Coventry University

Student Assignment Brief

This document is intended for Coventry University Group students for their own use in completing their assessed work for this module. It must not be passed to third parties or posted on any website. If you require this document in an alternative format, please contact your Module Leader.

Contents

The work you submit for this assignment must be your own independent work, or in the case of group assignment your own groups' work. More information is available in the 'Assessment Task' section of this assignment brief.

Assignment Information

Module Name	Programming and Algorithms 2
Module Code	5062CEM
Assignment Title	Encoding and Decoding a Message Hidden Inside an Image
Assignment Due	19/02/2024 18:00
Assignment Credit	10
Word Count	N/A
Assignment Type	**Percentage Grade** (Applied Core Assessment): You will be provided with an overall grade between 0% and 100%. You have one opportunity to pass the assignment at or above 40%.

Assignment Task

Tasks are to be undertaken in the Python programming language.

You will be expected to include comments in your code to explain the behaviour of your code and provide a justification for your algorithm selection. You are also *strongly advised* to test your code for compilation on a system other than your own, prior to submission. **Non-compiling code will not pass, see the marking rubric for further information.**

You are required to create a Coventry University GitHub *private* repository to store your sourcecode and manage the version control of your work appropriately. Evidence of version control must be included in your regular commits to the repository over the period between the hand-out date and due date. **The repository you create must be in the 5062CEM organisation.** Your eventual submission will be via TurnItIn, and a link to the repository must be included.

The solution you provide will be a single, cohesive tool. Whilst they may be multiple sourcecode files, there should ultimately be a single `main.py` to run the entire tool. **You must ensure that you only provide sourcecode files in both the Git repository and submission document that you want to be marked.** Therefore, you must remove any code or files that you do not want marked.

Remember, you should be using functions and classes appropriately to construct your solution. Minimal marks will be given to solutions that do not utilise functions and classes.

The assignment has been split into two parts:

1. **Encoding**: required to encode an image with a piece of given text
2. **Decoding**: required to decode an encoded image to extract the text

You are expected to build a piece of software that enables the user to encode a given image with a piece of text. The user will

supply the text to be encoded (i.e. it can be inserted via the command line). Once the image has been successfully encoded with the text, a method to decode the image to retrieve the text will also be required.

Task 1: Encoding an Image with Text (40%)

For this task, you will need to encode an image with text. You are required to use the tools and techniques that have been taught in this module, i.e. `OpenCV` for image handling etc. The end-user will have the option to insert the filename and extension for an image of their choice, alongside the text they would like to encode within the image.

For example, the user may want to use to the filename `cat.png`, therefore, they will have an option to enter the filename in software you design (verifying that it is a valid image file). The end-user will then be asked what text they would like to encode. Your terminal window may look something like this:

```
$ Image Filename (with extension): cat.png
$ Encoded Text: I like cats.
```

Once these details have been passed through, the software will encode the text within the image file and create a *new* image file, with the suffix `_encoded`, i.e. `cat_encoded.png`. The whole purpose of this task is that the resulting image looks exactly the same as the input image, i.e. there are no visible changes can be seen in the image.

You may want to consider thinking outside the box to increase the security of this method. Providing additional features to enhance security will ensure you hit the higher grade boundaries of the assignment. Review the marking rubric attached to this document for further details.

Task 2: Decoding the Image (30%)

For this task, you are expanding upon the software you designed in *task one* and add a feature to decode the encoded image. The new feature should take the image path of the previously encoded image (should be entered via the terminal by an end-user) and then extract the text that was coded within it.

The extracted text will be displayed inside the terminal window, similar to the example below:

```
$ Extracted Text: I like cats.
```

Any additional security that you have included in *task one* must also be retrospectively removed in this task.

Task 3: Testing (10%)

For each function that you have implemented, you will need to write a test case to ensure it is working as expected. Testing should be performed using the `unittest` module and **all** tests should pass, unless you have purposefully created a test to *fail*.

There may be some aspects of the tool that are *untestable*, and therefore, it is expected that these functions will not be tested.

Note: It will be expected that the unit testing will be provided in a separate class file, and following the convention as shown in the lectures and labs from the first year programming module.

Task 4: Documentation (10%)

To aid the end-user in using your tool, commenting is relevant for your implementation. For each function and class that is present in your tool, you need to provide a description of the function and methodology of how it performs. Commenting *must* be provided in the form of **Docstrings** for the functions and classes. **Single-line comments** will be acceptable, however, only where they are necessary, i.e. explaining a difficult control/conditional statement.

You must provide reasoning behind the justification for the algorithm you have used in your solution. This will be presented in the README documentation.

Additionally, documentation will be required to explain how the tool works. Therefore, it is important to include a **README** file outlining how the tool works, a quick synopsis of functions that exist, and any test-cases that have been performed along with their outcomes. You may want to consider including a `requirements.txt` file to provide the necessary packages to run your script/code.

