t07
# and the directory tests contains two subdirectories ex1 and ex2

print(os.getcwd())
# /Users/hong/ict582/t07

os.chdir("ex1")
print(os.getcwd())
# /Users/hong/ict582/t07/ex1

os.chdir("../ex2")
print(os.getcwd())
# /Users/hong/ict582/t07/ex2

os.chdir("/Users/hong/Desktop")
print(os.getcwd())
# /Users/hong/Desktop
```

Make and Remove Directory


```
import os

# create a new, empty directory new-dir under the current directory
os.mkdir("new-dir")

# go to the new directory
os.chdir("new-dir")
print(os.getcwd())
# /Users/hong/ict582/t07/new-dir

# go back to the original directory
os.chdir("..")
print(os.getcwd())
# /Users/hong/ict582/t07

# remove the empty new-dir
os.rmdir("new-dir")
```

Remove and Rename File

Murdoch UNIVERSITY

```
import os

# delete file "foo.txt" from the current working directory
os.remove("foo.txt")

# change the name of directory "ex2" to "new-ex2"
# in the current working directory
os.rename("ex2", "new-ex2")

# change the name of file "bar.txt" to "foo.txt"
# in the current working directory
os.rename("bar.txt", "foo.txt")
```

List Directory Entries

Murdoch UNIVERSITY

```
import os

# get a list of entries in the current directory
dir_entries = os.listdir(".")

# dir_entries is a list, print the list
print(dir_entries)

# print each entry in the parent directory
for ent in os.listdir(".."):
    print(ent)
```

Summary

Murdoch UNIVERSITY

- The concept of files and directories
- Absolute path vs relative path
- Open a file
- Read and write a text file
- Open a CSV file
- Read and write a CSV file
- Get operating system information
- Manipulate files and directories
- List directory entries