\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC121 PYTHON Programming**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LAB 12 Modular design

# Objectives

In this lab assignment, students will learn:

- How to use top-down approach to design a program

- How to use docstring for function specification

- How to create modules in Python

- How to import Python modules

# Goals

In this lab assignment, students will demonstrate the abilities to:

- Use top-down approach to design a program

- Use docstring for function specification

- Create modules in Python

- Import Python modules

# Instruction and Problems

## Problem 1

This problem is about modular design. We are writing a program to simulate a self-checkout system of a store named Wake-Mart.

Input prices

Process discount

Process payment

Process promotion

Main

Cash

Debit

This program has four tasks at the top level: input prices of items, process discount, process promotion and process payment. Processing of payment is further divided into two subtasks. The customer can choose either to pay by cash or by debit card.

You must write the following functions to implement the design above.

|  |  |
| --- | --- |
| Function | Specification |
| scanPrices() | This function gets the price of each item purchased. The customer enters the prices one by one with a loop. When there are no more prices to enter, 0 is entered to exit the loop. Whenever a negative price is entered, display “Price cannot be negative” and ask the user to enter next price. Every time a valid price is entered, display the number of prices entered and the total so far (exclude invalid entries). Return the number of prices entered and the total price of all items when the user has finished entering prices. |
| discount(count, total) | This function gives a 10% discount if 10 or more items are purchased. It receives items count and total as arguments. It checks whether 10 or more items are purchased, and reduces the total by 10% if the 10 or more items requirement is met. This function returns the total, either it has been changed or not. |
| promotion(count, total) | This function allows the user to buy a gift card with discount. It receives items count and total as arguments. If the total is $50 or higher, the user can choose to buy one $50 gift card for the price of $40. Update items count and total if the user chooses to buy gift card. This function returns items count and total, either they have been changed or not. |
| makePayment(balance) | This function allows the user to choose payment type [1 for cash, 2 for debit]. It receives balance as argument, and pass it to pay\_cash and pay\_debit. |
| payCash(balance) | This function receives cash payment. The self-checkout machine only accepts $10, $5 and $1 bills. Ask the user how many $10, $5 and $1 bills he is going to use. Calculate and display total payment. If the total payment is lower than the balance, ask the user to reenter the numbers of $10, $5 and $1 bills until the total payment is not lower than the balance. If customer has paid more than the balance, calculate and display change. |
| payDebit(balance) | This function receives debit payment. It asks customer to enter a 16-digit card number and a 4-digit PIN. It also asks customer to enter payment amount. Use a validation loop to ensure that payment is not lower than the balance. If payment is lower than the balance, ask the user to reenter payment amount until it is not lower than the balance. If payment is higher than balance, calculate and display cash back amount. |
| main() | This function calls the scanPrices function to input item prices, calls the discount function to process discount, calls the promotion function to process gift card promotion, and call the makePayment function to process payment. |

The following is an example. This customer chooses to pay cash.

Welcome to the self-checkout system of Wake-mart

Enter price of first item [or 0 to stop]: 7

Number of items: 1 Total: 7.0

Enter price of next item [or 0 to stop]: -2

Price cannot be negative

Enter price of next item [or 0 to stop]: -5

Price cannot be negative

Enter price of next item [or 0 to stop]: 8

Number of items: 2 Total: 15.0

Enter price of next item [or 0 to stop]: 2.5

Number of items: 3 Total: 17.5

Enter price of next item [or 0 to stop]: 8

Number of items: 4 Total: 25.5

Enter price of next item [or 0 to stop]: 7

Number of items: 5 Total: 32.5

Enter price of next item [or 0 to stop]: 5

Number of items: 6 Total: 37.5

Enter price of next item [or 0 to stop]: 4.25

Number of items: 7 Total: 41.75

Enter price of next item [or 0 to stop]: 8

Number of items: 8 Total: 49.75

Enter price of next item [or 0 to stop]: 8.25

Number of items: 9 Total: 58.0

Enter price of next item [or 0 to stop]: 2

Number of items: 10 Total: 60.0

Enter price of next item [or 0 to stop]: 4

Number of items: 11 Total: 64.0

Enter price of next item [or 0 to stop]: 6

Number of items: 12 Total: 70.0

Enter price of next item [or 0 to stop]: 0

you've got 10% disocunt for buying 10 items of more.

Number of items: 12 Total: 63.0

Do you want to buy a $50 gift card for $40? [y/n] y

Thank you for buying a giftcard.

