

## L07 Tasks

L07-T1: Reading & Writing text files

L07-T2: Copying from one file to another

L07-T3: Testing data inside a file, palindromes

L07-T4: CSV file

L07-T5: Menu-based program - calculator. Continues from L05-T5

### Submission:

Submit **all** the exercises to CodeGrade via Moodle before the deadline.

### Note:

- **(Important for this week!)** When opening and closing files, please use the method that has been taught in this course. If you use other methods to handle these files, CodeGrade will reject your work. The file-handling method can be found in the learning material for this week ([link](#)).
- If you see some "Code Structure Tests" hidden in CodeGrade but they are not mentioned in the task description, you don't need to worry about them. They are just there to make sure the code is ok.
- Be especially careful with spaces so that your output follows the sample output. Note that also each input-string in the program is ending with '\n'. The reason for this is that it makes the output in CodeGrade more readable.

## L07-T1: Reading & Writing text files

Let's practice writing and reading files in functions.

1. Write a program consisting of two subprograms in the style of a menu-based program from previous exercises. The functions needed are `file_write(name)` and `file_read(name)`.
2. In the main program, ask for the name of the file to be saved and pass it as a parameter to both subprograms.
3. `file_write(name)` function should ask the user for the names and save them in a text file:
  - a. Ask for the name in repeating structure and stop asking when the user inputs 0.
  - b. The file must be opened at the beginning of the function and any previous contents of the file must be destroyed upon opening.

- c. At the end of the function, the file must be closed.
4. `file_read(name)` function is for verification that the file has the desired content.
  - a. Open the file, read its contents, and print one name per line on the screen and finally close the file.

**Note:**

- Please note that in this task the repetition structure (L04), subroutines (L05) and file processing (L07) must be implemented correctly. An error in any of these points will prevent the program from working properly, so check these structures from previous weeks as needed.
- Make sure that your code follows the instructions, “Code structure tests” will be used to make sure that those files are handled correctly (open, handle, and close the file).

**Example run 1:**

```
Enter the name of the file to be saved:
Exercisel.txt
Enter a name to save to the file (0 to stop):
John
Enter a name to save to the file (0 to stop):
Catherina
Enter a name to save to the file (0 to stop):
Anna
Enter a name to save to the file (0 to stop):
0
The following names are stored in the file 'Exercisel.txt':
John
Catherina
Anna
```

## L07-T2: Copying from one file to another

Write a function `file_copy(fileA, fileB)` that copies the content of one text file to another. In Moodle, you can find the file “Exercise2.txt” which produces the following result:

**Example run 1:**

```
Please give the name of the source file:
Exercise2.txt
Please give the name of the destination file:
out.txt
File copied successfully!
```

## L07-T3: Testing data inside a file, palindromes

This task continues the earlier task L03-T5. This time the strings to be tested are read from the file and found palindromes are written to another file.

At the beginning of the main program, ask for the name of the text file to be tested. You **must** specify the name of the *output file* as Palindromes.txt. Open both files at the start. Then, read the test file one line at a time and perform the palindrome test.

Your program should print the content of each line and the result of the test to the screen: "XXX is a palindrome.", "XXX is not a palindrome." according to the example run below.

The actual test material is in the file Exercise3.txt. Some of its lines are palindromes and some are not.

Some hints to solve the task:

- Note that the `readline()` function also returns a newline in the string, i.e. "monkey" is read from the file as the string "monkey\n".
- The newline character should not be considered in palindrome testing. The line containing only the newline character stops reading the file (`len(row) == 0`).
- The texts may contain uppercase and lowercase letters. You should compare lowercase words. Use lower operation:

```
>>> "BorrowOrRob".lower()  
'borroworrob'
```

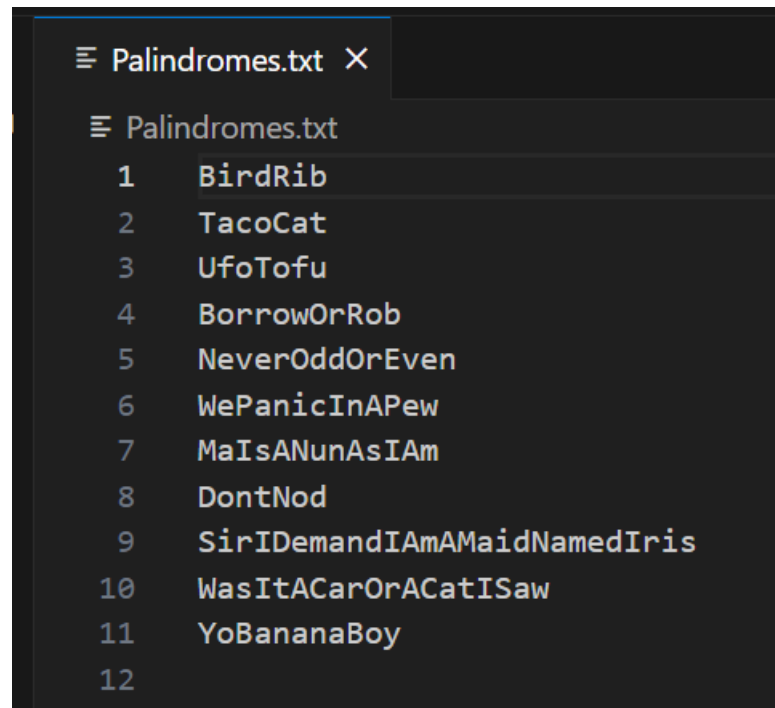
- The file to be read is UTF-8 encoded.

