|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task:** | | **6** | | |
| **Task Title:** | | **Portfolio** | | |
| **Task Code:** | | **AT2 POR-Task-6** | | |
|  | |  | | |
| Assessment type (): | | | | |
|  | Questioning (Oral/Written) | |  | Portfolio |
|  | Practical Demonstration | |  | Project |
|  | 3rd Party Report | |  | Other – Please Specify |

|  |
| --- |
| The base requirements this assessment task include:   * Web server, Python interpreter and database server * IDE or editor for developing Python programs (only PyCharm supported by the college) * Raspberry Pi with SenseHat or other IoT devices, like Arduino Uno or ESP32 * Access to Office 365 & Microsoft Word * Account with Adafruit.io * Report Template (Portfolio: AT2 POR-Task-4 Template) as/if supplied   Use of some of these items may not occur in this part of the assessment task. |
| Assessment Due This assessment is split into components that have several due dates:   * Week 07 17:00 (5:00PM) on the day of the scheduled lecture. * Week 09 17:00 (5:00PM) on the day of the scheduled lecture. * Week 12 17:00 (5:00PM) on the day of the scheduled lecture. * Week 14 17:00 (5:00PM) on the day of the scheduled lecture.   Refer to Blackboard for most accurate dates, which may alter due to unforeseen circumstances.  We also will endeavour to update these document(s) at the same time. |
| Instructions Follow the steps listed in this assessment item.  Submission of the documentation, code, and associated items is at the end of each part of the portfolio.  Each part of the portfolio has a deadline for submission.  It is advantageous to you to attempt to meet the deadline provided. |
| Important If you are using a different configuration of tools and equipment for this assessment item, then assistance in this and subsequent parts of the portfolio to ensure the systems work correctly will be limited. |
| Scenario You are currently working for a small Perth-based start-up company called Incredibly Obvious Technologies.  They are looking to create a presence in the home automation and monitoring market.  Download and read the complete “Portfolio-Task-Scenario” from Blackboard. |
| General Instructions We provide a document template for your answers.  Download the AT2 POR-Task-4 Template from Blackboard, and then DOUBLE CLICK it to create a new blank document for your answers.  Save the file as:   * XXX-POR-Task-4.docx   Replacing the XXX with your initials.  For example, Adrian Gould would use AG-POR-Task-4.docx for his submitted filename. |
| Answering Questions When a step includes a question, you must attempt to answer it.  There is a minimum and maximum number of words to use for each answer.  If a step has more than one question, these maxima and minima are a total for all the questions in that specific step.  All answers must be in complete sentences unless indicated.  If required, make sure to add any code you’ve written in a separate file to your submission. DO NOT put code in a Word document. |
| Sources of Information In industry, it is good practice to keep track of where information was obtained. This is especially true if it is a written document, or even code.  If you answer any questions using information from web sites, please include the site name and URL (Web site address) after the answer. Likewise, include the title and author for books and magazine articles. For example:   * RS Electronics Ltd: <https://au.rs-online.com/> * Slack API Documentation, Users List Method: <https://api.slack.com/methods/users.list>  Code Storage We advise that you create a GIT repository on GitHub and use this to store a copy of your work.  You may also use OneDrive within your college Office365 to store a backup of your code or keep a copy on a USB thumb drive. |
| *This space left intentionally blank.* |