Number of items: 13 Total: 103.0

Payment options:

Enter 1 for cash, 2 for debit: 1

This machine only accepts $10, $5 and $1 bills.

How many $10 bills are you going to pay? 9

How many $5 bills are you going to pay? 1

How many $1 bills are you going to pay? 1

Error: Total cash payment less than balance. Please reenter.

How many $10 bills are you going to pay? 9

How many $5 bills are you going to pay? 3

How many $1 bills are you going to pay? 0

Total cash paid: 105

Thank you for your payment.

Change: 2.0

The following is another example. This customer chooses to pay by debit.

Welcome to the self-checkout system of Wake-mart

Enter price of first item [or 0 to stop]: 7

Number of items: 1 Total: 7.0

Enter price of next item [or 0 to stop]: -2

Price cannot be negative

Enter price of next item [or 0 to stop]: -4

Price cannot be negative

Enter price of next item [or 0 to stop]: 5

Number of items: 2 Total: 12.0

Enter price of next item [or 0 to stop]: 6

Number of items: 3 Total: 18.0

Enter price of next item [or 0 to stop]: 8

Number of items: 4 Total: 26.0

Enter price of next item [or 0 to stop]: 1

Number of items: 5 Total: 27.0

Enter price of next item [or 0 to stop]: 5

Number of items: 6 Total: 32.0

Enter price of next item [or 0 to stop]: 6

Number of items: 7 Total: 38.0

Enter price of next item [or 0 to stop]: 8

Number of items: 8 Total: 46.0

Enter price of next item [or 0 to stop]: 9

Number of items: 9 Total: 55.0

Enter price of next item [or 0 to stop]: 5

Number of items: 10 Total: 60.0

Enter price of next item [or 0 to stop]: 0

you've got 10% disocunt for buying 10 items of more.

Number of items: 10 Total: 54.0

Do you want to buy a $50 gift card for $40? [y/n] n

Payment options:

Enter 1 for cash, 2 for debit: 2

Please enter a 16-digit card number: 1234567812345678

Please enter 4-digit pin: 9988

Please enter payment amount: 45

ERROR: Payment amount cannot be smaller than balance

Please enter payment amount: 54

Thank you for your payment.

Save your Python program in a file named **Lab12P1.py**. Submit the file for credit.

## Problem 2

This problem is about Python modules.

Crate a module currency, which includes the following three functions that do currency conversions:

to\_euro(dollar): This function receives US Dollar as an argument and converts it to Euro. 1 US Dollar = 0.81 Euro. Return Euro.

to\_yen(dollar): This function receives US Dollar as an argument and converts it to Japanese Yen. 1 US Dollar = 106.45 Yen. Return Yen.

to\_peso(dollar): This function receives US Dollar as an argument and converts it to Mexican Peso. 1 US Dollar = 18.58 Peso. Return Peso.

Store these three functions in a file named **currency.py**.

Create a file for the main module. Name the file **lab12P2.py**.

Define a main function in the main module to do the following:

1. Ask the user to choose a foreign currency: Euro, Japanese Yen or Mexican Peso.

Write a loop to validate user input. If an invalid choice is made, display an error message and ask the user to choose a foreign currency again until the choice is valid.

1. Ask the user to enter US dollar amount. Write a loop to validate user input. If the US dollar amount is negative, display an error message and ask the user to reenter it until it is non-negative.
2. Call one of the three functions in the currency module to convert US dollar to the foreign currency chosen by the user
3. Receive and display the converted foreign currency

The following is an example.

Converting US Dollar to a foreign currency.

Enter 1 for Euro, 2 for Japanese Yen, 3 for Mexican Peso: 4

Error: Invalid Choice

Enter 1 for Euro, 2 for Japanese Yen, 3 for Mexican Peso: 5

Error: Invalid Choice

Enter 1 for Euro, 2 for Japanese Yen, 3 for Mexican Peso: 2

Enter US Dollar: -100

Error: US Dollar cannot be negative.

Enter US Dollar: -200

Error: US Dollar cannot be negative.

Enter US Dollar: 100

It is converted to 10645.0 Yen

Submit the files **currency.py** and **lab12P2.py** for credits.

# Grading rubric for Program 1

Writing scanPrices function [10 points]

Writing discount function [10 points]

Writing promotion function [10 points]

Writing makePayment function [10 points]

Writing payCash function [10 points]

Writing payDebit function [10 points]

Writing main function [10 points]

# Grading rubric for Program 2

Writing currency module [15 points]

Writing main module [15 points]