I started my Journey of Master of Information Technology as follows:

Trimester 1

In "Software Development Studio 1"

* Became familiar to use Processing environment
* Completed Generative Artwork in Processing Environment where I needed to code an art where some kinds of animation type are shown.
* Completed Fly Swatter Game in Processing Environment where there is a swatter to kills the appropriate fly when it appears in the screen.
* Completed Missile Command Game in Processing Environment where there are bombs raining from top and there is a defence missile at the bottom and some cities like structure which needs to be saved from those bombs.

In "Introduction to Programming and UNIX Environment"

* I did 3 python programs of which the readme.md files are provided below:

1. Readme.md for 1st assignment

Codetown Cinema

This Python script is a simple ticket sales tracker that prompts the user for the number of adult, child, and concession tickets sold, calculates the total sale for the transaction, and keeps track of the total sales and number of tickets sold for each ticket type.

Usage

1. Run the script in a Python environment.

2. It will display "Is there a new ticket sale? (yes/no): "

3. As per the entry,

1. If you enter "no", you will be re-directed to step 6.

2. If you enter any other thing rather than "yes", you will be asked to enter either "yes" or "no" only and re-directed to step 2.

3. If you enter "yes", the program goes to next step.

4. It will ask for "Number of adult tickets: "

5. As per entry,

1. If you enter any value rather than number, it says "Please enter a integer number for adult tickets." and goes to step 3.4

2. If you enter any number less than 0, it says "Please enter a positive integer or 0 for adult tickets." and goes to step 3.4

3. If you enter any number >= 0, than it saves the value of adult tickets.

6. It will ask for "Number of child tickets: "

7. As per entry,

1. If you enter any value rather than number, it says "Please enter a integer number for child tickets." and goes to step 3.6

2. If you enter any number less than 0, it says "Please enter a positive integer or 0 for child tickets." and goes to step 3.6

3. If you enter any number >= 0, than it saves the value of child tickets.

8. It will ask for "Number of concession tickets: "

9. As per entry,

1. If you enter any value rather than number, it says "Please enter a integer number for concession tickets." and goes to step 3.8

2. If you enter any number less than 0, it says "Please enter a positive integer or 0 for concession tickets." and goes to step 3.8

3. If you enter any number >= 0, than it saves the value of adult tickets.

10. If you enter 0 for all three tickets values, you will be displayed a message "There must be at least one ticket in the transaction!" and will be redirected to step 3.4.

4. The script will calculate the total sale for the transaction, update the total sales and number of tickets sold for each ticket type, and display the total due for the transaction.

5. Repeat from step 2 until you are finished entering ticket sales.

6. The script will display the total sales and number of tickets sold for each ticket type.

Variables used in the program

`adult\_price`: the price of an adult ticket (integer)

`child\_price`: the price of a child ticket (integer)

`concession\_price`: the price of a concession ticket (float)

`total\_sales`: the total sales (float) formatted to two decimal places

`total\_adult\_tickets`: the total number of adult tickets sold (integer)

`total\_child\_tickets`: the total number of child tickets sold (integer)

`total\_concession\_tickets`: the total number of concession tickets sold (integer)

`adult\_tickets` : the number of adult tickets (integer)

`child\_tickets` : the number of child tickets (integer)

`concession\_tickets` : the number of concession tickets (integer)

Example 1

```

Is there a new ticket sale? (yes/no): yes

Number of adult tickets: 1

Number of child tickets: 1

Number of concession tickets: 1

Total due for transaction: $52.50

Is there a new ticket sale? (yes/no): no

Total sales: $52.50

Adult tickets sold: 1

Child tickets sold: 1

Concession tickets sold: 1

