|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task:** | | **2** | | |
| **Task Title:** | | **Portfolio** | | |
| **Task Code:** | | **AT2 POR-Task-2** | | |
|  | |  | | |
| 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 * Access to Office 365 & Microsoft Word * Report Template (Portfolio: Part 3 Document Template) as supplied   Use of some of these items may not occur in this part of the assessment task. |
| Assessment Due This assessment is due on the following date:   * Week 14 17:00 (5:00PM) on the day of the scheduled lecture.   Refer to Blackboard for the 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. Please note that **additional** information may be given.  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 / background In this Portfolio Task, we’re going to have a closer look at the four pillars of OO:   1. Abstraction 2. Polymorphism 3. Inheritance 4. Encapsulation |
| General Instructions We provide a document template for your answers.  Save the file as:   * XXX-IoT-Port-Part-3.docx   Replacing the XXX with your initials.  For example, Adrian Gould would use AG-IoT-Port-Part-3.docx for his submitted filename.  Upload any code as a PyCharm project in a zip-file. Remove the virtual environment (**venv** or **.venv**) from the zip-file before uploading it to Blackboard. Include your .git file with your submission. Your .git file must point to an origin on GitHub. |
| 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. |
| 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>   If you use ChatGPT (or any other AI) to help you answer the questions, you must reference the chat as a conversation (e.g. 2023, Conversation with ChatGPT) . Copying/pasting from ChatGPT or any other LLM without attribution is plagiarism. Your answer must be in **your** own words to be accepted. If you use ChatGPT to proof your answer, include “Proofed by ChatGPT” with your answer. Your lecturer may ask follow up questions to validate your understanding. Code Storage The initial source code for this project is available on GitHub. You must fork the repository and provide evidence of using Git. See for details:  https://github.com/NM-TAFE/civ-ipriot-smiley/blob/main/README.md |
| *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.  At any stage during this assignment, you may consult the stakeholder(s) or their representative(s). |  |
|  | *This space left intentionally blank.* |  |
| 01 | Download the “Smileys” project You can download the Smileys project from GitHub using the provided link. .  After cloning, the directory structure should look something like this:  Table  Description automatically generated  The **main.py** script is the entry point for the project. You should be able to run the script on your Raspberry Pi by typing **python3 main.py** in a terminal. Make sure to **cd** into your project directory (this means Change Directory).  You may ignore the **requirements.txt** for now. The file **README.md** contains a brief description of what the project is about. It is good practice to add a README to any project you create.  Take a screenshot of the result of the following two commands and put it in the answer document ($ is the prompt; don’t type that):  **$ ls $ python3 main.py** | n/a |
|  | *This space left intentionally blank.* |  |
| 02 | Find the classes The project contains a few classes. Some are “base classes” (sometimes called super classes) and some are “subclasses”, which are derived from the base classes.  Answer the following questions:   1. How many classes can you identify in the project? I can identify five classes: Blinkable, Happy, Sad, SenseHat, and Smiley. 2. In your own words, describe how ‘abstraction’ is visible in this project.   The “Blinkable” class is an abstract class that carries the abstract method “blink”. The blinkable class demonstrates abstraction by hiding the underlying details of the function within it’s own class – subclasses can then use a generalised version by simply calling the blink method without worrying about the underlying code (and different subclasses can also use the method in different ways).   1. Describe which of these classes are subclasses and which are base (or super) classes.   Blinkable, Smiley and SenseHat are base classes. Sad is a subclass of Smiley and Happy is a subclass of both Smiley and Blinkable.   1. What is the name of the process of deriving from base classes?   This is called “inheritance”. | n/a |
|  |  |  |
| 03 | Not all classes are the same Investigate the classes **Happy** and **Sad**.  Notice the similarities and the difference. Be careful, what may look like a similarity may actually contain a subtle but important difference.  In your own words, describe:   * The commonalities between **Happy** and **Sad** * The differences between **Happy** and **Sad** * Is there anything that stands out as a difference? | n/a |
|  | Both classes are subclasses of Smiley and share the draw\_mouth and draw\_eyes functions that “draw” the features of the face. However, what immediately stands out is that Happy is also a subclass of the abstract class Blinkable (of which Sad is not). This allows it to utilse the blink function from that class by manipulating the Boolean values in draw\_eyes() that open and close the eyes. Sad presently has no way to change these values so the eyes will always remain open. Finally, due to the “mouth” being different for both faces, although they both use draw\_mouth(), they overrise the function with unique mouth values for each (which changes the shape of the mouth expression in the LED display).  *This space left intentionally blank.* |  |
