Assessment Student Instructions

Assessment Title	Assignment 1 - Supermarket Self-Service Checkout

Competency Details

Unit code/s and title/s	ICTPRG443 Apply intermediate programming skills in different languages
Qualification code/s and title/s	ICT40120 - Certificate IV in Information Technology Programming)
Business unit/Work group	Business and Arts/ IT Studies

Instructions

Method/s of assessment	Product – Supermarket Self-Service Checkout Assignment
Overview of assessment	This assessment will require you to interpret a set of requirements specifications to design, develop, build and test a Python Console application to handle a Supermarket Self-Service Checkout at a Point of Sale. The assignment is broken down into 3 parts and you will need to complete all parts.
Task/s to be assessed	You will be assessed on the successful completion Parts 1 – 3 of requirements as follows: Part 1 – Design (UML Models) Part 2 – Coding Part 3 – Testing (Test Cases and Unit Tests)
Time allowed	Refer to your schedule for submission dates
Location of assessment	Assessment can be completed anywhere with access to the resources required. (see Resources Required section below)
Decision making rules	To receive a satisfactory outcome for this assessment you must complete all parts according to the assessment specification
Assessment conditions	This assessment must be undertaken in a workplace or simulated environment where the conditions are typical of those in a working environment in this industry. This is unsupervised assessment, and you may access any required resources. This is not group work and must be completed as an individual
Resources required	To complete this assessment, you will require the following: • Access to Learn with Internet access • Learn resources • Word processing software such as Microsoft Word. • PyCharm 3.8 or later with Python SDK 3.0 • PyCharm, Python SDK and Windows based machines are provided in your practical classes. You can use a Mac if you prefer but these are not provided in the classrooms. • **IT Works Python Coding Standards** • **Unittest Testing Standards**
Result notification and reassessment information	You will be provided feedback and the result for your assignment on TAFESA Learn. You will be and given the chance to resubmit with required corrections only once.

	Refer to the TAFE SA assessment policy for more information https://www.tafesa.edu.au/apply-enrol/before-starting/student-policies/assessment

Assignment 1 – Supermarket Self-Service Checkout

Requirements Specification – Overview

You have been hired by **ITWorks** as a developer for a **Console** based prototype to simulate a Supermarket Self-Service Checkout system.

The team's system analyst has decided to use an **Object Oriented (OO)** approach that will require the use of several **Classes** to implement the final system.

The project analyst/designers have requested that the details of all Product and Transaction entities related to this system be saved in text files until a decision can be made as to what database should be used to store these two entities.

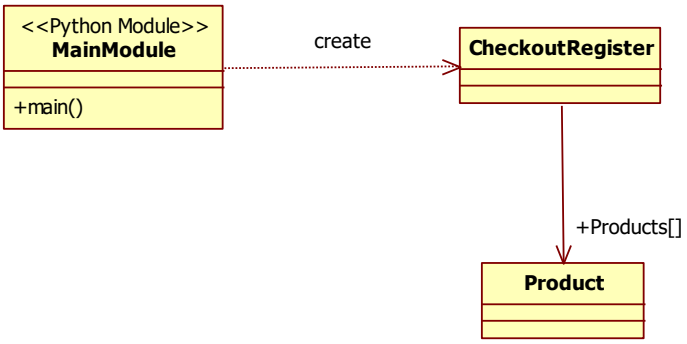
Use the following example **Workflow** (Fig 1.0) as a guide to understand a typical checkout scenario

Please enter the barcode of your item: 123
Milk,2 Litres - $3.50
Would you like to scan another item? (Y/N): Y
Please enter the barcode of your item: 456
Bread, 500g - $2.50
Would you like to scan another item? (Y/N): Y
Please enter the barcode of your item: 999
ERROR!! – scanned barcode is incorrect
Would you like to scan another item? (Y/N): n
Payment due: $5.50 Please enter an amount to pay: 5
Payment due: $0.50 Please enter an amount to pay: -3
ERROR!! – Negative amounts are not accepted
Payment due: $5.50 Please enter an amount to pay: 2
----- Final Receipt-----
Milk, 2Litres $3.00
Bread, 500g $2.50
Total amount due: $5.50
Amount received: $7.50
Balance given: $1.50

