COMP 1039

Problem Solving and Programming

Programming Assignment 2 – Part 2

Vending Machine

Prepared by
Andreas Jordan

South Australian Institute of Business and Technology
September 2018

Contents

INTRODUCTION

This document describes the second assignment for Problem Solving and Programming.

The assignment is intended to provide you with the opportunity to put into practice what you have learnt in the course by applying your knowledge and skills to the implementation of a **Python module** (that contains functions that operate on lists) and **a program that will simulate a vending machine.**

This assignment is an **individual task** that will require an **individual submission**. **You will be required to present your work to your practical supervisor during your practical session held in Week 12 of the study period**. Important: You must attend the practical session that you have been attending all study period in order to have your assignment marked.

This document is a kind of specification of the required end product that will be generated by implementing the assignment. Like many specifications, it is written in English and hence will contain some imperfectly specified parts. Please make sure you seek clarification if you are not clear on any aspect of this assignment.

ASSIGNMENT OVERVIEW

There are two parts to this assignment:

Part I: Writing a set of Useful Functions for our Vending Machine (list manipulation functions)

You are required to implement a Python module that contains functions that manipulate lists. Please ensure that you read sections titled 'Part I Specification' below for further details.

Part II: Simulate a Vending Machine
You are required to write a Python program that will simulate a vending machine. The program will display a number of products and their prices (using a `list` of inventory_item *objects*). Product information (i.e., name, price and quantity) will be stored in a text file that will be read in when the program commences. Once the initial product information has been read in from the file, the program should allow the user to interactively view, select and pay for a product. Please ensure that you read sections titled 'Part II specification' below for further details.

Please ensure that you read sections titled 'Part I Specification' and 'Part II Specification' below for further details.

PART II SPECIFICATION – VENDING MACHINE SIMULATOR

Write a **menu driven program** called `yourSaibtId_vending_machine.py` that will allow the user to enter commands and process these commands until the quit command is entered. The program will simulate the operation of a vending machine. Products in the vending machine will be represented as `inventory_item` objects (and stored in a list). Product information will be stored in a text file that will be read in when the program commences. Once the initial inventory data has been read in from the file, the program should allow the user to interactively purchase products, insert money and view product information.

Input
When your program begins, it will read in personal profile information from a file called `inventory.txt`. This is a text file that stores product information for the vending machine. An example input file called `inventory.txt` is provided on the course website (under the Assessment tab). You may assume that all data is in the correct format. Each line of text represents information for a single product (separated by a comma).

For example:

```
Mars Bar,2.40,7
```

- 1^{st} value – Product **name**
- 2^{nd} value – Product **price**
- 3^{rd} value – Product **quantity**

After the program has stored the data (using a List of `InventoryItem` objects), it will enter interactive mode as described in the following section.

Interactive Mode

Your program should enter an interactive mode after the product information has been read from the file. The program will allow the user to enter commands and process these commands until the quit command is entered. The following commands should be allowed:

1. **Display Products**
   Outputs the contents of the inventory list as seen below in the section titled *Screen Format*.

2. **Purchase a Product:**
   Prompts for an item number matching product displayed. If user enters a valid choice, prompt user to enter coins. Your program must continue to prompt for additional coins until the user has entered sufficient funds to purchase the product. When successful, your program must display a message to say they successfully purchased the product and the amount of change owed.

3. **Quit:**
   Causes the program to quit and output the contents of the list of inventory items to a file called `new_inventory.txt`. The format of this file should exactly match that of the input file.

⚠ The program should display an appropriate message to indicate whether a command has been successfully completed.

⚠ Please refer to the sample output (available as a separate document) to ensure that your program is behaving correctly and that you have the correct output messages.

PRACTICAL REQUIREMENTS (PART II)

It is recommended that you develop this part of the assignment in the suggested stages. Each stage is worth a portion of the marks.

It is expected that your solution will include the use of:

- Your solution in a file called `yourSaibtId_vending_machine.py`.
- The supplied `inventory_item.py` module (that defines the InventoryItem class). This is provided for you – **do NOT modify this file**.
- Appropriate and well constructed **`while`** and/or **`for`** loops. (Marks will be lost if you use **`break`** statements in order to exit from loops).
- Appropriate **`if`**, **`if-else`**, **`if-elif-else`** statements (as necessary).
- The following functions:
  - **`main()`**
    - A function that will contain your main **program** loop
    - Accepts no parameters
    - The function returns nothing
    - You **must use** a loop in your solution (obviously!)
  - **`display_main_menu()`**
    - A function that will contain your main **menu** loop
    - Accepts no parameters
    - Returns the choice made by user
    - You **must use** a loop in your solution (obviously!)
  - **`read_file(filename, inventory_list)`**
    - Takes two arguments as parameters:
      1. The name of the input file; and
      2. A **`list`** of inventory *objects* as parameters
    - Reads the contents of the file into list **`inventory_list`**
    - You **must use** a loop in your solution.
    - You **must use** list function **`insert_item`** defined in Part I to add items to the **`inventory_list`**
    - You may find the String methods **`split()`** and **`strip()`** useful here.
    - You **may** use String methods in this function only.
  - **`write_to_file(filename, inventory_list)`**
    - Accepts the name of the output file (filename) and list of inventory_item objects (inventory_list) as parameters
    - Output the contents of the inventory list (list of inventory_item objects) to a file **in the same format as the input file**
    - The file will need to be opened for writing in this function (and of course closed once all writing has been done).
    - The function returns nothing
    - You **must use** a loop in your solution.
  - **`display_vending_machine(inventory_list)`**
    - Accepts a list of inventory *objects* as a parameter
    - Displays the contents of the list to the screen in the format specified (see *Screen Format* on page **??**)

    - The function returns nothing
    - You **must use** a loop in your solution

  - **`accept_money(product_price)`**
    - Allows the user to purchase a product from the vending machine
    - Accepts a product's price as a parameter
    - The function returns the change from the purchase
    - You **must use** a loop in your solution

  - **`purchase_product(inventory_list)`**
    - Accepts a list of inventory *objects* as a parameter
    - Allows the user to purchase a product from the vending machine
    - The function returns nothing
    - You **must use** a loop in your solution

Your solutions **MAY** make use of the following:

- Built-in functions **`int()`**, **`input()`**, **`print()`**, **`range()`**, **`open()`**, `close()`,**`len()`** and **`str()`**.
- Concatenation (+) operator to create/build new strings.
- Access the individual elements in a string with an index (one element only). i.e. `string_name[index]`.
- Access the individual elements in a list with an index (one element only). i.e. `list_name[index]`.
- `inventory` objects and methods (as appropriate).
- The `list_function.py` module (that you wrote in part I of this assignment). You may like to make use of some of the functions defined in the `list_function.py` module for this part of the assignment (as appropriate). Not all will be suitable or appropriate.