| 04 | Where’s the Sense(Hat) in that? You will have noticed by now that the project uses the SenseHat to display the smileys on the display.  Yet, in **main.py** there is no mention of any SenseHat at all.  Answer the following questions:   1. Where is **SenseHat** used (in which **class**)?   SenseHat is used in the Smiley class by encapsulating the SenseHat object.   1. Which functionalities of **SenseHat** are used?   SenseHat is used to display pixels on the LED matrix (displaying a smiley face) as well as using the low light function in the dim display function to manipulate the light intensity of the LED display.   1. What is the process of storing and potentially hiding objects in classes called?   This is called encapsulation.   1. Draw a simple Class Diagram that shows the class **SenseHat** and the containing class (the answer from a.). Make sure to use the correct relationship and multiplicities. | n/a |
| 05 | Sad smileys can’t blink (or can they?) Unlike the **Happy** smiley, the current implementation of the **Sad** smiley does not have the ability to blink.  Investigate the **Happy** smiley to see how blinking has been implemented by looking at the method **blink()**, which takes in one argument. The argument determines how long the blink lasts.  You will now add the ability to blink to the **Sad** smiley.  Follow the instructions (and answer any questions):   1. In your own words, how does the **blink()**method make the smiley blink?   The draw\_eyes method has a flag to make the pixels for the eyes yellow if the wide open Boolean value is set to false (thus making the eyes appear “closed”). The blink() method sets this flag to flip between true and false on a delay, thus changing between open and closed eyes and making the eyes appear to blink.   1. Create a new method called blink in the **Sad** class and ensure you use the same prototype (prototype = name + arguments):   **def** blink(**self**, delay=0.25):  **pass** # your implementation goes here 2. Implement the code that makes the smiley blink. You may use the implementation from **Happy** as guidance.      1. Test the code on your Raspberry Pi and watch the sad smiley blink its eyes. (You may have to adjust the **main.py** script for this.) |  |
| 06 | If it walks like a duck… If you followed the instructions in the previous question closely, you didn’t use the class **Blinkable** to make **Sad** blink. (If you did, that’s fine. In that case, however, you should assume you did not when answering the following questions.)  You did not have to use **Blinkable** to allow the **Sad** smiley to blink.  Answer the following questions:   1. What kind of class is **Blinkable**? Look at its super class for a hint.   It is an abstract base class.   1. Any class that uses **Blinkable** is said to “implement” it. What is another (generic) name for a class like **Blinkable**, which may be implemented by other classes? Another name for these types of classes is called an interface. 2. What is the answer from the previous question an example of? Choose from: Abstraction, Polymorphism, Inheritance, and Encapsulation.   Abstraction.   1. Why were you able to use it like the one defined in **Happy**, yet without using **Blinkable**?   The blink method in the sad class was implemented identically to the one used in Happy – in other words, it was able to use this method as it matched the signature and return type in the Blinkable abstract class.   1. Regarding the answer to d., what is this called and why does it work in Python but not in some other languages, like C#? (A hint is in the title of this question.)   This works in python due to “duck typing” – Python is dynamically typed unlike other languages, so all type checking is done at runtime. This means that functions are not bound to their class and other classes can use these methods without actually being an instance of or related to that class. As long as the class has the correct attributes to use the method, it can use that function – even if it’s not a “duck”, the program will recognise that it is meant to “walk like a duck”. |  |
| 07 | Does a smiley have to be yellow? You don’t have to answer the above question, because obviously it doesn’t. But keep reading.  Smileys that do not feel too well are often green, while angry smileys are usually red or orange.  The current implementation only allows for yellow smileys, though. That means we’re a bit limited as to what emotions we can express with our SenseHat Smileys.  Answer the following questions:   1. Which colours are **defined** and where?   The current colours (red, green, white, yellow, black) are defined by in smiley.py by giving variable names to the RGB colour values for each colour and are inherited by the sad and happy classes.   1. What is the name of the **type of variables** that hold the colours?   These are class variables belonging to the smiley class – each colour is represented by a string (white represented by “WHITE”, red by “RED” and so on, sans black which is “BLANK”).   1. Where are the colour variables actually **used**?   The colours are used through all the draw functions. Whenever a function “draws” a feature (such as draw mouth) by specifying the placement of pixels, it will also using the variables specify the pixel colour.   1. What would be an easy, albeit rather naive, way to change the colour of the smileys, for example, to green?   As all the colour values are given variable names (such as “ YELLOW = (255, 255, 0)”), you could simply change the RGB value of YELLOW to match the colour green instead (by making the first 255 a 0) which would carry through to all the methods/classes using the “YELLOW” variable. |  |
