CSIT110

Fundamental Programming with Python

Files

Goh X. Y.

In this lecture

- Read from a text file
- Write to a text file
- Read from a CSV file
- Write to a CSV file

Opening a file

To open a file to read or write, we use the function `open()` which returns a file object.

Normally, we call the function with two arguments:

```
open(file_path, mode)
```

where:

- `file_path`: to specify the path to the file
- `mode`: to specify how the file is going to be used

File

`open(file_path, mode)`

`mode`: to specify how the file is going to be used

- `mode= "r"` : the file will only be read
- `mode= "w"` : for only writing (an existing file with the same name will be erased)
- `mode= "a"` : opens the file for appending, any data written to the file is automatically added to the end
- `mode= "r+"` : opens the file for both reading and writing

The mode argument is optional;

`mode = "r"` will be assumed if it's omitted.

Writing to a file

Write some silly sentences to a text file.

```
silly_file_path = "put/the/file/path/here/silly.txt"

with open(silly_file_path, "w") as silly_file:
    silly_file.write("Hi! ")
    silly_file.write("I am Sam.\n")
    silly_file.write("Would you like green egg and ham?\n")
```

When your program proceeds to the code outside the ‘with’ scope, the file automatically closes.

This is the content of the output file, silly.txt:

```
Hi! I am Sam.
Would you like green egg and ham?
```

Reading a text file with a while loop

```
text_file_path = "put/the/file/path/here/silly.txt"

with open(text_file_path) as silly_file:

    # read each line until end of file
    while True:
        line = silly_file.readline()

        # reach the end of file
        if(line == ""):
            break
        print(line)
```

<This line is akin to
`silly_file = open(text_file_path)`

`.readline()` **automatically returns the next available line of text in the file whenever it is called.**

Reading a text file with a for-each loop

```
text_file_path = "put/the/file/path/here/silly.txt"

with open(text_file_path) as silly_file:

  # read each line until end of file
  for line in silly_file:
    print(line)
```

Placing silly_file in the `for` statement automatically returns the next available line of text in the file in each loop.

Example: Write a times table to a file

Ask the user which number to generate times table and which file to write to.

```
# ask user to enter number
user_input = input("Enter a number to generate times table: ")
number = int(user_input)

# ask user to enter file path
file_path = input("Enter output file path: ")

# write times table to file
with open(file_path, "w") as timestable_file:

  for i in range(1, 10):
    timestable_file.write("{0} x {1} = {2}\n".format(number, i,
number*i))
```

```
Enter a number to generate times table: 7
Enter output file path: C:/Users/jsmith/doc/timestable.txt
```

Example: Write a times table to a file

```
Enter a number to generate times table: 7
Enter output file path: C:/Users/jsmith/doc/timestable.txt
```

Content of the output timestable.txt file:

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
```

CSV File

A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values.

A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record.

Each record consists of one or more fields, separated by commas.

CSV File

Here is an example of a CSV file that contains student information:

```
stn, first_name, last_name
1111, John, Smith
2222, Lee, May
3333, Ye, Zhang
```

Writing to a csv File

We want to write to a CSV the following content that contains student information

```
stn,first_name,last_name
1111,John,Smith
2222,Lee,May
3333,Ye,Zhang
```

For each line (except the title line), we need to construct a **dictionary** that contains the information of a student:

```
{"stn": "1111", "first_name": "John", "last_name": "Smith"}

{"stn": "2222", "first_name": "Lee", "last_name": "May"}

{"stn": "3333", "first_name": "Ye", "last_name": "Zhang"}
```

Writing to a csv File

Write to a CSV file: first write a header, then write each **dictionary** as a line

```
import csv

student_file_path = "put/the/file/path/here/student.csv"

with open(student_file_path, "w") as student_file:
    field_name_list = ["stn", "first_name", "last_name"]

    writer = csv.DictWriter(student_file, fieldnames=field_name_list)

    writer.writeheader()
    writer.writerow({"stn": "1111", "first_name": "John", "last_name": "Smith"})
    writer.writerow({"stn": "2222", "first_name": "Lee", "last_name": "May"})
    writer.writerow({"stn": "3333", "first_name": "Ye", "last_name": "Zhang"})
```

0: import the csv library

1: create an instance of the csv.DictWriter

2

3

Reading a csv File with the csv library

```
import csv

student_file_path = "put/the/file/path/here/student.csv"

with open(student_file_path) as student_file:
    reader = csv.DictReader(student_file)

    for row in reader:
      student_number = row.get("stn")
      fname = row.get("first_name")
      lname = row.get("last_name")
      print("{0:<10}{1:<10}{2:<10}".format(student_number, fname, lname))
```

create an instance of the csv.DictReader

Console output:

```
1111      John      Smith
2222      Lee       May
3333      Ye        Zhang
```

File – Using the csv library, writer example 2

We want to write to a CSV the following content that contains subject information

```
code,name,cp
MATH100,Algebra,6
CS200,C++,2
IT300,Biotechnology,3
```

For each line (except the title line), we need to construct a **dictionary** that contains the information of a subject:

```
{
  "code": "MATH100",
  "name": "Algebra",
  "cp": 6
}
{"code": "CS200", "name": "C++", "cp": 2}
{"code": "IT300", "name": "Biotechnology", "cp": 3}
```

File – Using the csv library, writer example 2

Write to a CSV file: first write a header, then write each **dictionary** as a line

```
import csv

subject_file_path = "put/the/file/path/here/subject.csv"

with open(subject_file_path, "w") as subject_file:
    field_name_list = ["code", "name", "cp"]
    writer = csv.DictWriter(subject_file, fieldnames=field_name_list)

    # write the header
    writer.writeheader()
    # write each record
    subject_dict = {"code": "MATH100", "name": "Algebra", "cp": 6 }
    writer.writerow(subject_dict)
   ...
```

File – Using the csv library, reader example 2

Read CSV file, use for-loop to get one line at a time:

```
import csv

subject_file_path = "put/the/file/path/here/subject.csv"

with open(subject_file_path) as subject_file:
    reader = csv.DictReader(subject_file)
    for row in reader:
        subject_code = row.get("code")
        subject_name = row.get("name")
        cp = row.get("cp")
        print("{0:<10}{1:<30}{2:<10}".format(subject_code, subject_name, cp))
```

Console Output:

```
MATH100   Algebra                        6
CS200     C++                            2
IT300     Biotechnology                  3
```

Try it yourself!

Student enrolment information is stored in a CSV file as follows:

```
stdn,subject,cp
1111111,MATH100,3
1111111,MATH111,6
1111111,CS121,6
2222222,ACCY100,6
2222222,PHY131,4
```

We want to read this file and display the info on the console output in the following format:

```
Student 1111111:
MATH100    3 cp
MATH111    6 cp
CS121      6 cp
Total: 15 cp

Student 2222222:
ACCY100    6 cp
PHY131     4 cp
Total: 10 cp
```

Any questions?