```

Example 2

```

Is there a new ticket sale? (yes/no): yes

Number of adult tickets: 1

Number of child tickets: 0

Number of concession tickets: 0

Total due for transaction: $20.00

Is there a new ticket sale? (yes/no): yes

Number of adult tickets: 1

Number of child tickets: 1

Number of concession tickets: 0

Total due for transaction: $35.00

Is there a new ticket sale? (yes/no): yes

Number of adult tickets: 0

Number of child tickets: 0

Number of concession tickets: 1

Total due for transaction: $17.50

Is there a new ticket sale? (yes/no): no

Total sales: $72.50

Adult tickets sold: 2

Child tickets sold: 1

Concession tickets sold: 1

```

Example 3

```

Is there a new ticket sale? (yes/no): maybe

Please enter "yes" or "no"

Is there a new ticket sale? (yes/no): n

Please enter "yes" or "no"

Is there a new ticket sale? (yes/no): no

Total sales: $0.00

Adult tickets sold: 0

Child tickets sold: 0

Concession tickets sold: 0

```

Example 4

```

Is there a new ticket sale? (yes/no): y

Please enter "yes" or "no"

Is there a new ticket sale? (yes/no): yes

Number of adult tickets: None

Please enter a number for adult tickets.

Number of adult tickets:

Please enter a number for adult tickets.

Number of adult tickets: -1

Please enter a positive integer or 0 for adult tickets.

Number of adult tickets: 0

Number of child tickets: None

Please enter a number for child tickets.

Number of child tickets:

Please enter a number for child tickets.

Number of child tickets: -1

Please enter a positive integer or 0 for child tickets.

Number of child tickets: 0

Number of concession tickets: None

Please enter a number for concession tickets.

Number of concession tickets:

Please enter a number for concession tickets.

Number of concession tickets: -1

Please enter a positive integer or 0 for concession tickets.

Number of concession tickets: 0

There must be at least one ticket in the transaction!

Number of adult tickets: None

Please enter a number for adult tickets.

Number of adult tickets:

Please enter a number for adult tickets.

Number of adult tickets: -1

Please enter a positive integer or 0 for adult tickets.

Number of adult tickets: 0

Number of child tickets:

Please enter a number for child tickets.

Number of child tickets: 0

Number of concession tickets: -1

Please enter a positive integer or 0 for concession tickets.

Number of concession tickets: 1

Total due for transaction: $17.50

Is there a new ticket sale? (yes/no): n

Please enter "yes" or "no"

Is there a new ticket sale? (yes/no): no

Total sales: $17.50

Adult tickets sold: 0

Child tickets sold: 0

Concession tickets sold: 1

```

1. Readme.md for 2nd assignment

Introduction:<br>

This program reads sales data from a file named 'transactions.txt' and returns the total sales for each movie in a dictionary.

The program calculates the transaction amount for each line and accumulates the sales data for each movie.

Finally, it sorts the sales data by total sales in descending order and prints it to the console.

How to use:<br>

To use this program, follow the instructions below:

1. Install Python 3.x on your computer if it is not already installed.

2. Open the terminal or command prompt on your computer.

3. Navigate to the directory where the 'transactions.txt' file is located.

4. Run the program by typing the following command:

`python3 sales.py`

5. The program will read the 'transactions.txt' file and print the sorted sales data to the console.

Function Description:<br>

The program consists of one function named 'read\_sales\_data', which takes a file name as an argument and returns a dictionary containing the sales data for each movie.

The function reads each line of the file, splits it into separate values, and calculates the transaction amount for each line.

It then accumulates the sales data for each movie and returns the resulting dictionary.

If the file format is incorrect, or if the ticket counts are negative or not integers, the function raises a 'ValueError' with a corresponding error message.

Variables:<br>

The program uses the following variables:<br>

`adult\_price`: A float variable representing the price of an adult ticket.<br>

`child\_price`: A float variable representing the price of a child ticket.<br>

`concession\_price`: A float variable representing the price of a concession ticket.<br>

`movie\_name`: A String variable representing the name of the movie.<br>

`transaction\_amount`: A float variable representing the calculated transaction amount.<br>

These variables are used to calculate the transaction amount for each movie name in the file.

Error Handling:<br>

If an error occurs while running the program, it will print an error message to the console.

