SPECIAL COURSE IN SOFTWARE ENGINEERING, AUTUMN 2024

EXERCISE 1 – PYTHON BASICS

In this exercise, you will create a simple program for an online shopping store.

THIS EXERCISE WILL BE GRADED (MAX: 10 points)

SUBMISSION DEADLINE: SEPTEMBER 22, 2024, MIDNIGHT

Task 1. Use the 'products' dictionary

Your program MUST use the '**products'** dictionary. You can add more products and categories if you want, but what has been provided to you should not be modified in any way.

Task 2. The main() function

- Define a **main()** function. The definition is already given in the repository you must have cloned.
- The **main()** function is where the program will start executing. Just like in Java.
- As soon as the program starts, ask the user their name and then their email address.
- Validate the name using the **validate_name(name)** function that's given to you. Fill the code to make sure that the user provides both first name and last name, and that these names contain only alphabets.
    - If the name is invalid, ask the user to enter a valid name again.
- Next, validate the email address using **validate_email(email)** function. The email address must have **@** sign in the address.
- After validating both the name and email address, the program should first show only the categories of products available for shopping. Use **display_categories()** function to do this.
    - Ask the user which category they would like to explore. Each category should be numbered, so that the user has to enter only numbers. If the user enters an incorrect number, inform the user and ask for a correct entry.
    - Once the user selects a category, the program should show all the products available under that category, including the price. The products should be numbered. Use the **display-products(products_list)** function to do this.
    - After displaying the product details, give user the option as follows

1. Select a product to buy

2. Sort the products according to the price.
3. Go back to the category selection.
4. Finish shopping

- If the user selects 1, then ask the user to enter the number corresponding to the product. Once the user enters a valid number, ask the user to enter the quantity they want to buy. The user should enter only numbers. Allow the users to shop for as many products as they like. Use the **add_to_cart(cart, product, quantity)** function to do this.
- If the user selects 2, which is the option to sort products by price, first ask the user if they want to sort ascending (1) or descending (2). Depending on what the user chooses, display all the products in that ascending/descending order. Use **display_sorted_products(product_list, sort_order)** function to do the sorting. Then display again the options to select a product, sort the products, go back to the category level, and finish shopping.
- If the user chooses 3, to go back to the category level, then go one level up where you show only the categories and the option to choose a single category. Just like you do at the beginning.
- If the user chooses 4, it means the user is done shopping. Depending on whether the user ordered a product, the program will show do different things.
  - If the user bought any product, even if it was just one, that product must be entered into a cart. Use a **list** to create this cart. So, after the user chooses 4, display the contents of this **cart.** Use the **display_cart(cart)** function to display the contents of the cart.
  - Then show the total cost of their shopping.
  - Ask them a delivery address.
  - Then generate a receipt where all the products, the quanity, and their total cost is displayed. And a message that the order will be delivered in 3 days, and the payment will be charged after the delivery is successful. Use **generate_receipt(name, email, cart, total_cost, address)** function to generate the receipt with all the necessary details
  - If the user did not buy a product, do not generate any receipt. Simply display a message like ***'Thank you for using our portal. Hope you buy something from us next time. Have a nice day***'

Task 3. display_categories()function

This function is responsible for simply displaying the categories from the products dictionary in a **numbered** format. Meaning, the display should be something like…

1. *IT products*
2. *Electronics*

…

This function does not take any argument and it does ***not return*** anything.

Task 4. display_products(products_list) function

This function takes the values from the **products_list** argument and displays them in a numbered format. These values will include the product name and the price as well.

This function will ***not return*** anything.

Task 5. display_sorted_products(products_list, sort_order) function

This function sorts the items present in the **products_list** depending on the user choice of ordering, which will be in the **sort_order** argument.

This function must ***return*** the sorted items.

Task 6. add_to_cart(cart, product, quantity) function

This function takes the product details from the **product** argument, and the quantity value from the **quantity** argument and adds them to the shopping cart list, which is the **cart** argument**.**

This function will ***not return*** anything.

Task 7. generate_receipt(name, email, cart, total_cost, address) function

This function simply displays the name (**name** argument), email address (**email** argument), and then the list of products purchased, which comes from the **cart** argument. At the end, the total cost (**total_cost** argument) is displayed. Next, the delivery address (**address** argument) is displayed. The last statement "***Your items will be delivered in 3 days. Payment will be accepted after successful delivery***" must be displayed at the end of the receipt.

Task 8. validate_name(name) function

This function simply takes the name argument and validates it to check they are made of two parts, first name and last name. And there are only alphabets.

Depending on the validation, the function will ***return*** either ***True*** or ***False***.

Task 9. validate_email(email) function

Same logic as **validate_name()** function.