

School of Computing, Creative Technology and Engineering

**Module: Fundamentals of Computer Programming**

**Academic Year: 2023/24**

**Level 4: Semester 1**

**Assignment Title: Academic Report**

**Date Due: 28th January 2024**

**Tutor: Mr. Ashish Acharya**

**Student Name: Rajendra Rana Magar**

**Student ID: 77466861**

**Report Writing of Task1, Task2 and Task3**

# Abstract

As a student, I recognize significant potential for growth in these projects. It wasn't just about coding, it involved grasping how a system could intelligently calculate pizza prices based on user preferences. Next, the Cat Shelter Log Analysis script served as a guide in adeptly handling data and unforeseen errors.

In the User Management System, I explored the complexities of encryption by using ROT 13. Managing users has become more than just a technical task, it provided me insight into the importance of security and encryption in the digital industry. These projects, beyond being mere coding exercises, offered engaging learning experiences.

**Table of Contents**

[Abstract 2](#_Toc157366088)

[Introduction 4](#_Toc157366089)

[Pizza Price Calculator 4](#_Toc157366090)

[Overview 4](#_Toc157366091)

[Brief 4](#_Toc157366092)

[Testing 5](#_Toc157366093)

[Cat Shelter Log Analysis 5](#_Toc157366094)

[Overview 5](#_Toc157366095)

[Brief 5](#_Toc157366096)

[Error Handling 6](#_Toc157366097)

[Testing 6](#_Toc157366098)

[User Management System 6](#_Toc157366099)

[Overview 6](#_Toc157366100)

[Brief 6](#_Toc157366101)

[Error Handling 8](#_Toc157366102)

[Testing 8](#_Toc157366103)

[Source Code 9](#_Toc157366104)

[Task1 9](#_Toc157366105)

[Task2 11](#_Toc157366106)

[Task3 12](#_Toc157366107)

# Introduction

This report explores three Python scripts: Pizza Price Calculator, Cat Shelter Log Analysis, and User Management System. The Pizza Price Calculator introduces user input and pricing logic, the Cat Shelter Log Analysis script focuses on data interpretation and error handling, and the User Management System deals with user administration through ROT13 encryption.

# Pizza Price Calculator

## Overview

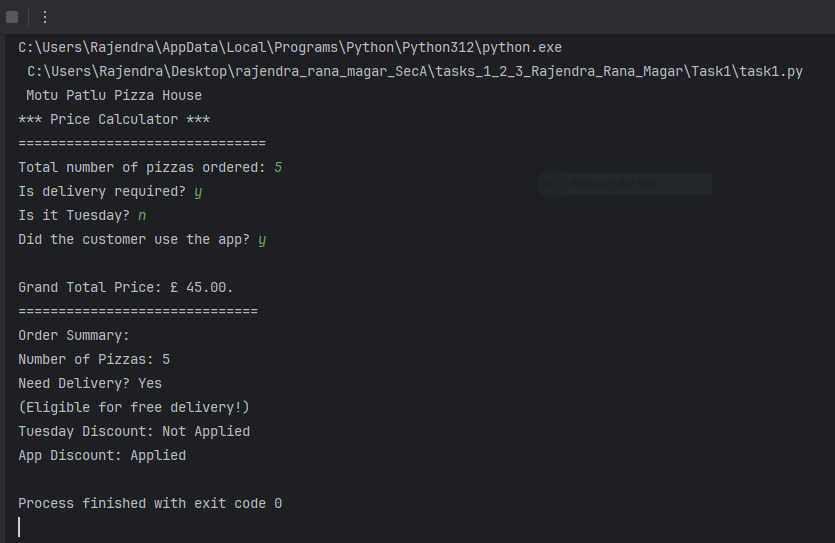
The provided Python script serves as a price calculator for a pizza ordering system. It includes functions for obtaining user input, such as the quantity of pizzas, delivery preferences, and whether the order falls on a Tuesday or if the customer used a specific app.