Specifically, if there is an issue with the file format or ticket counts, the function will raise a 'ValueError' with a corresponding error message.

The main program catches any 'Exception' that is raised and prints the error message to the console.

Conclusion:<br>

This program reads sales data from a file, calculates the transaction amount for each line, accumulates the sales data for each movie.

The program also performs error handling by raising 'ValueError' if the file format or ticket counts are incorrect.

1. Readme.md for 3rd assignment

Introduction:

This code is an implementation of a graphical user interface (GUI) for a cinema ticket booking system. The GUI is created using the tkinter library in Python. The user can input the number of tickets for each ticket type (adult, child, and concession) and the cost of the transaction is calculated based on the ticket prices and the number of tickets sold. The GUI also has a reset button to clear the input fields and a calculate button to perform the cost calculation.

Instructions:

Before running the program,

Install Python in the system

Import tkinter

Import messagebox

To run the program:

Enter the below code in bash/terminal/command prompt.

python3 tickets\_gui.py

How to Use:

Upon launching the application, a window titled "Code Cinema" will appear by default in 350X150 size.

Enter the number of adult, child, and concession tickets in the respective entry fields.

The "Calculate" button will be enabled only when all entry fields are filled.

The "Reset" button will be enabled only if there is any contents inside any entry fields.

Click the "Calculate" button to calculate the total cost of the tickets.

The calculated cost will be displayed after the "Cost" label.

The focus is set to "Calculate" button to ensure there is no unintended input after the calculate button is pressed.

To reset the transaction, click the "Reset" button. It will clear the entry fields and cost value and disable the "Calculate" button and "Reset" button.

You can resize the application window as per your choice.

You can exit the application by closing the window or pressing Ctrl+C in the terminal/command prompt.

Note:

The application validates the input to ensure that the number of tickets is a positive integer or zero.

If invalid input is entered, appropriate error messages will be displayed.

The application validates the input to ensure that there is at least one of the tickets are sold.

If all entry is 0 then, error message saying "It seems like there is no any ticket sold." will be displayed.

Overall, this code is a simple implementation of a "Code Cinema" ticket booking system using GUI in Python with tkinter.

In "Object Oriented Programming",

I did JAVA programming with 3 assignments.

It was like stepping into deeper JAVA learning from simple to complex from assignment 1 to 3.

In assignment 1,

It was just creating a normal simple JAVA program for “SeekAGeek” store where they can sell different kinds of car seat covers and users can search as they like.

In assignment 2,

It was same with more advanced form where now they can search in more advanced form and using more advanced search

In assignment 3,

It was now with other accessories as well…

In "Database Management Systems"

* Assignment 1,

CREATE TABLE customers

(

customer\_id INTEGER,

last\_name VARCHAR(50) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

address VARCHAR(200),

city VARCHAR(50),

state CHAR(3) NOT NULL CHECK (state IN ('NSW', 'VIC', 'QLD', 'ACT',

'TAS', 'NT',

'SA', 'WA')),

postcode VARCHAR(8),

PRIMARY KEY (customer\_id)

);

CREATE TABLE movies

(

movie\_id INTEGER,

movie\_title VARCHAR(100) NOT NULL,

director\_last\_name VARCHAR(50) NOT NULL,

director\_first\_name VARCHAR(50) NOT NULL,

genre VARCHAR(20) CHECK(genre IN ('Action', 'Adventure',

'Comedy', 'Romance',

'Science Fiction',

'Documentary', 'Drama', 'Horror')) NOT NULL,

release\_date DATE,

studio\_name VARCHAR(50),

PRIMARY KEY (movie\_id)

);