Your solutions **MUST NOT** use:

- Built-in functions (other than the **`int()`**, **`input()`**, **`print()`**, **`range()`**, **`open()`**, `close()`, **`len()`** and **`str()`** functions).
- Slice expressions to select a range of elements from a string or list. i.e. `name[start:end]`.
- String or list methods (i.e., other than those mentioned in the 'MAY make use' of section above).
- Global variables as described in week 8 lecture.
- The use of **`break`**, **`return`** or **`continue`** statements (or any other technique to break out of loops) in your solution – doing so will result in a significant mark deduction.

PLEASE NOTE: You are reminded that you should ensure that all input and output conform to the specifications. If you are not sure about these details you should check with the sample output provided at the end of this document or post a message to the discussion forum in order to seek clarification.

Please ensure that you use Python 3.6.0 or a later version (i.e. the latest version) in order to complete your assignments. Your programs **MUST** run using Python 3.6.0 (or latest version).

STAGES (PART II)

It is recommended that you develop this part of the assignment in the suggested stages. Many problems in later stages are due to errors in early stages. **Make sure you have finished and thoroughly tested each stage before continuing.**

The following stages of development are recommended:

STAGE 1 – SET UP

First, let's get set up! Follow the steps below:

1. To begin, download the following provided files (available on the course website):
   - **`vending_machine.py`**
   - **`inventory.txt`**
   - **`inventory_item.py`** – See page xxx for a description of available functions in this file
2. Rename **`vending_machine.py`** to **`yourSaibtId_vending_machine.py`**
3. Make a copy of **your** Assignment 2 Part I (list functions) and place it in the **same directory** as **`yourSaibtId_vending_machine.py`** (you will need to import this file in Stage 2)

STAGE 2 – IMPORTING MODULES `inventory_item` and `list_function`

In the stages that follow, you must use at least two list functions you wrote in the first part of the assignment:

- **`size()`**
- **`insert_item()`**

If your SAIBT Id starts with a number, you will need to add an alphabetical letter to the front (Python does not like numbers in front of files when importing).

If your solution to Part I does not run (i.e. compile) or your respective functions do not work as per the specification, you may use standard list functions **`len()`** and **`append()`**/**`insert()`** but will receive a **grade reduction**.

Additionally, this assignment introduces **objects**. You will be required to create, initialise and set/get an objects data attributes. However in order to do this we need to import the file containing the class definition:

```
inventory_item.py
```

So let's start by importing the two files to make their programming code available in your file **`yourSaibtId_vending_machine.py`**...

Step 1: At the top of your file (but **underneath** the comment header), start by importing the **`InventoryItem`** class. To do this add the following line of code to your program:

```
importinventory_item
```

Step 2: Now import the code you wrote for Part I. To do this add the following line of code to your program:

```
importlist_function
```

Run your program and fix any errors that may have resulted. If you have done this correctly, you should now be able to access the functions you created in Part I of this assignment. Furthermore, you can now create inventory item **objects**. However let's not get ahead of ourselves. For now, move on to the next stage. . . ...

STAGE 3 – IMPLEMENT FUNCTION `main()`

Now that we have covered functions, you will soon discover that writing self-contained units of code is a lot easier to manage. You will find that the more you practice programming, most of your code will appear inside your own user-defined functions. However there is always some code that does not appear to belong in any specific "function" per se. This may be the main loop or your calls to the various functions making up your program. This is where creating a **`main()`** function comes in handy!

Recall that functions can accept arguments. The idea is to pass whatever data a function needs as a parameter. This mitigates the use of global variables which can be difficult to keep track of (in terms of the values they hold throughout the execution of your program).

☞ The use of a **`main()`** function prevents the unnecessary creation of global variables.

☞ The variables you previously declared in your programs **outside of a function** (which up until Week 7 would be most of them) are in effect global variables. So with that said, let's move on. . .

Step 1: Let's start by reviewing the specification for outlined in section PRACTICAL REQUIREMENTS (PART II):

`main()`

- A function that will contain your main loop
- Accepts no parameters
- The function returns nothing
- You **must use** a loop in your solution

Step 2: Create a function called **`main()`**

– Place the function at the bottom of your program (it will be easier to find as your progress further into the assignment).

– For now, write a print statement to display the string **`'In main menu'`** as shown below:

Sample Output 1

```
def main():
    print('In function main.')
```

Step 3: Since **`main()`** is a also function, it too must be **called** in order to run/execute your program so now it's time to insert the code that will call your **`main()`** function as shown below in Sample Output 2.

Sample Output 2

```
def main():
    print('In function main.')

main()
```

⚠ If you are wondering why nothing is happening when you *run* your program. . .
⟹ Check that you are calling **`main()`**

☞ Test to ensure that this is working correctly before moving onto the next stage. You do not need to call any functions at this point, you may simply display an appropriate message to the screen as shown above.

STAGE 4 – IMPLEMENT FUNCTION `display_main_menu()`

Key Python Terms: `if-elif-else`, `print`, variables, comparison operators

Now it's time to implement the function that will handle the main interactive menu, i.e. to prompt for and read menu commands.

Step 1: Now is a good time to review the specification for this function:

> **`display_main_menu()`**
>
> – A function that will contain your main menu loop
> – Accepts no parameters
> – Returns the choice made by user
> – You **must use** a loop in your solution

☞ **Remember** that in this stage we are only **implementing the menu** and **not** the functions that options 1 & 2 in the menu will (eventually) call.

☞ The following code **should be** part of the **`main`** function.

Step 2: Write code to display the menu as shown in Sample Output 3 below:

Sample Output 3

```
1. Purchase item
2. Quit
```

Step 3: Add a prompt that asks the user to enter a number as shown in Sample Output 4 below:

Sample Output 4

```
1. Purchase item
2. Quit

Please enter 1-2 to select:
```

Step 4: Construct a loop that will continuously display (and prompt) the user until they enter option 3 from the menu (i.e. the user wants to quit the program). **Hint:** This is an ideal stopping condition for a loop!