## Brief

1. **my\_functions.py**
   1. **def get\_positive\_integer\_input(prompt):**
      1. Ensures the input is a positive integer.
      2. Displays appropriate error messages for invalid inputs.
   2. **def get\_yes\_or\_no\_input(prompt):**
      1. Accepts 'yes/no' or 'y/n' input, returning a boolean value.
      2. Provides clear instructions for valid responses.
   3. **def calculate\_price(pizza\_count, delivery\_opt, is\_tuesday, used\_app):**
      1. Calculates the total price based on various factors (quantity, delivery option, day, and app usage).
      2. Applies discounts or charges accordingly.
2. **Task1.py**
   * 1. Imports my\_functions.py script as module.
     2. Utilizes the input functions to collect user information.
     3. Calls the calculate\_price function to determine the grand total price.
     4. Displays an order summary including the number of pizzas, delivery details, and applied discounts.

## Testing

To run the program, execute task1.py. Enter relevant details about the pizza order. The program will then display the grand total price and an order summary.



# Cat Shelter Log Analysis

## Overview

The provided Python script analyses a cat shelter log file, providing insights into the visits of both the user's cat and intruder cats. The analysis includes the total number of visits, total time spent in the shelter by the user's cat, and statistics on visit durations.

## Brief

1. **Task2.py**
   1. **def format\_time(minutes):**
      1. A utility function that formats the given duration in minutes into a human-readable format (e.g., "02 hrs : 45 mins").
   2. **def analyze\_cat\_shelter(log\_filename):**
      1. Reads and analyses the cat shelter log file specified as a command line argument.
      2. Counts visits for the user's cat and intruder cats.
      3. Calculates total time in the shelter for the user's cat and provides statistics on visit durations (average, longest, shortest).

## Error Handling

* Missing command line argument.
* FileNotFoundError: If the specified log file is not found.
* Other exceptions: Displays a generic error message.

## Testing

To run the program, open command prompt, type python task2.py with log file name as parameter. Then program will analyse and then display cat shelter information.

testing task2.py


# User Management System

## Overview

The User Management System script is designed to facilitate user administration operations within a simple interactive interface. The system incorporates functionalities such as adding new users, deleting existing users, changing passwords, and user login. Passwords are encrypted using the ROT13 method to ensure a basic level of security.

## Brief

1. **add\_user.py():**

This script adds a new user to the system:

* + - 1. Prompts the user to enter a new username, full name, and password.
      2. Checks if the entered username already exists in the system. If it does, the user is prompted to choose a different username.
      3. Encrypts the user's password using ROT13.
      4. Appends the encrypted user details (username, full name, and password) to the password file.

1. **del\_user.py():**

This script deletes an existing user from the system:

* + - 1. Prompts the user to enter the username they wish to delete.
      2. Checks if the entered username exists in the system. If it does, the user is prompted to confirm the deletion by entering 'Y' or 'N'.
      3. If confirmed, removes the corresponding user details from the password file.

1. **change\_password.py()**

This script allows users to change their password:

* + - 1. Prompts the user to enter their current username and password.
      2. Validates the entered password using ROT13.
      3. If the current password is validated, prompts the user to enter a new password. Confirms the new password and updates the encrypted password in the system.

1. **Login.py**()

This script grants or denies access based on username and password validation:

* + 1. Prompts the user to enter their username and password.
    2. Checks if the entered username exists and if the password matches the encrypted password in the system using ROT13.
    3. If validation is successful, access is granted; otherwise, access is denied.

1. **Task3.py()**

This is main menu script serves as the entry point:

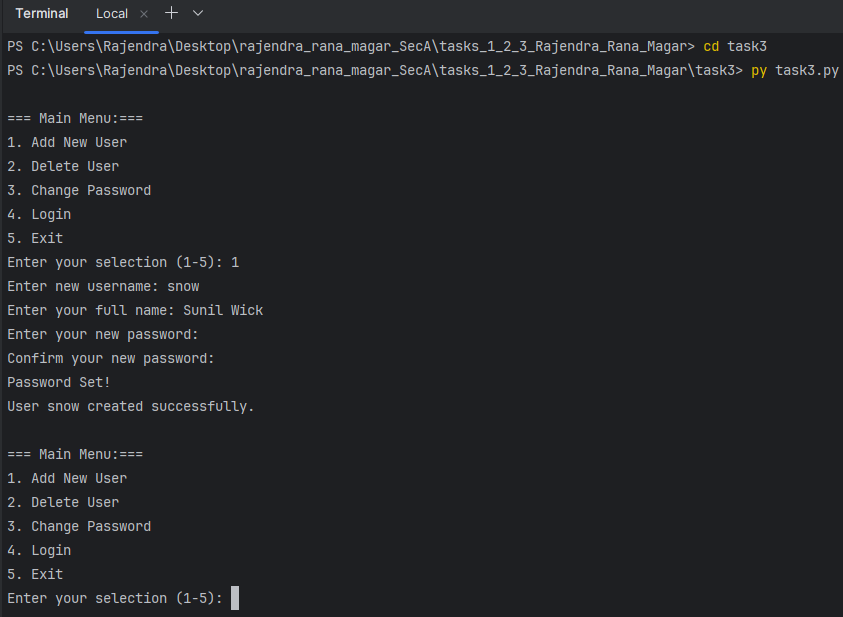
* + 1. Imports add\_user.py, del\_user.py, change\_password.py and login.py scripts as module.
    2. Provides a simple interface for executing different functionalities.
    3. Users can add new users, delete users, change passwords, log in, or exit the program.

## Error Handling

* Handles cases where usernames already exist during user addition.
* Deletion and password change functions confirm actions through password prompts.

## Testing

To run the program, execute the task3.py. The program will then display the corresponding selection. Enter relevant details about user to perform then the program displays the desired output, and write the user details in passwd.txt file.



# Source Code

## Task1

**my\_functions.py**

def get\_positive\_integer\_input(prompt):  
 *"""Only takes positive integer as valid input."""* while True:  
 try:  
 value = int(input(prompt))  
 if value > 0:  
 return value  
 elif value == 0:  
 print("Are you sure! Please confirm the order again.")  
 else:  
 print("Error! Please enter in a positive integer.")  
 except ValueError:  
 print("Error! Please enter in a valid number.")  
  
  
def get\_yes\_or\_no\_input(prompt):  
 *""" Only takes 'yes/no' or 'y/n' input, returns in bool."""* while True:  
 response = input(prompt).lower()  
 if response in ['y', 'yes']:  
 return True  
 elif response in ['n', 'no']:  
 return False  
 else:  
 print('Please answer in "Yes/No" or "Y/N".')  
  
  
def calculate\_price(pizza\_count, delivery\_opt, is\_tuesday, used\_app):  
 *"""Calculates total price, applies discounts or charges if conditions are valid."""* price\_per\_pizza = 12.00  
 delivery\_cost = 2.50  
 discount\_tuesday = 0.50  
 discount\_app = 0.25  
  
 total\_price = price\_per\_pizza \* pizza\_count  
  
 if is\_tuesday:  
 total\_price -= total\_price \* discount\_tuesday  
  
 if delivery\_opt:  
 if pizza\_count >= 5:  
 total\_price += 0 # Free delivery for 5 or more pizzas  
 else:  
 total\_price += delivery\_cost  
  
 if used\_app:  
 total\_price -= total\_price \* discount\_app  
  
 return round(total\_price, 2)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("checking if my\_function are working correctly:)")  
  
 test1 = get\_positive\_integer\_input("Order Quantity")  
 print(test1)  
  
 test2 = get\_yes\_or\_no\_input("Y/N")  
 print(test2)  
  
 p\_count = 4  
 d\_opt = True  
 is\_tue = True  
 app = True  
  
 test3 = calculate\_price(p\_count, d\_opt, is\_tue, app)  
 print(test3)