(Fig 1.0- Workflow Output)

Part 1 – Design (UML Models)

The project Analyst/Designers have adopted an Agile development approach, and at one of the 'Sprint Sessions' recommended the development of Python components as shown in the UML model below



UML Model – Product Class

Following the above recommendation of the Analyst/Designers, identify the **Product** class **fields** (attributes) and **methods** and refine the above **Product** class to include these fields and methods. Use Star UML to create the Product class model. The classes names, method and field names should conform to the **IT Works Python Coding Standards**

UML Model – CheckoutRegister Class

The methods of the **CheckoutRegister** class have been investigated at a previous **Agile Sprint** session and the following methods were identified:

(1**) init ( )** # Loads product data from the Products text file into a List of Product objects

(2) **scan_item**(product_Barcode)

(3) **accept_payment**(amount_paid)

(4) **print_receipt** ()

(5) **save_transction** (date, Barcode, amount)

NOTE*: you may require additional methods to help modularise your design further in keeping with the* **IT Works Python Coding Standards**

Using the above information modify the **CheckoutRegister class** to include the above methods and aggregate the Product and List of Product classes and any other fields you might require. Model the modified class using Star UML.

Documentation

Copy and paste the above modified UML models (Product and CheckoutRegister) you created using Star UML, into a word document named '**Software Architecture Document'**. You will also use this document to include other project requirements specified later.

Part 2 - Coding

Your team will use **PyCharm** to develop the coded solution and the required SDK version would be **Python 3.x.**

The senior software architect has recommended the following **Implementation** (coding) requirements.

1. Develop the Module/Classes in Python as defined in your completed UML Class Model including fields (attributes) and methods. Your Python code should adhere to the ITWorks Python Coding Standards with regards to the documentation and maintenance of your application code.
2. Use the **main( )** in the **Main Module** to execute the workflow shown in Fig 1.0
3. Modularize your code by adopting the **IT Works Python Coding Standards to** create the five separate methods required in the **CheckoutRegister** class , taking into account the passing parameters by value vs. reference, to **read** and **write** data from and to files ***and use content managers*** to control the opening (creating) and closing files (destroying) to optimize memory management
4. The System should accommodate the scanning of multiple products for multiple customers.
5. The System will also require the implementation of **data validation** checks to handle invalid input data, using an appropriate iteration structure (see Fig 1.0 workflow for guidance) and use Proper Exception Handling code in the relevant sections of coded solution
6. Use the **Python** language **Comment** features to document/explain all five methods of the **CheckoutRegister** class.
7. The system will required access to **two text files**, one to hold the product data (barcode, description, price) and the other to store each sale transaction (date, product bar code)

Part 3 – Testing (Test Cases and Unit Tests)

1. Your testing strategy would require you create several unit tests using **Python's unittest framework** to test the CheckoutRegister class.
2. Create a separate **Test Module** in **PyCharm** and implement the required test methods with proper unit test names as recommended in the Unittest Testing Standards.

   When creating your unit tests, use the **Unittest Testing Standards** for setting up a proper test scaffold using unittest to test all the getter methods of the Product class and the following methods of the CheckoutRegister class:

i. scan_item()
ii. accept_payment()
iii. init( )

All tests must record as a 'Pass' when run

3. Use the **PyCharm Debugging** feature to add at least **two break points** in your **CheckoutRegister** class methods and trace the execution of your code by including at least two captured images in the **Software Architecture Document** you created in **Part 1.**

4. Create a **Test Plan** and confirm compliance with program specification, by running and recording the results of relevant test **cases** as defined in (2) above. Document this in the **Software Architecture Document** you created in **Part 1**

Submission Details

Create a folder **<ID_NUM>_<NAME>_<Assign1>** (example: 3002323_JohnDoe_Assign1) and include your

- **Software Architecture Document** and
- Your **Python** project files, which includes the
    - **Main** module,
    - **Product** class
    - **CheckoutRegister** class
    - **product.txt**
    - **transactions.txt**

Zip the above folder and upload to Learn using the appropriate link.