- An example is shown below in Sample Output 5:

Sample Output 5

```
1. Purchase item
2. Quit

Please enter 1-3 to select: 1

Inside function purchase_item

1. Purchase item
2. Quit

Please enter 1-2 to select:
```

☞ You do not need to call any function(s) at this point, you may (and probably should) just **display an appropriate message** to the screen as per Sample Output 5 above.

Step 5: Now it's time to write code that will **validate** any **menu input** entered by the user. An appropriate message should be displayed if incorrect input is entered by the user.

- An example is provided in Sample Output 6 below:

Sample Output 6

```
1. Purchase item
2. Quit

Please enter 1-2 to select: 7
Please enter either 1-2.

Please enter 1-2 to select:
```

☞ Test to ensure that your menu is working correctly **before** moving onto the next stage.

STAGE 5 – IMPLEMENT FUNCTION `read_file()`

Key Python Terms: `while`, `open`, `close` `float`, `int`, `read/readline/readlines`, variables, comparison operators

Write the code for function `read_file()`.

Step 1: Now is a good time to review the specification for the `read_file` function:

> **`read_file(filename, inventory_list)`**
>
> – Takes two arguments as parameters:
>
>   1. The name of the input file; and
>   2. A `list` of inventory *objects* as parameters
>
> – Reads the contents of the file into list `inventory_list`
>
> – You **must use** list function `insert` defined in Part I to add items to the `inventory_list`
>
> – You **must use** a loop in your solution.
>
> – You **may** use String methods in this function only.
>
>   * You may find the String methods `split()` and `strip()` useful here.

Step 2: Now is also a good time to take a closer look at the contents of the file you must use. Sample Output 7 shows the contents of the file:

Sample Output 7 – Contents of `inventory.txt`

```
Pepsi,2.50,2
Coca-Cola,2.55,3
Mt. Ogabogee Spring Water,2.25,5
Corn Chips,3.00,2
Fruit Juice,4.00,1
Smarties,3.50,6
Mars-Bar,2.40,7
Snickers-Bar,2.50,8
M & M's,3.80,6
BBQ Chips,1.80,6
```

As can be seen in Sample Output 8, the contents of the text file represent an inventory of products. The key here is to understand the data and look for patterns in the text that you can use to make your job easier.

☞ Let's look at a single line from the file: `Smarties,3.50,6`

From this we can tell that each line contains three **comma-separated** elements:

- The first element, i.e. `Smarties`, is the **name** of the product
- The second element, i.e. `3.50`, is the **price** of the product
- The third element, i.e. `6`, is the **quantity** of the product

☞ **Hint:** You may find the String methods `split()` and `strip()` useful here.

⚠ **Confused?** – Review Week 9's lecture notes on File Input and Output Processing. After you've read the data in from the text file, you will need to break apart each line to store each relevant piece of information (about a product).

Step 3: Ensure the file `inventory.txt` is in the **same directory** as your python file (i.e. `yourSaibtId_vending_machine.`

Step 4: Write the code necessary to open the file.

- The name of the file is one of the parameters passed to this function, i.e. `filename`.

Step 5: Set up a loop to process each line of text. For each line of text, you will need to:

- Extract each (comma-separated) element to allow you to access the different elements as values in python.
- Create an inventory item **object** and initialise its data attributes with the name, price and quantity (obtained from the current line).
- Add the inventory item **object** to the list `inventory_list`

⚠ Remember that you are **not permitted** to use the built-in list function `append()`. Instead you **must use** list function `insert` defined in Part I to add items to the `inventory_list`.

⚠ If you have not implemented the list function `insert` in Part I, you may use `append()` but will receive a **mark reduction**.

By the time your loop has finished processing all of the text from the file, you should have created a number of inventory item **objects**. For each inventory item object created, you should have:

- Set its data attributes with the respective values from each line, i.e. name, price, quantity.
- Each inventory item object should have been added to the list `inventory_list`.

Step 6: Now we need to test that your `read_file()` function is behaving correctly (i.e. actually doing what it is supposed to do). **Remember** though that it needs a couple of arguments:

- `filename` – A string representing the name of the file.
- `inventory_list` – A list of inventory **objects**
- After reading in the file, write a loop that iterates through each element in the list `inventory_list`

☞ You may want to comment out the line where you call `main()` and simply add a line that directly calls your `read_file()` function.

⚠ Do not forget to `close()` the file once you have finished reading it!

STAGE 6 – IMPLEMENT FUNCTION `write_to_file()`

Now that you know the information is being correctly stored in your inventory list, write the code for function **`write_to_file()`**. This involves having to loop through the items in your inventory list and, for each inventory item, access its data attributes (i.e. name, price and quantity) and output these values to a text file. Similar to the **`inventory.txt`** file, each item must be separated by a comma in the output. Sample Output 8 below is an example of the expected format:

Sample Output 8

```
Pepsi,2.50,2
Coca-Cola,2.55,3
Mt. Ogabogee Spring Water,2.25,1
Corn Chips,3.00,2
Fruit Juice,4.00,1
Smarties,3.50,5
Mars-Bar,2.40,7
Snickers-Bar,2.50,4
M & M's,3.80,6
BBQ Chips,1.80,6
```

Step 1: As usual, let's review the specification for **`write_to_file()`** below:

`write_to_file(filename, profile_list)`

- Accepts the name of the output file (filename) and list of inventory item objects (**`inventory_list`**) as parameters.
- Output the contents of the **`inventory_list`** (list of inventory item objects) to a file in the **same format** as the **input file**.
- The file will need to be opened for writing **in this function** (and of course closed once all writing has been completed).
- The function returns nothing.
- You **must use** a loop in your solution.

Step 2: Start by writing a loop to iterate through the list of products contained in the list **`inventory_list`**.

Step 3: Inside the loop, use `print` statements to display to the screen each product's:

- name
- quantity
- price

Step 4: When you have that working, you can then add the code which **`opens`** and **`closes`** the file.

☞ **Placement** of these statements is **important**. You **do not** want to be opening a file for each iteration of the loop! Same goes with closing a file.

Step 5: Now replace your `print` statements with **`write`** commands

⚠ **Confused?** Review Week 9 lectures which covers File Input and Output Processing.