**task1.py**

from my\_functions import get\_positive\_integer\_input, get\_yes\_or\_no\_input, calculate\_price  
  
  
def main():  
 *"""Main function to run price calculator program."""* print(" Motu Patlu Pizza House\n"  
 "\*\*\* Price Calculator \*\*\*\n===============================")  
  
 pizza\_count = get\_positive\_integer\_input("Total number of pizzas ordered: ")  
 delivery\_opt = get\_yes\_or\_no\_input("Is delivery required? ")  
 is\_tuesday = get\_yes\_or\_no\_input("Is it Tuesday? ")  
 used\_app = get\_yes\_or\_no\_input("Did the customer use the app? ")  
  
 grand\_total\_price = calculate\_price(pizza\_count, delivery\_opt, is\_tuesday, used\_app)  
  
 print(f"\nGrand Total Price: £ {grand\_total\_price:.2f}.")  
 print("==============================\nOrder Summary:")  
 print(f"Number of Pizzas: {pizza\_count}")  
 print(f"Need Delivery? {'Yes' if delivery\_opt else 'No'}"  
 f"{'\n(Eligible for free delivery!)' if pizza\_count > 4 else ''}")  
 print(f"Tuesday Discount: {'Applied' if is\_tuesday else 'Not Applied'}")  
 print(f"App Discount: {'Applied' if used\_app else 'Not Applied'}")  
  
  
main()

## Task2

**task2.py**

import sys  
  
  
def format\_time(minutes):  
 *"""Formats minutes in 02 hrs : 45 min """* hours = minutes // 60  
 remaining\_minutes = minutes % 60  
  
 if hours == 0:  
 return f"{remaining\_minutes:02d} mins"  
 else:  
 return f"{hours:02d} hrs : {remaining\_minutes:02d} mins"  
  
  
def analyze\_cat\_shelter(log\_filename):  
 *"""Analyzes the cat shelter log file. """* try:  
 with open(log\_filename, 'r') as file:  
 lines = file.readlines()  
  
 visits\_my\_cat = 0  
 visits\_intruder\_cats = 0  
 total\_time\_in\_house = 0  
 visit\_durations = []  
  
 for line in lines:  
 if line.strip() == 'END':  
 break  
  
 data = line.strip().split(',')  
 cat\_type, entry\_time, exit\_time = data  
  
 entry\_time = int(entry\_time)  
 exit\_time = int(exit\_time)  
  
 duration = exit\_time - entry\_time  
  
 if cat\_type == 'OURS':  
 visits\_my\_cat += 1  
 total\_time\_in\_house += duration  
 visit\_durations.append(duration)  
 elif cat\_type == 'THEIRS':  
 visits\_intruder\_cats += 1  
  
 if visits\_my\_cat > 0:  
 average\_duration = sum(visit\_durations) // visits\_my\_cat  
 longest\_duration = max(visit\_durations)  
 shortest\_duration = min(visit\_durations)  
 else:  
 average\_duration = longest\_duration = shortest\_duration = 0  
  
 print("==================\nLog File Analysis\n==================")  
 print(f"Total Visits of My Cat: {visits\_my\_cat}")  
 print(f"Total Visits of Intruder Cats: {visits\_intruder\_cats}")  
 print(f"\n=========================\nShelter Info About My Cat\n=========================")  
 print(f"Total Time in Shelter: {format\_time(total\_time\_in\_house)}")  
 print(f"\nAverage Visit Length: {format\_time(average\_duration)}")  
 print(f"Longest Visit: {format\_time(longest\_duration)}")  
 print(f"Shortest Visit: {format\_time(shortest\_duration)}")  
  
 except FileNotFoundError:  
 print(f'Cannot open "{log\_filename}"!')  
 except Exception as e:  
 print(f'An error occurred: {e}')  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 if len(sys.argv) != 2:  
 print("Missing command line argument!")  
 else:  
 input\_log\_filename = sys.argv[1]  
 analyze\_cat\_shelter(input\_log\_filename)

## Task3

**task3.py**

from add\_user import add\_user  
from del\_user import del\_user  
from change\_password import change\_password  
from login import login  
  
  
def main\_menu():  
 *"""Main program starts here."""* while True:  
 print("\n=== Main Menu:===")  
 print("1. Add New User")  
 print("2. Delete User")  
 print("3. Change Password")  
 print("4. Login")  
 print("5. Exit")  
  
 choice = input("Enter your selection (1-5): ")  
  
 if choice == "1":  
 add\_user()  
 elif choice == "2":  
 del\_user()  
 elif choice == "3":  
 change\_password()  
 elif choice == "4":  
 login()  
 elif choice == "5":  
 print("Exiting program.")  
 break  
 else:  
 print("Invalid selection. Please enter a number between 1 and 5.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main\_menu()