Task 5: Version Control (5%)

The purpose of this task is to ensure that you are developing the tool iteratively and collaboratively using relevant version control features, such as *adding*, *committing*, *pushing*, *branching* and *merging*. You may also want to consider using some web-interface

features such as *issue tracking* and the *Wiki* to aid in the development of your tool.

The repository must be created in the **5062CEM** Git organisation and named in the following convention: `<STUDENT_ID>_CW`, where you need to replace `<STUDENT_ID>` with your own student identification number. For example, your repository name will look like: `123456789_CW`.

Any work submitted outside the Git organisation will not be marked. You must check that you have access to the Git organisation and any problems with access you must contact the module leader.

Task 6: Submission Guidelines (5%)

You are required to submit your work using the TurnItIn platform on Aula. You will be submitting a Microsoft Word document (`docx`) or a Portable Document Framework file (`pdf`).

Inside the document, you will need to provide the following:

- The **URL** of the Git repository stored on the *Coventry University* Git service.
- For each source-code file of your submission, you need to provide:
  - a title page, which is the name of the file, i.e. `main.py`
  - the source-code of the file **with** syntax highlighting and tabs/spacing intact

Note: Screenshots will not be accepted, and the code must be copied and pasted into the document.

A submission example can be found at the following links:

- Microsoft Word Submission Example
- PDF Submission Example

You will either be awarded zero marks, or the full marks, depending on how you follow the guidelines.

Your source-code will be submitted to a plagiarism checker, so please ensure that any source-code acquired online is appropriately referenced. You are not to push or commit code to the GitHub repository after you have submitted your coursework. Timestamps will be checked, and if any changes made after the submission, timestamp will not be marked.

Marking and Feedback

How will my assignment be marked?

Your assignment will be marked by by the module team. Provisional grades will be released by the module team upon completion. Feedback will be provided by the module team alongside grades release

Details of the marking criteria for this task can be found at the bottom of the assignment brief.

How will I receive my grades and feedback?

Provisional marks will be released once internally moderated.

Feedback will be provided by the module team alongside grades release. You will be able to access your feedback via the TurnItIn Platform.

Your provisional marks and feedback should be available within 2 weeks (10 working days).

What will I be marked against?

Details of the marking criteria for this task can be found at the bottom of this assignment brief.

Assessed Module Learning Outcomes

The Learning Outcomes for this module align to the marking criteria which can be found at the end of this brief. Ensure you understand the marking criteria to ensure successful achievement of the assessment task. The following module learning outcomes are assessed in this task:

1. Understand algorithm efficiency in order to select and implement the most appropriate for a given task.
2. Evaluate patterns and paradigms appropriate for specific tasks.

Assignment Support and Academic Integrity

If you have any questions about this assignment or require information on Extenuating Circumstances, please see the Student Guidance on Coursework for more information.

Spelling Punctuation and Grammar

You are expected to use effective, accurate, and appropriate language within this assessment task.

Academic Integrity

The work you submit must be your own, or in the case of groupwork, that of your group. All sources of information need to be acknowledged and attributed; therefore, you must provide references for all sources of information and acknowledge any tools used in the production of your work, including Artificial Intelligence (AI). We use detection software and make routine checks for evidence of academic misconduct.

Definitions of academic misconduct, including plagiarism, self-plagiarism, and collusion can be found on the Student Portal. All cases of suspected academic misconduct are referred for investigation, the outcomes of which can have profound consequences to your studies. For more information on academic integrity please visit the Academic and Research Integrity section of the Student Portal.

Support for Students with Disabilities or Additional Needs

If you have a disability, long-term health condition, specific learning difference, mental health diagnosis or symptoms and have discussed your support needs with health and well-being you may be able to access support that will help with your studies.

If you feel you may benefit from additional support, but have not disclosed a disability to the University, or have disclosed but are yet to discuss your support needs it is important to let us know so we can provide the right support for your circumstances. Visit the Student Portal to find out more.

Unable to Submit on Time?

The University wants you to do your best. However, we know that sometimes events happen which mean that you cannot submit your assessment by the deadline or sit a scheduled exam. If you think this might be the case, guidance on understanding what counts as an extenuating circumstance, and how to apply is available on the Student Portal.

Administration of Assessment

Module Leader Name	Dr Ian Cornelius
Module Leader E-Mail	ab6459@coventry.ac.uk
Assignment Category	Individual Programming Assignment
Attempt Type	Normal
Component Code	CW1

Assignment Marking Criteria

Task	Fail	Third	Lower Second	Upper Second	First	First +
1	No attempt was made. **Or**, the code does not compile.	An image was able to be successfully loaded using the OpenCV framework. Hard-coded strings have been defined for encrypting within the image. There has been no attempt at encoding the hard-coded string inside the image.	An image chosen by the end-user was able to be successfully loaded using the OpenCV framework. Hard-coded strings have been defined for encrypting within the image. There has an attempt at	An image chosen by the user was able to be successfully loaded using the OpenCV framework. User-defined strings have been	An image was able to be successfully loaded using the OpenCV framework. User-defined strings have been captured from the end-user for encrypting within the image. The software is able to	All necessary components of the task have been implemented beyond expectations. There are additional thoughts and processes in place for their implementation.