Step 6: Add code to call this function to ensure it is working correctly. A call to this function should result in two things:

1. A file getting created in the same directory as your program called `filename`.

   For example, if the value held in parameter `filename` is:

   `'new_inventory.txt'`

   . . . then the new text file will be called `new_inventory.txt`

2. The new text file will contain the name, price and quantity of each item contained in the list `inventory_list`

   For example, if an inventory item object has data attribute values as follows:

   - `'Corn Chips'`
   - `3.00`
   - `2`

   . . . then the new text file will contain the following line of text: `Corn Chips,3.00,2`

STAGE 7 – IMPLEMENT FUNCTION `display_vending_machine()`

In this stage you should make extensive use of Python's `format` command. For a more detailed explanation, see Section USEFUL BUILT–IN PYTHON FUNCTIONS – REQUIRED FOR PART II on page 24.

Step 1: Let's review what the specification states about the function `display_vending_machine()`:

> **`display_vending_machine(inventory_list)`**
>
> - Accepts a list of inventory *objects* as a parameter
> - Displays the contents of the list to the screen in the format specified in Sample Output 10 below.
> - The function returns nothing
> - You **must use** a loop in your solution

Step 2: Let's take a look at an example of what the function `display_vending_machine()` actually displays:

Sample Output 9

```
 ___________________________________________________
|                                                   |
|         *** Vending Machine Simulator ***          |
|___________________________________________________|
|                                                   |
| 1. Pepsi                                  $2.50   |
|                                                   |
| 2. Coca-Cola                              $2.55   |
|                                                   |
| 3. Mt. Ogabogee Spring Water              $2.25   |
|                                                   |
| 4. Corn Chips                             $3.00   |
|                                                   |
| 5. Fruit Juice                            $4.00   |
|                                                   |
| 6. Smarties                               $3.50   |
|                                                   |
| 7. Mars-Bar                               $2.40   |
|                                                   |
| 8. Snickers-Bar                           $2.50   |
|                                                   |
| 9. M & M's                                $3.80   |
|                                                   |
| 10. BBQ Chips                             $1.80   |
|                                                   |
|___________________________________________________|
|                                                   |
|      \____________________________________/       |
|                                                   |
|___________________________________________________|
```

Step 3: Define the function `display_vending_machine()`. The function will need:

- A loop to iterate over the items in the list
- Several `print` statements
- Several `format` statements

If you haven't done so already, now would be a good time to review Section USEFUL BUILT–IN PYTHON FUNCTIONS – REQUIRED FOR PART II on page 24.

A good idea would be to open a new file and experiment using the `format` command with some strings to see what happens. That is familiarise yourself with the command by doing and not just reading about it. This way you won't risk breaking something in your actual program!

STAGE 8 – IMPLEMENTING FUNCTION `accept_money()`

In this stage, you will write the necessary code that will allow a user to (simulate) entering coins into your vending machine.

Step 1: Let's review what the specification states about the function `accept_money()`:

`accept_money(product_price)`

- Allows the user to purchase a product from the vending machine
- Accepts a product's price as a parameter
- The function returns the change from the purchase
- You **must use** a loop in your solution

So basically, you will need to:

- Prompt the user to enter coins using the following denominations:
  - Ten cents
  - Twenty cents
  - Fifty cents
  - One dollar
- Validate the user input
  - Display an appropriate message if incorrect input is entered by the user.
- Use a loop!

Step 2: Write code to **define** the function keeping in mind that it accepts a single parameter.

Step 3: Now write code to display the following message:

```
Please continue to enter coins until amount reached.
```

Step 4: Now write the code to display the valid choices and an option to cancel (should the user change their mind about the purchase). Follow the format shown in Sample Output 10 below:

Sample Output 10

```
Please continue to enter coins until amount reached.

Valid choices for coins are:
[1] Ten cents
[2] Twenty cents
[3] Fifty cents
[4] One dollar
[5] Cancel
```

Step 5: Now write the code to **prompt** the user to enter coins. Use the following message in your prompt:

```
Enter coin [1-4] or 5 to cancel:
```

☞ A more complete example is given in Sample Output 11 that shows how your various `print` and `input` statements show flow. Note: that the main menu is **not** a part of this function. It is included here to provide continuity between displaying the menu (by calling `display_main_menu`) and output handled by this function.

Sample Output 11

```
 ___________________________________________________
|                                                   |
|        *** Vending Machine Simulator ***          |
|___________________________________________________|
|                                                   |
| 1. Pepsi                                 $2.50    |
|                                                   |
| 2. Coca-Cola                             $2.55    |
|                                                   |
| 3. Mt. Ogabogee Spring Water             $2.25    |
|                                                   |
| 4. Corn Chips                            $3.00    |
|                                                   |
| 5. Fruit Juice                           $4.00    |
|                                                   |
| 6. Smarties                              $3.50    |
|                                                   |
| 7. Mars-Bar                              $2.40    |
|                                                   |
| 8. Snickers-Bar                          $2.50    |
|                                                   |
| 9. M & M's                               $3.80    |
|                                                   |
| 10. BBQ Chips                            $1.80    |
|                                                   |
|___________________________________________________|
|                                                   |
|      \____________________________________/       |
|                                                   |
|___________________________________________________|

1. Purchase item
2. Quit

Please enter 1-2 to select: 1

Enter selection (1-10): 1

You selected Pepsi. Is this correct (y/n): j
Invalid choice. Please choose 'y' or 'n'

You selected Pepsi. Is this correct (y/n): y

Please continue to enter coins until amount reached

Valid choices for coins are:
[1] Ten cents
[2] Twenty cents
[3] Fifty cents
[4] One dollar
[5] Cancel

Enter coin [1-4] or 5 to cancel:
```

Step 6: Now modify your function by adding a loop to allow the user to continue to enter coins until **either**:

1. The user enters 5, in which case you should:
   - Return the money already entered (as a total)
2. The user enters sufficient funds to purchase product, in which case you should:
   - Display a message to indicate the transaction was successful
   - Return the change (even if that amount is zero)

STAGE 9 – IMPLEMENTING FUNCTION `purchase_product()`

In this stage, you will write the necessary code that will allow a user to select a product to purchase (from the vending machine).

Step 1: Let's review what the specification states about the function `purchase_product()`:

> `purchase_product(inventory_list)`
>
> - Accepts a list of inventory *objects* as a parameter
> - Allows the user to purchase a product from the vending machine
> - The function returns nothing
> - You **must use** a loop in your solution

So basically you will need to:

- Prompt the user to select a product (by entering a value between 1 and 10)
- Accept a single parameter `inventory_list` which is a list containing inventory objects
- Call the function `accept_money()` to handle the financial transaction
- Validate the user input
  - An appropriate message should be displayed if incorrect input is entered by the user.
- Use a loop!