|  |  |  |
| --- | --- | --- |
| **STEP** | **Task to perform** | Words Min/Max |
| 00 | Create Evidence Document Make sure you have followed the instructions on creating the answer document, as given in the General Instructions.  Familiarise yourself with the content and document your progress in this assessment.  Make sure that you complete the title page of the document.  If the question requires you to submit code with your final submission, you may add it as a zip-file **OR in a separate folder**. |  |
|  | *This space left intentionally blank.* |  |
| 01 | Standard cryptographic algorithms Research cryptographic algorithms and describe, in your own words, at least three modern cryptographic algorithms and the key differences between them. | 30-80 |
|  | The definition of cryptographic algorithms is the mathematical equation used to scramble the data to make it unreadable. It secures information, ensure privacy and authenticate data.  3 examples of modern cryptographic algorithms are hash functions, symmetric-key algorithms and asymmetric-key algorithms.  Hash function is an algorithm that converts a value into a fixed-sized string of bytes, which is typically a sequence of numbers and letters. The output is called hash value or hash. Even a small change in the input value will produce a significantly different hash value. It is often used for data integrity verification and digital signatures.  Symmetric-key algorithm is used for encryption and decryption. As it says in the name, the sender and the receiver both share the same secret key. It is more computationally efficient than asymmetric-key algorithms, making it a better choice for encrypting large amounts of data.  On the other hand, asymmetric-key algorithm uses a pair of keys, a public key and a private key. The public key is used for encryption but the private key is kept secret and used for decryption. As we have a pair of keys, it offers stronger security compared to symmetric-key algorithm. |  |
| 02 | Prepare file to encrypt and decrypt In this assessment, you will be encrypting and decrypting a file that is unique to you and is timestamped.  Follow these instructions:  Go to the url **https://www.qr-code-generator.com/**.  *Text  Description automatically generated*  Click on  Enter the following information:  **Your Full Name/Student ID/Date**  The date must be the current date formatted as YYYY-MM-DD. For example, student John Smith with Student ID J123456 who does his assessment on 15 October 2021, would enter the following information:  John Smith/J123456/2021-10-15  *Qr code  Description automatically generated*  On the right side, enable the ‘SCAN ME‘ icon.  The web page should look like this:  *Graphical user interface, text, application, chat or text message  Description automatically generated*  **Important: DO NOT put a URL in the QR code!**  **Doing so will be an instant fail for this assessment.**  *A green logo with white text  Description automatically generated with low confidence*  Next, click the button and the image will automatically download to your computer (be patient, it may take a few seconds).  Please note that the downloaded file is actually a PNG image. The name of the downloaded file (at the time of writing) is **frame.png**.  Keep this file safe as you will need it in the following steps.  Paste a screenshot of your QR code below. | n/a |
|  |  |  |
| 03 | Download the project folder structure For this portfolio task, you can download an existing project. It contains the scaffolding for encrypting and decrypting data. It is incomplete and therefore not yet functional.  The folder structure looks like this:  Graphical user interface, text, application  Description automatically generated  Follow these instructions:  Download the zip-file from Blackboard and unzip it. After unpacking the zip-file, you will have to create a new Virtual Environment. You may also want to enable VCS Integration (Git).  Set up your project so you can run the unit tests (and run them). Answer the following questions:   * How many unit tests ran? * How many unit tests failed and how many passed? * IF any passed, can you explain why?   Before you continue to the next step, browse the project, and get a feel for what is in there and what might be missing. Use the unit tests to your advantage in the next steps. | 30-60 |
|  | Answer the following questions:   * How many unit tests ran? * How many unit tests failed and how many passed? * IF any passed, can you explain why? * Total of 7 unit tests ran. * 5 failed, and 2 passed. * The 2 unit tests that passed are the tests that are trying to verify that the objects can be instantiated successfully using a plaintext key. But the tests will always pass, regardless of whether the objects instantiation succeed or fail.   They do have except clause where if the conversion goes wrong, it catches an error, and the tests will fail, but instantiating the Decryptor and Encryptor objects does not involve any conversion operations that could potentially raise the binascii error. |  |
| 04 | Filling in the gaps The ultimate goal for this assessment is the ability to easily encrypt (and decrypt) small-ish files.  For this, we will use a library from the cryptography package that performs symmetric cryptography.  Follow these instructions:  Install the package **cryptography** in your project using **pip**.  Read the documentation about the Fernet implementation.  Update the requirements.txt by running the following command (in a terminal):  pip freeze > requirements.txt  Make sure to run the command in the appropriate directory.  If you have set up VCS integration, commit the file. | n/a |
|  |  |  |
| 05 | Fix the broken unit test One of the unit tests is broken because the exception type **InvalidToken** is not known.  Follow these instructions:  Use the online documentation for Fernet to write the correct import statement so you don’t have to change any of the code in the unit test **test\_decryption.py**.  Run the unit tests and observe what happens. Did anything change? If so, what? If not, why? | 20-50 |