		There are no functions being used in the source code. There are no classes being used in the source code.	encoding the hard-coded string inside the image. However, it does not function as intended, i.e. the image is malformed. There are some functions being used in the source code. There are no classes being used in the source code.	captured from the end-user for encrypting within the image. The software is able to successfully encode the string within the image. There is good use of functions being used in the source code. There is some use of classes being used in the source code.	successfully encode the string within the image. Further functionality has been provided, such as: ◦ the ability of using a custom key to encrypt the string within the image has been provided. ◦ moving the encoded text, so it is not located the first few bytes of an image There is excellent use of functions being used in the source code. There is excellent use of classes being used in the source code.	
2	No attempt was made. **Or**, the code does not compile.	An image was able to be successfully loaded using the OpenCV framework. There has been no attempt at decoding the message hidden inside the image. There are no functions being used in the source code. There are no classes being used in the source code.	An image chosen by the end-user was able to be successfully loaded using the OpenCV framework. The string has not been able to be successfully decoded from the image, but an attempt was made. There are some functions being used in the source code. There are no classes being used in the source code.	An image chosen by the user was able to be successfully loaded using the OpenCV framework. The string was successfully decoded from the image and displayed to the end-user. There is good use of functions being used in the source	An image chosen by the user was able to be successfully loaded using the OpenCV framework. The string was successfully decoded from the image by using the custom key provided in the encryption process. i.e. the key was extracted from the image, and used to decipher the message. The decoded message is displayed to the end-user, along with the key used	All necessary components of the task have been implemented beyond expectations. There are additional thoughts and processes in place for their implementation.

				code. There is some use of classes being used in the source code.	for encryption. There is excellent use of functions being used in the source code. There is excellent use of classes being used in the source code.	
3	No attempt was made. **Or**, the code does not compile.	There is some evidence of testing occurring; however, it is not unit-testing.	There is a collection of test cases, but it does not cover the obvious areas of the project.	There are a good range of test cases for each function in the project, using a variety of different assertions. The unit test cases are provided in a separate class file.	There is a full test-suite, with extensive coverage. In some cases, there is evidence that other types of testing are being utilised, i.e. integration testing.	All necessary components of the task have been implemented beyond expectations. There are additional thoughts and processes in place for their implementation.
4	No attempt was made.	Single-line comments have been provided, albeit they are not descriptive about what is happening inside the source code. There are no Docstrings provided. There is no justification on algorithm efficiency and the choice of algorithm. There is no README document for the end-user.	Single-line comments have been provided, albeit they are not descriptive about what is happening inside the source code. Docstrings have been provided, albeit they are not descriptive about the functionality of the functions. There is no justification on algorithm efficiency and the choice of algorithm. There is a limited README document for the end-user, providing instructions on how to run the tool.	Single-line comments have been provided, they are descriptive about what is happening inside the source code. Docstrings have been provided, they are descriptive about the functionality of the functions and classes. Justification on algorithm efficiency and the choice of algorithm is present.	Single-line comments have been provided, they are descriptive about what is happening inside the source code. Docstrings have been provided, they are descriptive about the functionality of the functions and classes. Justification on algorithm efficiency and the choice of algorithm is present. There is a README document for the end-user, providing instructions on how to run the tool and a list of requited modules	All necessary components of the task have been implemented beyond expectations. There are additional thoughts and processes in place for their implementation.

				There is a README document for the end-user, providing instructions on how to run the tool and a list of requited modules for the tool to run.	for the tool to run. The README document contains test cases on the unit tests that have been performed. The README document provides an overview of the functions in the tool. A `requirements.txt` file has been provided with a list of modules for the tool to work.	
5	No attempt was made.	Some evidence of version control; however, there is no evidence of iterative development. Some commentary has been provided in the commit messages, but it is little or not clear on the changes that have been made. There is no use of the Git web services.	The repository is organised, with source-code and test-cases separate. However, it does not utilise the version control features well. There are clear comments in the commit messages, albeit not that in-depth. There is no use of the Git web services.	The repository is organised, with source-code and test-cases separate. Comments are clear and descriptive between commits. There is evidence of branching being utilised, with new features developed off the main branch. There is some use of the Git web services such as issue tracking.	The repository is organised, with source-code and test-cases separate. Comments are clear and descriptive between commits. There is evidence of branching being utilised, with new features developed off the main branch and merged once the new feature has been implemented. There is good use of the Git web services such as issue tracking and the Wiki.	All necessary components of the task have been implemented beyond expectations. There are additional thoughts and processes in place for their implementation.
6	Did not follow the submission guidelines.					Followed the submission guidelines.