Goal

This task gives you the opportunity to create a program implementing a given real-world problem. You will design using pseudocode, then develop a program using Python, and test with a formal test plan. Being able to use this process to create and implement Python programs is a valuable skill and will help you prepare for the external examination and certification in this course.

|  |  |
| --- | --- |
| **Format/Product** | Artefact – technical and scientific |
| **Type** | Individual |
| **Weighting** | 50% |
| **Length** | Program length can vary. No minimum or maximum length. |
| **Course learning outcomes (CLOs) assessed in this task** | 1, 2, 3 |
| **Due date** | Friday of Week 9 at 4 pm (AEST) |

Assessment criteria

This assessment will measure your ability to demonstrate:

* analysis of the given problem and appropriate report structure
* creative approaches to solving the problem
* application of relevant programming concepts
* correctness of the program and output, and program documentation

Task Instructions

You can choose **one (1) of the three** problems below. You are required to create a program that performs your solution. For those students who are looking for higher grades such as Distinctions or High Distinction, you should aim to implement 1 or more of the advanced options.

Program Options

1. Spot the Phishing email

**Overview:**

Implementing cybersecurity often involves automating processes to help spot problems. This program aims to look at suspicious emails and determine if they could be a problem.

**Standard Features:**

1. Read in a setup file of suspicious words and phrases. This can be a text, csv, xl file. The file should have 3 columns, the first is the start of the segment of the email you will be checking, the second column is the end of the segment and the third column should contain the word/phrase to check. For the standard feature you should only have to search the “<body” segment. See sample file - [ICT703 - Task 2 - Sample setup file-1.xlsx](https://learn.usc.edu.au/courses/31174/files/2503842?wrap=1)[Download ICT703 - Task 2 - Sample setup file-1.xlsx](https://learn.usc.edu.au/courses/31174/files/2503842/download?download_frd=1) (please see attached)
2. Read in an email as a file (use .eml files as they are text based and easier to understand). You decide the data structure to store the email. See sample email file - [ICT703 - Test email message 1.eml](https://learn.usc.edu.au/courses/31174/files/2489543?wrap=1)[Download ICT703 - Test email message 1.eml](https://learn.usc.edu.au/courses/31174/files/2489543/download?download_frd=1) (please see attached)
3. Search the segment of the email (within the html tags) looking for suspicious words or phrases
4. Print out the email segment, including the line number that the segment starts
   1. If there is a suspicious phrase, then identify the phrase and provide a failure message.
   2. If here is no suspicious phrase, then provide a pass message
5. You should create your own emails to test (either saving an email and editing or editing the sample one.

**Advanced Features:**

1. Expand the setup file by adding more columns to determine whether the segment is a single or multiple line segment or if it is a html segment. Add more segments like x-ms-exchange-crosstenant-id or x-ms-exchange-crosstenant-userprincipalname: and then add phrases (like XFCjGeORft8x7Ol)
2. Store any the failure emails in a database (like SQL or NoSQL) with details of the problem and where it occurs.  Have the option to process this database and print out the number of occurrences that a problem occurs.

2. Cyber Security Training System

**Overview:**

A key defence strategy in cybersecurity is the implementation of staff training in areas that might impact them (for example, phishing emails). From an administration perspective, it is key to understand who has done training and who has not yet completed training. A system like this should add new training courses and provide reports on who has done/not done training.

**Standard Features:**

1. User management
   1. Users login to system to complete training.
   2. Users can complete training. For this exercise the user only needs to answer a question and then they are deemed to have completed the course. Mark the course as completed (with a date and time). In the future you might add a training module.
   3. Users can view training they have not yet completed or training they have completed. Sort these by date and time.
2. Administration
   1. Create a special user (username 'admin' and password 'secret') that will be used to add training modules and report on current use.
   2. Admin can add a training module record. The training module would consist of a title, length and a quiz question and answer. This would be marked as not complete for all users and added to their data.
   3. Admin can print a list of users.
   4. Amin can print the courses completed (in date/time order) or not yet completed for a given user.
3. File storage
   1. Store all user information in a file that is read on startup and written on exit.

**Advanced Features:**

1. Create GUI front end.
2. Use and external database to store information (like SQLite).

3.    eCommerce Picking.

**Overview:**

The goal of this program is to manage the inventory of an eCommerce shop. You should decide what your product range will be - it could be sports, food, grocery, etc.

**Standard Features:**

1. Read from a file a list of products, name, price, location and inventory level. The location should contain the aisle number, the shelf number and then the bin number on that shelf (for example 12-3-2 is aisle 12, shelf 3 and bin 2)
2. After providing instructions, you should allow a shopper to purchase as many items as they wish, specifying the quantity of each. You need to check if there is enough inventory and update. Once the customer is finished you should gather their details (name, address).
3. Create a picking slip for the warehouse person that specifies the location, product, and quantity to pick. This should be in order of the aisles, shelf, etc. not as they were entered. This should also have the customer's details to deliver the goods.
4. Create an invoice for the customer which contains the product list, prices, quantities and the total cost to the customer.
5. Write the updated inventory file back out to a file.

**Advanced Features:**

1. Create GUI front end.
2. Use and external database to store information (like SQLite).

Resource

For information on testing your program with a formal test plan, refer to Module 8 Topic 2: How to build a test plan: [Module 8 - Topic 2 - Testing](https://learn.usc.edu.au/courses/31174/modules/items/758395)

Referencing guidelines

Please use APA 7th edition as your referencing style. For more information, see the [Referencing Guides and Academic Integrity page](https://www.usc.edu.au/current-students/student-support/academic-and-study-support/referencing-guides-and-academic-integrity/apa).

The reference list is not included in the word count. In-text citations are included in the word count.

Submission format

You should submit two documents:

* 1. Report **(20 points)**

Your report is to be presented as a professional business report. It must be submitted as a separate **Word document** (not pdf) with the following structure:

* + - 1. Title page
      2. Table of contents
      3. Introduction
      4. Problem description (what your program is and does, and any limitations)
      5. Design of your solution. This can include pseudocode, flow chart or structured discussion.
      6. Test plan. Discuss your test cases and the results you expected/achieved.
      7. List of references (if used)
  1. Python program **(30 points)**
     + The working Python program should be submitted as a single Python text file. If you have testing files or configuration files, you can compress all files into a zip folder and submit.
     + You should include documentation throughout the program.
     + If your Python program opens a file, please ensure it is opened from the same folder as your program.

In keeping with University of the Sunshine Coast policy, **all assignments are to be submitted in Canvas via Turnitin.**

To be accepted, your assessment submission *must* generate a similarity score (you are responsible for checking this). Submitting in Word is the best way to do this. If your submission does not generate a similarity score, it cannot be checked for plagiarism and therefore *will not be marked*.