Step 2: Write code to **define** the function

Step 3: Write code to accept input from the user (valid choices are 1 – 10).

Step 4: After the user has entered a choice, write code that validates the user input.

- An appropriate message should be displayed if incorrect input is entered by the user. An example is shown below in Sample Output 12.

Step 5: Now add another prompt that:

- Displays the **name** of the product the user selected
- Asks whether this is the correct choice

Sample Output 12

```
Enter selection (1-10): 22
Invalid choice. Please enter numbers between 1 and 10

Enter selection (1-10): 1

You selected Pepsi. Is this correct (y/n): h
Invalid choice. Please choose 'y' or 'n'

You selected Pepsi. Is this correct (y/n): y
```

Step 6: Once you have validated the user input, retrieve the product from the list `inventory_list`.

☞ **Hint:** Use the number entered by the user to access the list `inventory_list`. You will have to modify the actual number the user entered. The reason is that list indexes start at 0 but when we display our products, numbering begins at 1.

Step 7: From the product retrieved in the previous step, extract the **name**, **price** and **quantity**.

Step 8: Check the quantity of the selected inventory item.

- If the quantity is **greater than zero**, allow the transaction.
- If the quantity is **zero**, cancel the transaction and return to the main menu

Step 9: If the transaction is allowed, you should then call the function `accept_money()`

☞ **Remember** that the function `accept_money()` returns the change from the transaction, so be sure to assign the returned *change* to a suitable variable.

⚠ When the user enters 2 from the main **menu**, this function (i.e. `purchase_product()`) should be called from *inside* your `main()` **function**. Don't worry about adding the call to this function from `main()` just yet (we do this in the next stage).

STAGE 10 – INTEGRATING MENU OPTIONS WITH YOUR FUNCTIONS

Congratulations on getting this far! If you've completed all the previous stages, you've pretty much done all the hard work. Now its time to finish off your `main()` function.

Let's start by reviewing what your program should be capable.

Step 1: First, let's list the functions and what they enable your program to do:

`display_main_menu()`

- Responsible for displaying the main menu options (i.e. purchase item and quit)
- Does **not accept** any parameters
- **Prompts** user to make a selection
- **Returns** the choice made by the user

`read_file(filename, inventory_list)`

- Enables your program to read in the contents of a text file

  - Populates list **`inventory_list`** with inventory item objects

`write_to_file(filename, inventory_list)`

  - Enables your program to write the contents of list **`inventory_list`** to a text file

`display_vending_machine()`

  - Responsible for displaying the products in our inventory and their respective prices to the screen

`accept_money()`

  - Responsible for prompting the user to enter coins to pay for their product
  - **Returns** change from the transaction

`purchase_product()`

  - Responsible for prompting the user for a choice (from the list of products being displayed)

Phew! So there you go!

Step 2: Now think about the overall sequence of your code. When your program is run:

- It should first **read** in the contents of the file
- Next your program should **display** the vending machine and below it, the menu options
- Based on user input, your program should then either:
  1. Allow a user to purchase a product; **or**
  2. Allow a user to quit and exit your program
- If the user chooses to buy something from the vending machine, then your program must prompt the user to choose which product they want to buy and then accept their money
- When the transaction is complete, your program should then display the vending machine and the menu options underneath
- This process should continue until the user enters option 2 (to quit) from the main menu

Step 3: At this stage all your main function does at the moment is display a simple message. Let's fix that! Comment out the **`print`** statement (previously added in STAGE 3 – IMPLEMENT FUNCTION **`main()`**)

Step 4: Now you should add code to call function **`read_file()`**

Step 5: Now you should add code to call function **`display_vending_machine()`**

Step 6: Now you should add code to call function **`display_main_menu()`**. Remember this function **returns** the user's choice.

Step 7: Now its time to set up your main program loop. Your program needs to be able to allow a user to continue interacting with your vending machine until they choose to quit (i.e. they enter 2 in the main menu).

- Inside the loop, use an **`if/elif/else`** statement to allow your program to execute along a different path. Check for:
  - Whether the user has entered 1 (i.e. to purchase a product)
    - If the user enters option 1, you should call function **`purchase_item`**
  - Whether the user has entered 2 (i.e. to quit the program)
    - If the user enters option 2, your program should exit

  ☞ Don't forget you also have to call functions **`display_vending_machine()`** and **`display_main_menu()`** here as well

Step 8: The final step to complete your function is to write the contents of the list `inventory_list` to a file. This is now simply a matter of calling your function `write_to_file()`.

☞ This should be called **only** when the user has entered 2 from the main menu.

Final Stage – Fine-Tuning Your Code

So you've written your code and (hopefully) behaves as per the specification. Now all that remains is to:

- Check the sample output (provided as a separate file) and if necessary, modify your code so that:
  - The output produced by your program **EXACTLY** adheres to the sample output provided.
  - Your program behaves as described in these specs and the sample output provided.
- Remove extraneous `print` statements
  - Throughout the development of your program, you will most likely have added numerous `print` statements, and calls to functions to test various behaviours of the code you have implemented. Check your code and remove any print statements that previously served as test output only.
- Ensure you are calling all functions in the right places and remove any function calls **where they shouldn't be** (i.e. used only for testing).

Cleaning up your code

Throughout the development of your program, you will most likely have added numerous `print` statements, and calls to functions to test various behaviours of the code you have implemented along the way. Now it is time to look over your code as a whole and check if the sequencing is correct. For example, check:

- Where/When are reading in the text file?
- Where/When are you writing to file?
- Are your functions accepting the correct number of parameters?
- Are your functions returning the correct value(s)?

Bonus Stages

For those seeking an additional challenge, you can gain extra marks by attempting one or both of the following **bonus** stages.

⚠ Note that you cannot exceed the maximum mark for this part of the assignment.

