|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task:** | | **3** | | |
| **Task Title:** | | **Portfolio** | | |
| **Task Code:** | | **AT2 POR-Task-3** | | |
|  | |  | | |
| 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? There are 6 classes in total including some subclasses. 2. In your own words, describe how ‘abstraction’ is visible in this project. Abstraction is visible in this project when there is the word “abc” mentioned like in blinkable.py there’s from abc import ABC, abstractmethod. And if you wonder what “abc” stands for in this case then it’s a short word for “Abstract Base Classes” which is not too apparent that it is an abstraction, another way how we may know if something is an abstraction is when they mention the term “abstractmethod” because that can also be considered as an abstraction. 3. Describe which of these classes are subclasses and which are base (or super) classes. There are a lot of super and subclasses across all the .py project file. In the blinkable.py files, the superclass is Blinkable however there is no subclass identified. In the happy.py files, Happy is considered as the subclass while Smiley and Blinkable is superclass. In the main.py file, there is no class let alone subclass. In the sad.py file, Smiley is the superclass while Sad is a subclass. In the sensehat.py files there is the sensehat class. As for the Smiley.py file there is the Smiley class 4. What is the name of the process of deriving from base classes? It is Inheritance | n/a |
|  | *This space left intentionally blank.* |  |
| 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 main common points between the two is that both of them shares the same Smiley class as their superclasses. On top of that they both have the same def \_\_init\_\_(self) and def draw\_mouth(self) as well as def draw\_eyes(self, wide\_open=True): which pretty much shares the same body/content except for the draw\_eyes mouth pixel which doesn’t have the same pixel co-ordination number, which we will cover in the next question. Not just that but when it comes to OOP Structure, both classes follow the same structure that basically describe what as in each method are supposed to do in the program, for example: the draw mouth method is responsible for the shape in which the mouth is draw. * The differences between **Happy** and **Sad** The differences between the two are that while both carry the same inheritance of Smiley as their superclass. The happy.py program also carry blinkable at its abstract base class aka ABC. The other thing is the mouth pixel co-ordination between while both are responsible for drawing mouth they both have different co-ordination. * Is there anything that stands out as a difference? Apart from those mentioned above there is one other distinctive difference that I have noticed and it is that sad.py did not import the blinkable class from blinkable.py files. Which as a result mean that it won’t blink in sad.py | n/a |
|  | *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**)? They are using the SenseHat within the Smiley class in smiley.py files 2. Which functionalities of **SenseHat** are used? In the Smiley class within smiley.py files. From def \_\_init\_\_(self): method, the Sensehat is used to create an instance of SenseHat() class, then they will assign that to the sense\_hat variable name through the = which is an assigning operator.   Afterward, in the def dim\_display(self, dimmed=True): method, the sensehat is responsible for dimming the display by taking the parameter dimmed which means if the method returns “false” then don’t dim otherwise if it is “true” then the dimmed parameter will take effect and start dimming.  As for the def show(self): method, the sensehat is used to paint smiley and displays it on to the screen for us so the way in how it works in the following code “self.sense\_hat.set\_pixels(self.pixels)” is that it takes self.sense\_hat in this case (sense\_hat) is a variable name which has been referred to earlier in the program and is assigned with the SenseHat() class through, after that they uses the set\_pixels method which allow the users to allocate the specific pixel co-ordination then the (self.pixels) is the variable name which carries all the pixel co-ordination earlier in the program and as mentioned earlier it’d allow the users to allocate all of those pixel co-ordination and get it to work.   1. What is the process of storing and potentially hiding objects in classes called? It is called encapsulation. | n/a |
|  | *This space left intentionally blank.* |  |
| 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? I think the way in which the blink() method made the smiley blink is because of the sequential order in which the program runs. And what I mean by that is as the blink() method body content run line-by-line, first it told the eye to close when they said self.draw\_eyes(wide\_open=False), the =False cause the eye to close then after that it uses self.show() to display smiley after that the time.sleep(delay) took the parameter called delay within the blink() method to delay the program for 0.25 seconds after that it uses self.draw\_eyes(wide\_open=True) which now mean since it is =True the smiley eyes open again followed by self.show() which displays it for us. 2. 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 3. Implement the code that makes the smiley blink. You may use the implementation from **Happy** as guidance. 4. Test the code on your Raspberry Pi (or with the provided classes) and watch the sad smiley blink its eyes. (You may have to adjust the **main.py** script for this.) |  |
|  | *This space left intentionally blank.* |  |
| 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. In this case the Blinkable class is considered as ABC class aka Abstract Base Class 2. 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? Abstract Base Classes aka ABC 3. What is the answer from the previous question an example of? Choose from: Abstraction, Polymorphism, Inheritance, and Encapsulation. It is an example of Abstraction 4. Why were you able to use it like the one defined in **Happy**, yet without using **Blinkable**? 5. 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 space left intentionally blank.* |  |
| 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 colours are defined within the class Smiley itself and before the def \_\_init\_\_(self): method, where they have all the colours and supposedly their RGB Colour number. 2. What is the name of the **type of variables** that hold the colours? It is tuples since it is holding multiple RGB values for a specific colour within a single variable name 3. Where are the colour variables actually **used**? In regards to structure, the colour variables were first worked from within the def \_\_init\_\_(self): method where they assigned the pixels variable from the self.pixels a co-ordinated map of colour. Then in the same smiley.py files, it is used above the self.pixels that we ran through earlier where it said “Y = self.YELLOW” 4. What would be an easy, albeit rather naive, way to change the colour of the smileys, for example, to green? The easiest way in which you can change the colour of smileys is definitely to change the Y = self.YELLOW initially to a Y = self.GREEN. However it is important not to touch the Y because that’s the colour co-ordination variable name so it won’t work without having to change every letter within the co-ordinated map to the variable name you decide to set it to which obviously will cost you a lot of time. |  |
|  | *This space left intentionally blank.* |  |
| 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**.   def complexion(self):  return self.YELLOW   1. (Technically speaking, smileys don’t have a complexion as they have no skin, but it sounds a bit nicer than just “colour”.) 2. Update the subclasses that use the colour variable directly to, instead, use the new method **complexion**. 3. 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**.) It is encapsulation 4. Check that the new code works correctly. It should still just show a yellow smiley… |  |
|  | *This space left intentionally blank.* |  |
| 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.)Technically speaking, just like in Question 8 where they said “smileys don’t have a complexion as they have no skin, but it sounds a bit nicer than just color”. As in this situation, instead of using multiple complexion within the same program which will lead to confusion, you can just name it my\_complexion in order to minimised the possibility of confusion especially if fixes/changes are meant to be implemented not just that but it will also make the whole code look cleaner. 4. Also assign **complexion** to **Y**. (Okay, not the best name, but because the scope is so small, we’ll allow it for now.) 5. Finally, update the **method** **complexion** and have it return **self.my\_complexion**. 6. 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. 4. Run code to show that the happy smiley is still yellow. 5. 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.) Code shown below, however when it comes to an angry face, you will have to find out the mouth = [49, 54, 42, 43, 44, 45] and eyes = [10, 13, 18, 21] exact co-ordination value in order to draw an angry smiley face. Because in this case I left it as it is in sad face. But when it comes to real life development in a job setting, find the specific pixel co-ordination values for each elements that make up the angry smiley face like mouth, nose, eyebrows, eyes, etc. endless of options that you can choose from based on your liking or the project requirement honestly. |  |
|  | *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”.