TASK

# Capstone Project III — Files

Visit our website

# Introduction

## WELCOME TO THE THIRD CAPSTONE PROJECT!

This Capstone is a milestone in your learning so far. You will consolidate the knowledge that you have gained and apply it to solve a problem in this project! At this point, you should have a comprehensive understanding of string handling and working with external data sources. This project will focus on incorporating these subjects to build a useful task management application. Be creative — you'll be tasked with a set of criteria to meet but the rest is up to you! You will add further functionality to this Capstone Project later in this level. It is worth spending time and effort to make this a project that you can be proud of! It is another project you can add to your developer portfolio!

Get in touch

**Connect for support**

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

A note from the

**HyperionDev Team**

Information Technology (IT) and Computer Science (CS) are often used interchangeably, but you might be surprised to know that they have two very different meanings. Even specialists with tertiary education in Computer Science, Engineering or related fields sometimes have predetermined (and quite possibly incorrect) ideas about what these terms mean. Find out more about the difference between these two terms **here**.

## DEVELOPER PORTFOLIO

Developers who have the edge are those who find ways to apply their newfound skills from the get-go. A **developer portfolio** (a collection of software that you have made) allows you to demonstrate your skills rather than just telling people about them. It's a way of bringing your CV to life and introducing yourself to the world. As you learn more skills and put these into practice, each project that you complete will become more efficient and eye-catching.

This application series offers you the means to create the first project of your very own developer portfolio, allowing you to walk away from this course not only with a certificate but, more importantly, with a headstart into your career!

## A BRIEF RECAP

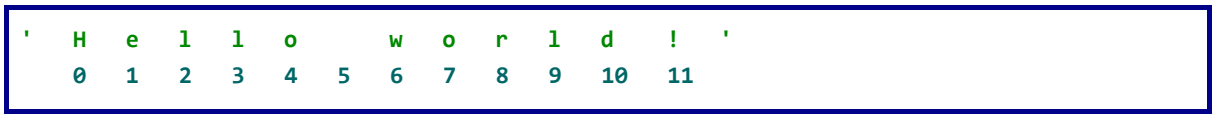
A code example that serves as a recap on how to deal with file input and output, which will be needed for this project, can be found below:

```
# Write a file
out_file = open("test.txt","w")
out_file.write("This Text is going to out file\nLook at it and see!")
out_file.close()

# Read a file
in_file = open("test.txt","r")
text = in_file.read()
```

```
in_file.close()
print(text)
```

Also, remember that you can think of the string 'Hello world!' as a list — each character in the string as an item with a corresponding index.




The space and exclamation mark are included in the character count, so 'Hello world!' is 12 characters long, from 'H' at index 0 to '!' at index 11.

```
string = "Hello"
print(string[0]) # H
print(string[1]) # e
print(string[2]) # l
print(string[3]) # l
print(string[4]) # o
```

Remember that if you specify an index, you'll get the character at that position in the string. You can also **slice strings** by specifying a range from one index to another, the starting index is included and the ending index is not.

**Note** that slicing a string does not modify the original string. You can capture a slice from one variable in a separate variable. Try typing the following into the interactive shell:

```
new_string = 'Hello world!'
fizz = new_string[0:5]
print(fizz)
```

By slicing and storing the resulting substring in another variable, you can have both the whole string and the substring handy for quick, easy access.

The most common of string methods are (where *s* is the variable that contains the string we are working with):

- **s.lower()** and **s.upper()** — convert a string to either uppercase or lowercase.
- **s.strip()** — removes any whitespaces from the beginning or end of a string.
- **s.find('text')** — searches for a specific text and returns its position in the string you are searching. If the string isn't found, -1 is returned.

- **s.replace('oldText', 'newText')** — replaces any occurrence of 'oldText' with 'newText'.
- **s.split('word')** — breaks down a string into a list of smaller pieces. The string is separated based on what is called a *delimiter*. This is a string or char value that is passed to the method. If no value is given, it will automatically split the string using whitespace as the delimiter. For example, if *s = "Hello"* and we execute *s.split('e')*, the program will output the list *['h', 'llo']*.

Some useful escape sequences are listed below:

- **\n** - Newline
- **\t** - Tab
- **\s** - Space

## THE TASK AT HAND

You will add further functionality to this Capstone Project later in this level. This part of the project will focus on working with files and string manipulation. You will also have to use conditional logic and loops in this task.

