**School of Computing**

|  |  |  |
| --- | --- | --- |
| Module Code | M30299 |  |
| Module Title(s) | Programming |
| Module Coordinator  Other lecturers | [Matthew Poole](mailto:matthew.poole@port.ac.uk)  <[matthew.poole@port.ac.uk](mailto:matthew.poole@port.ac.uk)>  [Nadim Bakhshov](mailto:nadim.bakhshov@port.ac.uk)  <[nadim.bakhshov@port.ac.uk](mailto:nadim.bakhshov@port.ac.uk)>  [Mani Ghahremani](mailto:mani.ghahremani@port.ac.uk)  <[mani.ghahremani@port.ac.uk](mailto:mani.ghahremani@port.ac.uk)> |
| Assessment Item number | Item 6 |
| Assessment Title | Python Assignment -  Smart Home |
| Date Issued | 17 Feb 2023 |

**Schedule and Deliverables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Deliverable** | **Value** | **Format** | **Deadline Date** | **Late deadline**  **ECF deadline** |
| A zipped folder containing all Python files for the application | 25% of the module | Zip file containing your code, named using your student number (e.g. 2123456.zip) | 23:00,  14 Mar 2023  Demo sessions: 15, 16, 17 Mar 2023 | 23:00,  28 Mar 2023 |

**Notes and Advice**

* The [Extenuating Circumstances procedure](https://myport.port.ac.uk/my-course/extenuating-circumstances) is there to support you if you have had any circumstances (problems) that have been serious or significant enough to prevent you from attending, completing or submitting an assessment on time. If you complete an Extenuating Circumstances Form (ECF) for this assessment, it is important that you use the correct module code, item number and deadline (not the late deadline) given above.
* [ASDAC](http://www2.port.ac.uk/additional-support-and-disability-advice-centre/) are available to any students who disclose a disability or require additional support for their academic studies with a good set of resources on the [ASDAC Moodle site](https://moodle.port.ac.uk/course/view.php?id=3012)
* The University takes any form of academic misconduct (such as plagiarism or cheating) seriously, so please make sure your work is your own. Please ensure you adhere to our [Code of Student Behaviour](https://policies.docstore.port.ac.uk/policy-053.pdf).
* Any material included in your coursework should be [TECFAC 08 Plagiarism](https://www.youtube.com/watch?v=2a0QJnCmfEs) fully cited and referenced in **APA 7** format. Detailed advice on referencing is available from the [library](https://library.port.ac.uk/w165.html).
* Any material submitted that does not meet format or submission guidelines, or falls outside the submission deadline could be subject to a cap on your overall result or disqualification entirely.
* Instead of referring to external sites to look for help, join our [Module Discord channel](https://docs.google.com/document/d/1G-3lPyALv771OFLrkdIWGMbzNltJoKHcICW4Os2pmfE/edit?usp=sharing) to post your questions.
* If you need additional assistance, ask your personal tutor or the student engagement officer at [ana.baker@port.ac.uk](mailto:ana.baker@port.ac.uk). The faculty academic tutors are also available for you to book a one-to-one session with them using [the Moodle page for faculty academic tutors](https://moodle.port.ac.uk/course/view.php?id=9173).
* If you are concerned about your mental well-being, please contact our [Well-being service](https://myport.port.ac.uk/guidance-and-support/health-and-wellbeing).

**M302399**

**Item 6, Python Coursework:**

***Smart Home***

# Scenario

A fictional UK-Finnish company, Alikoti, develops eco-friendly smart home technologies. Since 1999 they have developed fully integrated Smart Home Solutions. Since 2011 they have begun selling standalone smart home devices, including smart lights and smart fridges, to be supported by third-party software. Last year the company made the decision to build its own Smart Home App. They have asked you to develop a Tkinter GUI prototype in Python. The prototype must provide a graphical user interface that interacts with a smart home object. The smart home object is responsible for managing data for all smart devices. The company will use this prototype to make a business case for a fully developed application.

# 

# Tasks

The table below outlines the tasks that you need to complete. [Task 1](#_uhf7bcfoc7pb) and [task 2](#_i9fgkayijo1g) describe what you need to do for the back-end (written in a file called backend.py). [Task 3](#_3voi63oqp1yk), [task 4](#_fm13apo2vyi8) and [task 5](#_evw9b78pleid) describe the requirements for the front-end code (to be written in a file called frontend.py).