### Example run 1:

```
Enter the name of the file to be read:  
Exercise3.txt  
row 'BirdRib' is a palindrome.  
row 'TacoBell' is not a palindrome.  
row 'TacoCat' is a palindrome.  
row 'UfoTofu' is a palindrome.  
row 'UfosAreHere' is not a palindrome.  
row 'BorrowOrRob' is a palindrome.  
row 'ItsNowOrNever' is not a palindrome.  
row 'NeverOddOrEven' is a palindrome.  
row 'PanicNapic' is not a palindrome.  
row 'WePanicInAPew' is a palindrome.  
row 'ThisIsNonsense' is not a palindrome.  
row 'MaIsANunAsIAm' is a palindrome.  
row 'WhyNot' is not a palindrome.  
row 'DontNod' is a palindrome.  
row 'SirIDemandIAmAMaidNamedIris' is a palindrome.
```

```
row 'AppleMaple' is not a palindrome.  
row 'WasItACarOrACatISaw' is a palindrome.  
row 'YoBananaBoy' is a palindrome.
```

Then the output file Palindromes.txt will look like this:



```
Palindromes.txt X  
Palindromes.txt  
1 BirdRib  
2 TacoCat  
3 UfoTofu  
4 BorrowOrRob  
5 NeverOddOrEven  
6 WePanicInAPew  
7 MaIsANunAsIAm  
8 DontNod  
9 SirIDemandIAmAMaidNamedIris  
10 WasItACarOrACatISaw  
11 YoBananaBoy  
12
```

## L07-T4: CSV file

Write a program that reads a CSV (Comma-Separated Values) file and prints only the second column.

In Moodle, you can find the file Exercise4.csv

### Example run 1:

```
Give the name of the CSV file:  
Exercise4.csv  
30  
25  
35
```

## **L07-T5: Menu-based program - calculator. Continues from L05-T5**

This task modifies the previously made calculator, where the selection structure of the menu-based calculator was made in task L03-T3, the repetition structure L04-T5 and the subroutine structure L05-T5.

This time the task is to add file processing: read inputs from one file and write the results to another file. As a starting point for this task, you should use the program L05-T5.

This time, you only need to change the main program and the subprogram that handles reading the numbers.

1. Change the main program to ask for the name of the file to be read and open it for reading.
2. After that, open the file Exercise5\_output.txt to write the results.
3. When the user chooses to enter numbers, the main program must call the `read_value()` with a "file handle" as a parameter. This subroutine returns the current number that was read from the file.
4. Call the same subroutine a second time, and you will get another number. Print them on the screen according to the example run.
5. If the user is trying to divide something by 0, the result will be "Cannot divide by zero.", and this result will be written to the file Exercise5\_output.txt.
6. If there are no more values in the file, the subprogram will print the message "End of values, end the program." and return 0 as shown in the example run. In that case, it is sensible to stop running the program by entering 0 in the menu.
7. After a successful calculation, the program writes the string returned by the calculation subroutine to a new line in the file and prints the information about the recording on the screen.
8. After the user selects stop, the program closes both files and exits the program according to the example run.

Moodle has a file Exercise5\_numbers.txt that contains inputs for the program.

Moodle also has a file Exercise5\_output.txt that contains outputs for the program.

### Example run 1:

```
Enter the name of the file to read:
Exercise5_numbers.txt
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
1
Values 2 and 8 were read
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
2
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
3
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
4
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
```

```
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
5
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
1
Values 1 and 3 were read
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
2
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
3
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
1
Values 2 and 2 were read
This calculator can perform the following functions:
```

```
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
4
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
5
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
1
End of values, end the program.
End of values, end the program.
Values 0 and 0 were read
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
0
Stopping
```



Then the output file Exercise5\_output.txt will look like this:

```
≡ Exercise5_output.txt
1  sum 2 + 8 = 10
2  subtract 2 - 8 = -6
3  multiply 2 * 8 = 16
4  division 2 / 8 = 0.25
5  sum 1 + 3 = 4
6  subtract 1 - 3 = -2
7  multiply 2 * 2 = 4
8  division 2 / 2 = 1.0
9
```

### Example run 2:

```
Enter the name of the file to read:
Exercise5_numbers.txt
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
1
Values 5 and 0 were read
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
5
Result saved in file.
This calculator can perform the following functions:
1) Enter the values
2) Sum
```

```
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
1
Values 2 and 8 were read
This calculator can perform the following functions:
1) Enter the values
2) Sum
3) Subtract
4) Multiply
5) Divide
0) Stop
Select the function (0-5):
0
Stopping
```

Then the output file Exercise5\_output.txt will look like this:

```
≡ Exercise5_output.txt
1  Cannot divide by zero.
2
```