Bonus Stage – Write a function to Display an Inventory [5 marks]

Write code that adds an additional menu item to the main menu called "Display inventory". When a user enters this menu option the name of each product is displayed followed by how many of each product is left (in the vending machine). The format **must** match the sample provided below:

Sample Output 13 – Bonus

```
1. Purchase item
2. Current Profit
3. Display inventory
4. Quit

Please enter 1-5 to select: 3

 __________________________________________________
| Product            Inventory            Quantity |
|                    =========                     |
|__________________________________________________|
|                                                  |
| 1. Pepsi...................................2     |
|                                                  |
| 2. Coca-Cola...............................3     |
|                                                  |
| 3. Mt. Ogabogee Spring Water...............5     |
|                                                  |
| 4. Corn Chips..............................2     |
|                                                  |
| 5. Fruit Juice.............................1     |
|                                                  |
| 6. Smarties................................6     |
|                                                  |
| 7. Mars-Bar................................7     |
|                                                  |
| 8. Snickers-Bar............................8     |
|                                                  |
| 9. M & M's.................................6     |
|                                                  |
| 10. BBQ Chips..............................6     |
|                                                  |
|__________________________________________________|
```

Submission Details

You are required to do the following in order to submit your work and have it marked:

- You are required to submit an electronic copy of your program via learnonline before *** **INSERT DATE HERE** ***, 9am.
- **You are also required to demonstrate your assignment to your practical supervisor during your week 8 practical class for marking. The supervisor will mark your work using the marking criteria included in this document. You MUST attend the practical session that you have been attending all study period in order to have your assignment marked.**

Assignments submitted to learnonline, but not demonstrated during your allocated practical session, will NOT be marked. Likewise, assignments that have been demonstrated during the practical session, but have not been submitted via learnonline, will NOT be marked. Assignments are submitted to learnonline in order to check for plagiarism.

All students must follow the submission instructions below:

- Ensure that your files are named correctly (as per instructions outlined in this document).
- Ensure that the following files are included in your submission:
  - `yourEmailId_game.py`
  - `yourEmailId_converter.py`

  For example:
  - `bonjy007_game.py`
  - `bonjy007_converter.py`
- All files that you submit must include the following comments.

```
#
# File: fileName.py
# Author: your name
# Email Id: your email id
# Description: Assignment 1 - place assignment description here...
# This is my own work as defined by the University's
# Academic Misconduct policy.
#
```

Assignments that do not contain these details may not be marked.

You must have submitted your program **before the online due date** and demonstrate your work to your marker. You will also be required to demonstrate that you have correctly submitted your work to learnonline. Work that has not been correctly submitted to learnonline will not be marked.

It is expected that students will make copies of all assignments and be able to provide these if required.

Extensions And Late Submissions

There will be **no** extensions/late submissions for this course without one of the following exceptions:

1. A medical certificate is provided that has the timing and duration of the illness and an opinion on how much the student's ability to perform has been compromised by the illness. Please note if this information is not provided the medical certificate WILL NOT BE ACCEPTED. Late assessment items will not be accepted unless a medical certificate is presented to the Course Coordinator. The certificate must be produced as soon as possible and must cover the dates during which the assessment was to be attempted. In the case where you have a valid medical certificate, the due date will be extended by the number of days stated on the certificate up to five working days.

2. A Learning and Teaching Unit councillor contacts the Course Coordinator on your behalf requesting an extension. Normally you would use this if you have events outside your control adversely affecting your course work.
3. Unexpected work commitments. In this case, you will need to attach a letter from your work supervisor with your application stating the impact on your ability to complete your assessment.
4. Military obligations with proof.

Applications for extensions must be lodged via learnonline before the due date of the assignment.

Note: Equipment failure, loss of data, 'Heavy work commitments' or late starting of the course are not sufficient grounds for an extension.

ACADEMIC MISCONDUCT

Students are reminded that they should be aware of the academic misconduct guidelines available from the University of South Australia website.

Deliberate academic misconduct such as plagiarism is subject to penalties. Information about Academic integrity can be found in Section 9 of the Assessment policies and procedures manual at: http://www.unisa.edu.au/policies/manual/

MARKING CRITERIA

USEFUL BUILT–IN PYTHON FUNCTIONS – REQUIRED FOR PART II

You can use the `format()` function to format the way integers and strings are displayed to the screen.

In the following example: `print(format(total_user_score, ’10d’))`

The value assigned to variable `total_user_score` is printed in a field that is 10 spaces wide. By default the number is right-aligned within the field width.

There are other alignment options as follows:

Option	Meaning
'<'	Forces the field to be left-aligned within the available space (this is the default for most objects).
'>'	Forces the field to be right-aligned within the available space (this is the default for numbers).
'∧'	Forces the field to be centered within the available space.

More examples of use (including output):

```
>>> total_user_score = 100
>>> format(total_user_score, '10d')
'       100'
>>> format(total_user_score, '^10d')
'   100    '
>>> format(total_user_score, '<10d')
'100       '
>>> format(total_user_score, '>10d')
'       100'
```

This can be used inside a print function:

```
>>> print(format(total_user_score, '<10d'))
100
>>> print(format(total_user_score, '>10d'))
100
```

Formatting Text (aligning the text and specifying a width)

Examples of use, nested with the print function (including output):

```
>>> format("You", '10s')
'You       '
>>> format("You", '^10s')
'   You    '
>>> format("You", '<10s')
'You       '
>>> format("You", '>10s')
'       You'

>>> print(format("You", '<10s'))
You
>>> print(format("You", '>10s'))
You
```

LECTURES – KEY TOPICS

Week 1	Variables – Creating and Naming
Week 2	User controlled input and output Control structures: if statements
Week 3	Control structures: if statements & while loops
Week 4	Strings More on while loops
Week 5	List data type Control structures: for loops
Week 6	Problem Solving Strategy Debugging
Week 7	Functions – Part 1
Week 8	Functions – Part 2
Week 9	File Input and Output
Week 10	Classes and Objects

Sample Output

Sample Output 14

```
 ___________________________________________________
|                                                   |
|        *** Vending Machine Simulator ***          |
|___________________________________________________|
|                                                   |
| 1. Pepsi                                  $2.50   |
|                                                   |
| 2. Coca-Cola                              $2.55   |
|                                                   |
| 3. Mt. Ogabogee Spring Water              $2.25   |
|                                                   |
| 4. Corn Chips                             $3.00   |
|                                                   |
| 5. Fruit Juice                            $4.00   |
|                                                   |
| 6. Smarties                               $3.50   |
|                                                   |
| 7. Mars-Bar                               $2.40   |
|                                                   |
| 8. Snickers-Bar                           $2.50   |
|                                                   |
| 9. M & M's                                $3.80   |
|                                                   |
| 10. BBQ Chips                             $1.80   |
|                                                   |
|___________________________________________________|
|                                                   |
|      \____________________________________/       |
|                                                   |
|___________________________________________________|

1. Purchase item
2. Quit

Please enter 1-4 to select: 1

Enter selection (1-10): 2

You selected Coca-Cola ($2.55). Is this correct (y/n): y

Please continue to enter coins until amount reached

Valid choices for coins are:
[1] Ten cents
[2] Twenty cents
[3] Fifty cents
[4] One dollar
[5] Cancel

Balance: $0.00 Remaining: $2.55

Enter coin [1-4] or 5 to cancel: 4
Balance: $1.00 Remaining: $1.55

Enter coin [1-4] or 5 to cancel: 4
Balance: $2.00 Remaining: $0.55
```