It is suggested that you complete the tasks in this order. Note that it is not compulsory to complete [the challenge features](#_8qy59j853m9q). Only attempt these if you feel confident and have the time.

Once you are ready to submit, zip (compress) the folder containing both files (backend.py and frontend.py) and then submit the resulting zip file. In **Windows** File Explorer you can do this by putting your .py files in a folder, then right-clicking on the folder and selecting send to > compressed (zipped) folder. On **macOS**, open Finder and put your .py files in a folder then right-click and select “compress”. Please make sure you submit a .zip file (not .rar, .7z or .tar …).

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Title** | **Details** | **Marks** |
| 1 | Design, implement and test a  **SmartPlug** class | In backend.py, create a class SmartPlug. This class represents a smart plug object with the properties and behaviour described below:  **Instance variables**   * switchedOn: Represents whether the plug is turned on. switchedOn can be either on (True) or off (False). * consumptionRate: An integer that gives the consumption rate of the plug. The consumptionRate can be any value between 0 and 150.   **Initialisation method (constructor)**  The class should have an \_\_init\_\_ method that sets switchedOn to False and consumptionRate to 0.  **Methods**   * toggleSwitch that toggles the value of switchedOn. * An accessor method (a getter) for switchedOn. * Accessor (getter) and mutator (setter) methods for consumptionRate. * A \_\_str\_\_ method that returns a string representation of the smart plug. (Printing a smart plug should show whether it is on and the value for consumptionRate).   **Testing**  Test your SmartPlug using a testSmartPlug function. The testSmartPlug function should:   1. Create an instance of the SmartPlug class. 2. Toggle this plug by calling the toggleSwitch method. 3. Print the value of switchedOn using the appropriate accessor method. 4. Print the value of consumptionRate, set it to a new value (of your choice), and then print it again. 5. Print the SmartPlug object. | 15 |
| 2 | Design, implement and test a custom smart device class based on your student number  (see [table 2](#_lg7te4s09o5e)) | Start by looking at [table 2](#_lg7te4s09o5e) to find out which device you need to create. This device must have the properties and behaviours listed below. This class needs to be added to backend.py.  **Instance variables**   * switchedOn: Represents the switch state of the device, either on (True) or off (False). * The option of your device (listed in [table 2](#_lg7te4s09o5e)).   **Initialisation method (constructor)**  The class should have an \_\_init\_\_ method that sets switchedOn to False and the option to its default value (listed in [table 2](#_lg7te4s09o5e)).  **Methods**   * toggleSwitch that toggles the value of switchedOn. * An accessor method for switchedOn. * Accessor and mutator methods for the option (from [table 2](#_lg7te4s09o5e)). * A \_\_str\_\_ method that returns a string representation of the device. (Printing a device should show whether it is on and the value for the option instance variable.)   **Testing**  Test your device in a testDevice function which should:   1. Create an instance of your device class. 2. Toggle this device by calling the toggleSwitch method. 3. Print the switchedOn instance variable using the appropriate accessor method. 4. Print the current value of the option (from [table 2](#_lg7te4s09o5e)). Then set it to a new value (of your choice). Next, print it again. 5. Print the custom device object. | 15 |
| 3 | Design, implement and test a **SmartHome** class | In backend.py, add a SmartHome class that manages and controls a collection of devices.  **Instance variable**   * devices: A list of devices. Each element in the list can be of any kind, i.e. a SmartPlug or a custom device (from [table 2](#_lg7te4s09o5e)).   **Initialisation method (constructor)**  The class should have an \_\_init\_\_ method that initialises devices to an empty list.  **Methods**   * getDevices that returns the list of devices. * getDeviceAt that takes an index as a parameter and returns the device/plug at the given index in the list of devices. * addDevice - this method should take an object (either a custom device or a SmartPlug) and add it to the list of devices. * toggleSwitch - this method should take the index of an item in the list of devices and switch the corresponding device on. * turnOnAll - this should turn on all the devices in the devices list. * turnOffAll - this should turn off all the devices in the devices list. * A \_\_str\_\_ method that returns a well-formatted string representation of all the devices in the devices list.   **Testing**  Test your class in a testSmartHome function which should:   1. Create an instance of the SmartHome class. 2. Create two instances of the SmartPlug class and store them in variables plug1 and plug2. 3. Create an instance of your custom device. 4. Turn on plug2 by calling its toggleSwitch method. 5. Set the value of consumptionRate in plug2 to 45. 6. Set the option of the custom device to a value of your choice. 7. Add plug1, plug2 and your custom device to the SmartHome object. 8. Print the SmartHome object. 9. Turn on all devices in the SmartHome. 10. Print the SmartHome object again. | 15 |
| 4 | Create an instance of SmartHome to be used by the GUI  (see [task 5](#_evw9b78pleid)) | Add code to frontend.py for the following:   1. Import the three classes that you have created in backend.py. 2. Create an instance of SmartHome class as a global variable. 3. A function called setupHome which creates and adds five devices to the SmartHome object. (The devices you should add are listed in [table 1](#_uhf7bcfoc7pb) and are determined by the penultimate digit of your student number). 4. A main function (or equivalent) that calls setupHome to set up the smart home. 5. A call to the main function. | 5 |
| 5 | Design, implement and test the GUI **display** for the smart home system | Next, in frontend.py, create an instance of the Tk class (from the Tkinter library) and store it as a global variable called mainWin.  mainWin will act as a GUI for the SmartHome object defined in [task 4](#_fm13apo2vyi8). Here are the requirements for your application:   1. It should display all devices and their properties (added to the SmartHome object in [task 4](#_fm13apo2vyi8)). 2. For each device there needs to be a text widget and a ‘Toggle this’ button. The text widget should display the string representation of the device. 3. There should be a single ‘Turn all on” button. 4. There should be a single ‘Turn all off” button.   Here is what your GUI needs to look like. The example below is generated given a student number ending with 10 (for example “up2123410”). | 15 |
| 6 | GUI: Button Interactivity | Once you have successfully built your GUI you now need to design, implement and test the button functionality.  For each of the buttons on the GUI ([task 5](#_evw9b78pleid)), you need to design the code that will implement the functionality of the button; i.e. you need to implement functionality for the:   1. Button to turn on all devices 2. Button to turn off all devices 3. Button for each row (i.e. the toggle button for each device). | 15 |