Before you begin:

A key focus of this project will be ensuring that your code is correct and adheres to the PEP 8 style guide. In this regard, make sure that you do the following before submitting your work:

1. Make sure that you have identified and removed all syntax, runtime and logical errors from your code.

2. Make sure that your code is readable. To ensure this, add comments to your code, use descriptive variable names and make good use of whitespace and indentation.

3. Make sure that all output that your program provides to the user is easy to read and understand. Labelling all data that you output (whether in text files or to the screen) is essential to make the data your program produces more user-friendly. For example, compare the readability of the outputs in the images below. Notice how using spacing and labelling the output makes the second output much more user-friendly than the first:

Output 1:

```
admin, Register Users with taskManager.py, Use taskManager.py to add the usernam
es and passwords for all team members that will be using this program., 10 Oct 2
019, 20 Oct 2019, No
admin, Assign initial tasks, Use taskManager.py to assign each team member with
appropriate tasks, 10 Oct 2019, 25 Oct 2019, No
```

Versus Output 2:

```
----------------------------------------------------------

Task:                   Assign initial tasks
Assigned to:            admin
Date assigned:          10 Oct 2019
Due date:               25 Oct 2019
Task Complete?          No
Task description:
 Use taskManager.py to assign each team member with appropriate tasks
----------------------------------------------------------
```

# Compulsory Task Part 1

Follow these steps:

- In this task, you will be creating a program for a small business that can help it to manage tasks assigned to each member of the team. You will extend this task in the next Capstone project. Create a Python file called **task_manager.py** in this folder.

- This program will work with two text files, **user.txt** and **tasks.txt**. Open each of the files that accompany this project and take note of the following:

    - **tasks.txt** stores a list of all the tasks that the team is working on. Open the **tasks.txt** file that accompanies this project. Note that this text file already contains data about two tasks. The data for each task is stored on a separate line in the text file. Each line includes the following data about a task in this order:
        - The username of the person to whom the task is assigned.
        - The title of the task.
        - A description of the task.
        - The date that the task was assigned to the user.
        - The due date for the task.
        - Either a 'Yes' or 'No' value that specifies if the task has been completed yet.

    - **user.txt** stores the username and password for each user that has permission to use your program (**task_manager.py**). Open the **user.txt** file that accompanies this project. Note that this text file already contains one default user that has the username, 'admin' and the password, 'adm1n'. The username and password for each user must be written to this file in the following format:
        - First, the username followed by a comma, a space and then the password.

        - One username and corresponding password per line.

- Your program should allow your users to do the following:

    - Login. The user should be prompted to enter a username and password. A list of valid usernames and passwords are stored in a text file called **user.txt**. Display an appropriate error message if the user enters a username that is not listed in **user.txt** or enters a valid username but not a valid password. The user should repeatedly be asked to enter a valid username and password until they provide appropriate credentials.

      The following menu should be displayed once the user has successfully logged in:

```
Please select one of the following options:
r - register user
a - add task
va - view all tasks
vm - view my tasks
e - exit
```

    - If the user chooses *'r'* to register a user, the user should be prompted for a new username and password. The user should also be asked to confirm the password. If the value entered to confirm the password matches the value of the password, the username and password should be written to **user.txt** in the appropriate format.

    - If the user chooses *'a'* to add a task, the user should be prompted to enter the username of the person the task is assigned to, the title of the task, a description of the task and the due date of the task. The data about the new task should be written to **tasks.txt**. The date on which the task is assigned should be the current date. Also assume that whenever you add a new task, the value that indicates whether the task has been completed or not is 'No'.

- If the user chooses *'va'* to view all tasks, display the information for each task on the screen in an easy to read format.

- If the user chooses *'vm'* to view the tasks that are assigned to them, only display all the tasks that have been assigned to the user that is currently logged-in in a user-friendly, easy to read manner.

## Compulsory Task Part 2

- Now format your program so that:
  - Only the user with the username 'admin' is allowed to register users.
  - The admin user is provided with a new menu option that allows them to display statistics. When this menu option is selected, the total number of tasks and the total number of users should be displayed in a user-friendly manner.

Rate us

**Share your thoughts**

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

**Click here** to share your thoughts anonymously.