Sample Output 14 cont...

```
Enter coin [1-4] or 5 to cancel: 2
Balance: $2.20 Remaining: $0.35

Enter coin [1-4] or 5 to cancel: 2
Balance: $2.40 Remaining: $0.15

Enter coin [1-4] or 5 to cancel: 2
Balance: $2.60 Remaining: $0.00
Change: $0.05

Congratulations on your purchase! Please use us again.


 __________________________________________________
|                                                  |
|        *** Vending Machine Simulator ***         |
|__________________________________________________|
|                                                  |
| 1. Pepsi                                  $2.50  |
|                                                  |
| 2. Coca-Cola                              $2.55  |
|                                                  |
| 3. Mt. Ogabogee Spring Water              $2.25  |
|                                                  |
| 4. Corn Chips                             $3.00  |
|                                                  |
| 5. Fruit Juice                            $4.00  |
|                                                  |
| 6. Smarties                               $3.50  |
|                                                  |
| 7. Mars-Bar                               $2.40  |
|                                                  |
| 8. Snickers-Bar                           $2.50  |
|                                                  |
| 9. M & M's                                $3.80  |
|                                                  |
| 10. BBQ Chips                             $1.80  |
|                                                  |
|__________________________________________________|
|                                                  |
|      \___________________________________/       |
|                                                  |
|__________________________________________________|

1. Purchase item
2. Quit

Please enter 1-2 to select: 2

Are you sure [y/n]? y

Goodbye.
```

Sample Output 15

```
 __________________________________________________
|                                                  |
|        *** Vending Machine Simulator ***         |
|__________________________________________________|
|                                                  |
| 1. Pepsi                                  $2.50  |
|                                                  |
| 2. Coca-Cola                              $2.55  |
|                                                  |
| 3. Mt. Ogabogee Spring Water              $2.25  |
|                                                  |
| 4. Corn Chips                             $3.00  |
|                                                  |
| 5. Fruit Juice                            $4.00  |
|                                                  |
| 6. Smarties                               $3.50  |
|                                                  |
| 7. Mars-Bar                               $2.40  |
|                                                  |
| 8. Snickers-Bar                           $2.50  |
|                                                  |
| 9. M & M's                                $3.80  |
|                                                  |
| 10. BBQ Chips                             $1.80  |
|                                                  |
|__________________________________________________|
|                                                  |
|      \___________________________________/       |
|                                                  |
|__________________________________________________|

1. Purchase item
2. Quit

Please enter 1-4 to select: 1

Enter selection (1-10): 10

You selected BBQ Chips ($1.80). Is this correct (y/n): y

Please continue to enter coins until amount reached

Valid choices for coins are:
[1] Ten cents
[2] Twenty cents
[3] Fifty cents
[4] One dollar
[5] Cancel

Balance: $0.00 Remaining: $1.80

Enter coin [1-4] or 5 to cancel: 4
Balance: $1.00 Remaining: $0.80

Enter coin [1-4] or 5 to cancel: 4
Balance: $2.00 Remaining: $0.00
Change: $0.20

Congratulations on your purchase! Please use us again.
```

Sample Output 15 cont. . .

```
 __________________________________________________
|                                                  |
|        *** Vending Machine Simulator ***         |
|__________________________________________________|
|                                                  |
| 1. Pepsi                                 $2.50   |
|                                                  |
| 2. Coca-Cola                             $2.55   |
|                                                  |
| 3. Mt. Ogabogee Spring Water             $2.25   |
|                                                  |
| 4. Corn Chips                            $3.00   |
|                                                  |
| 5. Fruit Juice                           $4.00   |
|                                                  |
| 6. Smarties                              $3.50   |
|                                                  |
| 7. Mars-Bar                              $2.40   |
|                                                  |
| 8. Snickers-Bar                          $2.50   |
|                                                  |
| 9. M & M's                               $3.80   |
|                                                  |
| 10. BBQ Chips                            $1.80   |
|                                                  |
|__________________________________________________|
|                                                  |
|      \____________________________________/      |
|                                                  |
|__________________________________________________|

1. Purchase item
2. Quit

Please enter 1-4 to select: 1

Enter selection (1-10): 1

You selected Pepsi ($2.50). Is this correct (y/n): y

Please continue to enter coins until amount reached

Valid choices for coins are:
[1] Ten cents
[2] Twenty cents
[3] Fifty cents
[4] One dollar
[5] Cancel

Balance: $0.00 Remaining: $2.50

Enter coin [1-4] or 5 to cancel: 4
Balance: $1.00 Remaining: $1.50

Enter coin [1-4] or 5 to cancel: 1
Balance: $1.10 Remaining: $1.40
```

Sample Output 15 cont...

```
Enter coin [1-4] or 5 to cancel: 1
Balance: $1.20 Remaining: $1.30

Enter coin [1-4] or 5 to cancel: 1
Balance: $1.30 Remaining: $1.20

Enter coin [1-4] or 5 to cancel: 5

Transaction cancelled. Refund: $1.30

 ___________________________________________________
|                                                   |
|        *** Vending Machine Simulator ***          |
|___________________________________________________|
|                                                   |
| 1. Pepsi                                  $2.50   |
|                                                   |
| 2. Coca-Cola                              $2.55   |
|                                                   |
| 3. Mt. Ogabogee Spring Water              $2.25   |
|                                                   |
| 4. Corn Chips                             $3.00   |
|                                                   |
| 5. Fruit Juice                            $4.00   |
|                                                   |
| 6. Smarties                               $3.50   |
|                                                   |
| 7. Mars-Bar                               $2.40   |
|                                                   |
| 8. Snickers-Bar                           $2.50   |
|                                                   |
| 9. M & M's                                $3.80   |
|                                                   |
| 10. BBQ Chips                             $1.80   |
|                                                   |
|___________________________________________________|
|                                                   |
|      \____________________________________/       |
|                                                   |
|___________________________________________________|

1. Purchase item
2. Quit

Please enter 1-4 to select: 1

Enter selection (1-10): 2

You selected Coca-Cola ($2.55). Is this correct (y/n): y

Please continue to enter coins until amount reached

Valid choices for coins are:
[1] Ten cents
[2] Twenty cents
[3] Fifty cents
[4] One dollar
[5] Cancel

Balance: $0.00 Remaining: $2.55
```