## 

# Challenge features

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Title** | **Details** | **Marks** |
| 7 | Challenge Features | You are free to add as many of the following features/enhancements to your system. The following tasks are of varied difficulty. The marker (during the demo) will judge how many additional marks your challenge features deserve. As a guide, the features labelled with \*\* will gain more marks than those labelled with \*.   * **\*** Display the number of devices that are turned on in the home. This should be displayed using a label widget at the bottom of the GUI created in [task 4](#_fm13apo2vyi8). This value should update automatically as devices are toggled. * **\*** Add documentation using docstrings for every function, method, and class. * **\*\*** Use inheritance in the back-end representations of SmartPlug and your custom smart device. This should reduce repetition and the size of your back-end code. * **\*\*** Add a button to the GUI next to each device that opens a new window to modify each device (i.e. set its options). Notice that the options for your custom smart device differ from the smart plug. * **\*\*** Provide buttons to add or delete devices from the home. When adding a device, the user will need to choose whether they want a smart plug or a custom smart device. The device should then be created with default values and added at the bottom of the list of devices. | 20 |

# Table 1 (penultimate digit)

The **penultimate digit** of your student number determines the order you need to add devices or smart plugs to your smart home object from [task 4](#_fm13apo2vyi8). Make sure you add them in the specified order.