**add\_user.py**

from encryption import rot13  
from getpass import getpass  
  
password\_file = "passwd.txt"  
  
  
def add\_user():  
 *"""Adds only new users to the database."""* while True:  
 username = input("Enter new username: ").lower()  
  
 with open(password\_file, "r") as file:  
 if any(line.startswith(username + ":") for line in file):  
 print("Username already exists. Please choose a different username.")  
 continue  
  
 real\_name = input("Enter your full name: ")  
  
 while True:  
 new\_password = getpass("Enter your new password: ")  
 confirm\_password = getpass("Confirm your new password: ")  
  
 if new\_password == confirm\_password:  
 print("Password Set!")  
 break  
 else:  
 print("Error! Passwords do not match. Try again.")  
  
 with open(password\_file, "a") as file:  
 encrypted\_password = rot13(confirm\_password)  
 file.write(f"{username}:{real\_name}:{encrypted\_password}\n")  
  
 print(f"User {username} created successfully.")  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 add\_user()

**del\_user.py**

password\_file = "passwd.txt"  
  
  
def del\_user():  
 *"""Deletes an existing user."""* username = input("Enter username to delete: ").lower()  
  
 with open(password\_file, "r") as file:  
 lines = file.readlines()  
  
 user\_found = False  
  
 for line in lines:  
 parts = line.strip().split(":")  
 if not parts[0].lower() == username:  
 continue  
  
 user\_found = True  
 password = input("Confirm deletion (Y/N): ").lower()  
 if password in ['y', 'yes']:  
 lines.remove(line)  
 print(f"User {username} Deleted.")  
 else:  
 print(f"Deletion canceled.")  
 return  
  
 if not user\_found:  
 print("User not found. Nothing changed.")  
 return  
  
 with open(password\_file, "w") as file:  
 file.writelines(lines)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 del\_user()

**login.py**

from encryption import rot13  
from getpass import getpass  
  
password\_file = "passwd.txt"  
  
  
def login():  
 *"""Gives access to the user if entered correct passwords."""* username = input("Enter username: ").lower()  
 password = getpass("Enter password: ")  
  
 with open(password\_file, "r") as file:  
 for line in file:  
 parts = line.strip().split(":")  
 if parts[0] == username:  
 if parts[2] == rot13(password):  
 print("Access granted.")  
 return  
 else:  
 print("Access denied.")  
 return  
  
 print(f"sorry! Username {username} do not exist.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 login()

**change\_password.py**

from encryption import rot13  
from getpass import getpass  
  
password\_file = "passwd.txt"  
  
  
def change\_password():  
 *"""Changes current passwords if entered correctly."""* username = input("Enter your username: ").lower()  
  
 with open(password\_file, "r") as file:  
 lines = file.readlines()  
  
 for i, line in enumerate(lines):  
 parts = line.strip().split(":")  
 if parts[0].lower() == username:  
 current\_password = getpass("Enter your current password: ")  
 encrypted\_current\_password = rot13(current\_password)  
  
 if encrypted\_current\_password == parts[2]:  
 new\_password = getpass("Enter your new password: ")  
 confirm\_password = getpass("Confirm your new password: ")  
  
 if new\_password == confirm\_password:  
 encrypted\_new\_password = rot13(new\_password)  
 lines[i] = f"{username}:{parts[1]}:{encrypted\_new\_password}\n"  
 with open(password\_file, "w") as file:  
 file.writelines(lines)  
 print("Password changed successfully.")  
 return  
 else:  
 print("Error! Passwords do not match. Password change canceled.")  
 return  
 else:  
 print("Incorrect current password. Password change canceled.")  
 return  
  
 print("Username not found. Password change canceled.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 change\_password()

**encryption.py**

import codecs  
  
  
def rot13(text):  
 *"""Encrypts the passwords"""* return codecs.encode(text, 'rot\_13')