|  | Fixed the import error,   Now InvalidToken is known, but the test still does not pass.  The test uses decrypt method of Decryptor class, and the decrypt method uses a helper function called \_decrypt. \_decrypt always raises NotImplementedError, so the decrypt method will always raise the error when used. |  |
| 06 | Let’s encrypt! (1) The provided framework is still missing some parts, which are clearly marked with appropriate comments.  We will focus on getting encryption working first by filling in more of the gaps.  Follow these instructions:  Examine the **Encryptor** class, specifically the initialiser method **\_\_init\_\_()**.  The following snippet from that method prepares the key to be used seamlessly with the Fernet class.  def \_\_init\_\_(self, key):  key32\_ = key32(key)  key64\_ = key64(key32\_)  The key is, obviously, not stored anywhere.  You now have a choice to make:   1. Use the key to instantiate the object that will do the actual encryption and store that object in an instance variable. 2. Simply store the prepared key (the value of **key64\_**) in an instance variable.   You may need to consult the documentation again to guide your choice.  Depending on which option you choose, you will need to adjust one of the unit tests.  Don’t forget to add any **import** statement you may need. | n/a |
|  |  |  |
| 07 | Let’s encrypt! (2) You will now implement the method **\_encrypt()** in the **Encryptor** class.  It starts with an **\_**, which means it is considered to be a “protected” method. As you know, Python does not really have “protected” and “private” methods like other languages, like C#, do. Instead, this property is a convention.  Depending on your decision in the previous step, the implementation of this method will look different.  Follow these instructions:  Examine the method **\_encrypt()** closely. Pay attention to its “prototype”. What type of data does it accept and what does it return?  Also look at the related unit tests. What is expected there? Does it still match your code (based on your earlier decision)?  Implement the method in one of two ways:   1. Use the encryption object that you stored in an instance variable to encode the data and return the result, **or**: 2. Instantiate the encryption object using the key you stored in an instance variable to encode the data and return the result.   Ensure your unit tests are still valid. For example, what might happen if someone makes changes to the initialiser? Would your unit tests pick up on that?  Run the unit tests and examine the result. Did more tests run without errors?  Provide evidence of you running the unit tests, for example, by adding a few screenshots. | n/a |
|  |  |  |
| 08 | Let’s decrypt! You will now prepare the **Decryptor** class (the initialiser!) and implement the method **\_decrypt()** in the **Decryptor** class.  The steps are very similar to the ones for the **Encryptor** class.  Follow these instructions:  Finish the implementation of the initialiser method. You can make the same choice regarding the key as you did for the encryption in the previous two steps.  Implement the method **\_decrypt()**. Again, pay attention to its “prototype”. What type of data does it accept and what does it return?  Also look at the related unit tests. What is expected there? Does it still match your code (based on your decision)?  Ensure your unit tests are still valid. For example, what might happen if someone makes changes to the initialiser? Would your unit tests pick up on that?  Run the unit tests and examine the result. Did more tests run without errors?  Provide evidence of you running the unit tests, for example, by adding a few screenshots.  Fix the code so all unit tests run without problems. | n/a |
|  |  |  |
| 09 | Encrypt, decrypt a file The final step is to encrypt and decrypt the file you created in Step 2.  You can take guidance from some of the unit tests that use a **TestAdapter** class, which implements a single method called **read\_data()**. Please note that the method returns an object of type **bytes**, and so the data can be fed directly into the **encrypt** and **decrypt** methods.  Follow these instructions:  In the **main.py** file, create a new class called **FileAdapter**. Its initialiser method should accept a single argument: **filename**. Store the filename in an instance variable.  Add the method **read\_data()**. Implement the method by coding the following steps:   1. Open a file using the filename stored in the instance variable. 2. Read the data from the file and return the result.   Finally, add code to the **main.py** file that tests encryption and decryption of a file. Remember, you can simply pass the adapter object into the **encrypt** and **decrypt** methods as they know how to handle it.  Please note that you can use the same adapter class for encryption and decryption. Use **different filenames** for encrypted and decrypted data.  Compare the contents of the decrypted file against the original. If no errors occurred, they should be exactly the same.  Make sure you include all versions of the file you encrypted in your final submission to Blackboard: the original, the encrypted version, and the decrypted version. | n/a |
|  |  |  |
|  | Submission of Portfolio Work To submit the portfolio, do the following:   * Save the document with your answers as a MS Word file (.docx). * Put all your code inside a zip-file. * Open Blackboard, and locate the AT2 Portfolio Task 6 assessment * Open the assessment and upload the original word-processed document and the zip-file as separate files.   Click Submit. |  |

# Appendix A: Code Style Guidelines

Your code will follow the PEP 8 standard.

Readability Counts  
- Zen of Python

Explicit is better than implicit.  
- Zen of Python

Other code standards available in the Presentation, “Python Coding Standards for North Metropolitan TAFE”.