Sample Output 15 cont. . .

```
Enter coin [1-4] or 5 to cancel: 4
Balance: $1.00 Remaining: $1.55

Enter coin [1-4] or 5 to cancel: 4
Balance: $2.00 Remaining: $0.55

Enter coin [1-4] or 5 to cancel: 2
Balance: $2.20 Remaining: $0.35

Enter coin [1-4] or 5 to cancel: 2
Balance: $2.40 Remaining: $0.15

Enter coin [1-4] or 5 to cancel: 2
Balance: $2.60 Remaining: $0.00
Change: $0.05

Congratulations on your purchase! Please use us again.


 __________________________________________________
|                                                  |
|        *** Vending Machine Simulator ***         |
|__________________________________________________|
|                                                  |
| 1. Pepsi                                 $2.50   |
|                                                  |
| 2. Coca-Cola                             $2.55   |
|                                                  |
| 3. Mt. Ogabogee Spring Water             $2.25   |
|                                                  |
| 4. Corn Chips                            $3.00   |
|                                                  |
| 5. Fruit Juice                           $4.00   |
|                                                  |
| 6. Smarties                              $3.50   |
|                                                  |
| 7. Mars-Bar                              $2.40   |
|                                                  |
| 8. Snickers-Bar                          $2.50   |
|                                                  |
| 9. M & M's                               $3.80   |
|                                                  |
| 10. BBQ Chips                            $1.80   |
|                                                  |
|__________________________________________________|
|                                                  |
|      \____________________________________/      |
|                                                  |
|__________________________________________________|

1. Purchase item
2. Quit

Please enter 1-4 to select: 2

Are you sure [y/n]? y

Goodbye.
```

Sample Output 16

```
 ___________________________________________________
|                                                   |
|        *** Vending Machine Simulator ***          |
|___________________________________________________|
|                                                   |
| 1. Pepsi                                  $2.50   |
|                                                   |
| 2. Coca-Cola                              $2.55   |
|                                                   |
| 3. Mt. Ogabogee Spring Water              $2.25   |
|                                                   |
| 4. Corn Chips                             $3.00   |
|                                                   |
| 5. Fruit Juice                            $4.00   |
|                                                   |
| 6. Smarties                               $3.50   |
|                                                   |
| 7. Mars-Bar                               $2.40   |
|                                                   |
| 8. Snickers-Bar                           $2.50   |
|                                                   |
| 9. M & M's                                $3.80   |
|                                                   |
| 10. BBQ Chips                             $1.80   |
|                                                   |
|___________________________________________________|
|                                                   |
|      \____________________________________/        |
|                                                   |
|___________________________________________________|

1. Purchase item
2. Quit

Please enter 1-4 to select: 1

Enter selection (1-10): 5

Sorry out of Fruit Juice.
 ___________________________________________________
|                                                   |
|        *** Vending Machine Simulator ***          |
|___________________________________________________|
|                                                   |
| 1. Pepsi                                  $2.50   |
|                                                   |
| 2. Coca-Cola                              $2.55   |
|                                                   |
| 3. Mt. Ogabogee Spring Water              $2.25   |
|                                                   |
| 4. Corn Chips                             $3.00   |
|                                                   |
| 5. Fruit Juice                            $4.00   |
|                                                   |
| 6. Smarties                               $3.50   |
|                                                   |
| 7. Mars-Bar                               $2.40   |
|                                                   |
| 8. Snickers-Bar                           $2.50   |
```

Sample Output 16 cont…

```
 |                                                   |
 | 9. M & M's                                $3.80   |
 |                                                   |
 | 10. BBQ Chips                             $1.80   |
 |                                                   |
 |___________________________________________________|
 |                                                   |
 |      \____________________________________/       |
 |                                                   |
 |___________________________________________________|

1. Purchase item
2. Quit

Please enter 1-4 to select: 1

Enter selection (1-10): 3

You selected Mt. Ogabogee Spring Water ($2.25). Is this correct (y/n): y

Please continue to enter coins until amount reached

Valid choices for coins are:
[1] Ten cents
[2] Twenty cents
[3] Fifty cents
[4] One dollar
[5] Cancel

Balance: $0.00 Remaining: $2.25

Enter coin [1-4] or 5 to cancel: 4
Balance: $1.00 Remaining: $1.25

Enter coin [1-4] or 5 to cancel: 4
Balance: $2.00 Remaining: $0.25

Enter coin [1-4] or 5 to cancel: 4
Balance: $3.00 Remaining: $0.00
Change: $0.75

Congratulations on your purchase! Please use us again.


  ___________________________________________________
 |                                                   |
 |        *** Vending Machine Simulator ***          |
 |___________________________________________________|
 |                                                   |
 | 1. Pepsi                                  $2.50   |
 |                                                   |
 | 2. Coca-Cola                              $2.55   |
 |                                                   |
 | 3. Mt. Ogabogee Spring Water              $2.25   |
 |                                                   |
 | 4. Corn Chips                             $3.00   |
 |                                                   |
 | 5. Fruit Juice                            $4.00   |
 |                                                   |
```

Sample Output 16 cont. . .

```
| 6. Smarties                                        $3.50  |
|                                                           |
| 7. Mars-Bar                                        $2.40  |
|                                                           |
| 8. Snickers-Bar                                    $2.50  |
|                                                           |
| 9. M & M's                                         $3.80  |
|                                                           |
| 10. BBQ Chips                                      $1.80  |
|                                                           |
|___________________________________________________________|
|                                                           |
|      \____________________________________/            |
|                                                           |
|___________________________________________________________|

1. Purchase item
2. Current Profit
3. Display inventory
4. Quit

Please enter 1-4 to select: 4

Are you sure [y/n]? y

Goodbye.
```