| 08 | Flexible colours – Step 1 Changing the colour of the smileys once is easy. But it’s not very flexible, is it?  To allow for smileys to have different colours, we best not hardcode any values in any class, which you discovered in question 7c.  Some classes have a built-in assumption about the colour that is used. Let’s remove that assumption one step at a time.  Follow these instructions:   1. Added an instance method called **complexion** to the class **Smiley**. You can make it return **self.YELLOW**.  (Technically speaking, smileys don’t have a complexion as they have no skin, but it sounds a bit nicer than just “colour”.)      1. Update the subclasses that use the colour variable directly to, instead, use the new method **complexion**.      1. Which of the four pillars of OO have we used here: Abstraction, Polymorphism, Inheritance, or Encapsulation? (If you feel more than one applies, pick the one that applies **best**.)   Encapsulation – we are hiding the information about the complexion colour within the complexion method, and instead adding a method by which that information can be accessed.   1. Check that the new code works correctly. It should still just show a yellow smiley… |  |
| 09 | Flexible colours – Step 2 The subclasses no longer make any assumptions about the colour (complexion) of the smiley. Instead, it simply “asks” the superclass by calling the new method we defined.  While that still does not provide the flexibility we crave, it will make it easier to provide this flexibility.  We will now update the base class to allow flexible colours.  Follow these instructions:   1. Add a default argument to the magic method **\_\_init\_\_()** of the **Smiley** class and call it **complexion**. Assign **YELLOW** to it. (Yes, this is allowed.) 2. Create a new **instance variable** called **my\_complexion** and assign **complexion** to it. 3. Why do we need to call this instance variable **my\_complexion** and can’t just use **complexion**? (You may refer to Question 8.)   In question 8, we named the function to return the colour value “complexion” so to ensure we refer to the instance variable and not the method, we would need to call the instance variable something else. By defining it as an attribute of the base class, it is then applied to each subclass, wherein they can specify the type of complexion used by the class.   1. Also assign **complexion** to **Y**. (Okay, not the best name, but because the scope is so small, we’ll allow it for now.) 2. Finally, update the **method** **complexion** and have it return **self.my\_complexion**. 3. Run the program and see that it still just creates a yellow smiley… |  |
|  | *This space left intentionally blank.* |  |
| 10 | Flexible colours – Step 3 We have updated our code to allow for smileys to have different colours. The colours that are provided for us are Yellow, Green, Red, and Blue.  Let’s update the sad smiley to be Blue. (See what we did there?)  Follow these instructions:   1. Locate the initialiser method of the class **Sad** and within it the call to its super class’s initialiser method. 2. Change the call as follows:  **super().\_\_init\_\_(complexion=self.BLUE)** 3. Run code to show that the sad smiley is now blue.      1. Run code to show that the happy smiley is still yellow.      1. In your own words, **describe** how you would create a new angry smiley that has a Red complexion and angry looking eyes. (You don’t have to write this code, but you may do this as an exercise.)   At first, I would create another subclass of the Smiley class called Angry that specifies in the class attributes that complexion =self.Red and otherwise has the same methods as Happy/Sad. Although one could add additional pixels into the draw\_eyes function to give the smiley “angry” eyebrows, this would cause the eyebrows to “blink” as well. As such, I would create another function called draw\_eyebrows in the Angry class, that specifies the pixels to be used for the eyebrows only: |  |
|  | *This space left intentionally blank.* |  |
|  | Submission of Portfolio Work To submit the portfolio, do the following:   * Save the document with your answers as a MS Word file (.docx). * Save the code that you have produced or changes in a zip-file. Make sure to remove any virtual environment (.venv) and project directories (.idea) from the zip-file before uploading. * Open Blackboard, and locate the AT2 Portfolio Task 3 assessment * Open the assessment and upload the original word-processed document and the zip-file. * Click submit.   Whilst there is no need to use any other word processing software as you have access to Office 365 for free as a student, if you use Apple Pages, or Open Office, we will then require you to upload the original file **AND** a PDF version. |  |

# 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”.