SP: a SmartPlug

C: custom smart device (based on [table 2](#_lg7te4s09o5e))

|  |  |
| --- | --- |
| **Penultimate digit** | **Smart Home Structure** |
| **0** | SP, C, C, SP, C |
| **1** | C, SP, C, SP, C |
| **2** | C, C, SP, SP, C |
| **3** | SP, SP, C, C, C |
| **4** | C, SP, SP, C, C |
| **5** | SP, C, SP, C, C |
| **6** | C, C, C, SP, SP |
| **7** | SP, SP, SP, C, C |
| **8** | C, SP, C, C, SP |
| **9** | SP, C, SP, SP, C |

# Table 2 (final digit)

Using the **final digit** of your student number you are required to build a custom smart device. The table below shows what the name of the class should be and the option that it needs to have.

|  |  |  |  |
| --- | --- | --- | --- |
| **Final Digit** | **Custom Smart Device** | **option** | **default value** |
| **0** | SmartLight | brightness  (percentage 0-100) | 0 |
| **1** | SmartFridge | temperature  (1–4 degree Celsius) | 2 |
| **2** | SmartHeater | setting  (0–5) | 0 |
| **3** | SmartTV | channel  (1-734) | 1 |
| **4** | SmartSpeaker | streaming  ("Amazon", "Apple" or "Spotify") | "Amazon" |
| **5** | SmartDoorBell | sleepMode  (silenced bell: True or False) | False |
| **6** | SmartOven | temperature  (0–260 degree Celsius) | 0 |
| **7** | SmartWashingMachine | washMode  ("Daily wash", "Quick wash" or "Eco") | "Daily wash" |
| **8** | SmartDoor | locked  (True or False) | True |
| **9** | SmartAirFryer | cookingMode  ("Healthy", "Defrost" or "Crispy") | "Healthy" |

# Demonstration & Mark Allocation

You need to **demonstrate your program** to a member of staff in your Programming **practical session timetabled in the period 15th—17th March**. We will execute your submitted program, and we will ask you question(s) about how you wrote it and how it works.

All the marks for the assignment will be awarded during the demonstration, so you must attend: **failure to attend the demonstration will result in zero marks**, and demonstrating your program late may result in your mark for the assignment being capped to 40%. **Late demonstrations must be done by the 31st of March** or you will receive a mark of zero. You need to ask [Nadim Bakhshov](mailto:nadim.bakhshov@port.ac.uk), [Matthew Poole](mailto:matthew.poole@port.ac.uk) or [Mani Ghahremani](mailto:mani.ghahremani@port.ac.uk) at the beginning of a drop-in (Mon 11am) or a practical class if you wish to do a late demo.

Formal written feedback and your assignment mark will be sent to you via email immediately after your demonstration has been completed. If you do not receive this email, then your mark may not have been recorded and it is your responsibility to inform Nadim if this happens.

## Marking

Each **task** will individually be marked for functionality and code quality. The total marks for each task is given in the right hand column of that task.

* Tasks 1, 2 and 3 functionality will be verified through the execution of the test function.
* Task 4 will rely on code inspection.
* Tasks 5 and 6 functionality will be verified through the execution of the GUI.

## Explanation of program features [2 marks]

We will award up to 2 marks for your responses to the question(s) we ask you in the demo, independently of the other marks your submission receives.

# General advice

Most importantly, **start early and do not leave finishing the work until just before the deadline**. Your work will almost always suffer if you leave it until too late. Furthermore, technical problems are likely to be overcome if encountered early, and do not usually constitute an acceptable reason for lateness.

If you find the task very difficult, remember that you do not have to provide a complete solution to achieve a pass mark. Make sure that your program executes and gives some graphical output, and that you demonstrate what your program does.

# Support

The last few practicals before the deadline can be used for help with the coursework. [Alex Bennett, Simon Jones and Eleni Noussi](https://docs.google.com/document/d/1Z9YZWjyjYNS5tex9mfeZLiKv-En64q1PcfpFTonHEdE) are also able to give advice on the coursework via one-to-one support sessions. Alternatively, join our [Discord](https://docs.google.com/document/d/1G-3lPyALv771OFLrkdIWGMbzNltJoKHcICW4Os2pmfE/edit) channel and post your questions. Any reported errors on this handout, or other common issues, will be communicated via the Discord channel and the [Smart home Coursework Frequently Asked Questions document](https://docs.google.com/document/d/1d_s8hHToJcBYHVVFLNyIvPR-Mzi-5TeDXSY7qYSCvrc/edit?usp=sharing) so please check there before asking a question.

Other queries should be addressed to me by email ([nadim.bakhshov@port.ac.uk](mailto:nadim.bakhshov@port.ac.uk)) or to Mani ([mani.ghahremani@port.ac.uk](mailto:mani.ghahremani@port.ac.uk)).