CREATE TABLE stock

(

movie\_id INTEGER,

media\_type VARCHAR(20) CHECK (media\_type IN ('DVD', 'Blu-Ray',

'Stream-Media')),

cost\_price REAL CHECK(cost\_price > 0) NOT NULL,

retail\_price REAL CHECK(retail\_price > 0) NOT NULL,

current\_stock REAL CHECK(current\_stock >= 0) NOT NULL,

PRIMARY KEY (movie\_id, media\_type),

FOREIGN KEY (movie\_id) REFERENCES movies(movie\_id) ON DELETE CASCADE ON

UPDATE CASCADE

);

CREATE TABLE shipments

(

shipment\_id INTEGER,

customer\_id INTEGER NOT NULL,

movie\_id INTEGER NOT NULL,

media\_type VARCHAR(20) NOT NULL,

shipment\_date DATE,

PRIMARY KEY (shipment\_id),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE

CASCADE ON UPDATE CASCADE,

FOREIGN KEY (movie\_id, media\_type) REFERENCES stock(movie\_id, media\_type)

ON DELETE CASCADE ON UPDATE CASCADE

);

* Assignment 2,

-- COSC210 Practical Assignment Template

-- Please complete the assignment questions using the view templates

-- provided below.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- IMPORTANT

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Make sure that you do not alter the names of the views or their

-- attribute values. If you do your assignment will not work in the

-- auto-marking software and you may lose marks!

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CREATE VIEW movie\_summary(movie\_title, release\_date, media\_type, retail\_price)

AS

SELECT m.movie\_title, m.release\_date, s.media\_type, s.retail\_price

FROM movies m

JOIN stock s ON m.movie\_id = s.movie\_id;

CREATE VIEW old\_shipments(first\_name, last\_name, movie\_id, shipment\_id, shipment\_date)

AS

SELECT c.first\_name, c.last\_name, s.movie\_id, s.shipment\_id, s.shipment\_date

FROM customers c

JOIN shipments s ON c.customer\_id = s.customer\_id

WHERE s.shipment\_date < '2010-01-01';

CREATE VIEW trilogy(movie\_title)

AS

SELECT movie\_title

FROM movies

WHERE movie\_title LIKE '%Rings%' OR movie\_title LIKE '%Wars%';

CREATE VIEW retail\_price\_hike(movie\_id , retail\_price, new\_price)

AS

SELECT movie\_id, retail\_price, Round((retail\_price \* 1.25)::numeric, 2)

FROM stock;

CREATE VIEW value\_summary(total\_stock\_cost, total\_retail\_cost)

AS

SELECT SUM(s.cost\_price \* s.current\_stock),

SUM(s.retail\_price \* s.current\_stock)

FROM stock s

JOIN movies m ON s.movie\_id = m.movie\_id

WHERE s.media\_type <> 'Stream-Media';

CREATE VIEW profits\_from\_movie(movie\_id, movie\_title, total\_profit)

AS

SELECT m.movie\_id, m.movie\_title, SUM(s.retail\_price) - SUM(s.cost\_price)

FROM movies m

JOIN stock s ON m.movie\_id = s.movie\_id

JOIN shipments sh ON s.movie\_id = sh.movie\_id AND s.media\_type = sh.media\_type

GROUP BY m.movie\_id, m.movie\_title;

CREATE VIEW followers\_of\_melkor AS

SELECT c.first\_name, c.last\_name

FROM customers c

WHERE c.customer\_id NOT IN (

SELECT s.customer\_id

FROM shipments s

JOIN stock st ON s.movie\_id = st.movie\_id AND s.media\_type = st.media\_type

JOIN movies m ON st.movie\_id = m.movie\_id

WHERE m.movie\_title = 'The Lord of the Rings: The Fellowship of the Ring'

);

CREATE VIEW sole\_angry\_watcher(first\_name, last\_name)

AS

SELECT c.first\_name, c.last\_name

FROM customers c

JOIN shipments s ON c.customer\_id = s.customer\_id

JOIN movies m ON s.movie\_id = m.movie\_id

WHERE m.movie\_title = '12 Angry Men'

GROUP BY c.customer\_id, c.first\_name, c.last\_name

HAVING COUNT(\*) = 1;

Now moving to Trimester 2,