

8.16 File handling

created by HardyWen

1. Theory

- **Definition**
 - programming statements that allow texts files to be opened, read from, written to, and closed
 - the storage and access of data in a file
- **Why do we need file handling?**
 - to store the data/output of the program so that they can be accessed even after the program has stopped running, and therefore we don't need to run the program again to generate the output
- **Steps**
 - these are **key** steps in writing file-handling codes in pseudocode and Python
 1. opening file
 2. performing read/write
 3. closing file
- **file handling modes**
 - **"r", READ**
 - in this mode, you could only read data from the file
 - **"w", WRITE**
 - in this mode, you can write data into the file
 - **A new file would be created and any existing data inside the file will be lost!**
 - **"a", APPEND**
 - for data to be added to the file **after** any existing data

2. Practical

1. Reading

1. Reading in Pseudocode

- the *READFILE* function in Pseudocode takes in two parameters, and
 - specifies from which file the data should be read from
 - specifies to which variable the read data should be stored
 - remember that the *READFILE* function in Pseudocode only returns the data of 1 line of pseudocode when being used once
 - for example, considering the file "a.txt",

```
/* a.txt */
I love computer science.
But so many to be memorized in IG.
which is awful.
```

by using **READFILE a.txt LineofText** once, the content returned to *LineofText* would be the first line "I love computer science". Hence, we need to use a loop together with *EOF* to make sure we read every line from the file.

- the *READFILE* function is always used with *EOF()*
 - *EOF* stands for "end of file"
 - *WHILE NOT EOF()* means "when the file isn't finished"

```
DECLARE LineofText: STRING // LineofText is a function that stores a
line in the file
OPENFILE <filename> FOR READ
WHILE NOT EOF(<filename>)
    READFILE <filename>, LineofText
    /* perform operations to the function LineofText here */
CLOSEFILE <filename>
```

2. Reading in Python

- there are three reading functions in Python
 1. *myfile.read()*
 - this returns all the content in the file as a string
 2. *myfile.readline()*
 - this returns one line of the file as a string
 - actually it is a self-iterator but you don't need to care about this in IG I believe
 3. *myfile.readlines()*
 - this returns each line of the file as a single string inside a list
- considering the file "a.txt"

```
/* story.txt */
I love coding.
Coding is fun.
Yes you are right.
```

- by using

```
myfile = open("story.txt", 'r') # the 'r' symbolizes that the
mode is "read only"
content = myfile.read()
myfile.close()
```

the value inside *content* would be

```
"I love coding.\nCoding is fun.\nYes you are right."
```

while the "\n" here symbolizes end of the line (换行)

- by using

```
myfile = open("story.txt", 'r') # the 'r' symbolizes that the
mode is "read only"
content = myfile.readline()
myfile.close()
```

the value returned would be

```
"I love coding.\n"
```

- by using

```
myfile = open("story.txt", 'r') # the 'r' symbolizes that the
mode is "read only"
content = myfile.readlines()
myfile.close
```

the value returned would be

```
["I love coding\n", "Coding is fun\n", "Yes you are right."]
```

2. Writing/Appending

1. Writing/Appending in Pseudocode

- writing and appending are much easier in Pseudocode than reading
- the general format is

```
OPENFILE <filename> FOR WRITE // or FOR APPEND if you want to carry out
appending
WRITEFILE <filename>, <variable> // the variable here stores the content
to be written
CLOSEFILE <filename> // remember to close the file at last
```

2. Writing/Appending in Python

- the same thing happens in Python considering *file.write()* and *file.writeline*
 1. *file.write()* would write a single string inside the file
 2. *file.writelines()* would write a list of strings into the file

```
myfile = open(filename, "w") # or "a" is you want to carry out appending
myfile.write(content) # or myfile.append(), the content here stores what
should be written in
myfile.close() # remember to close it!
```

