CIND830 - Python Programming for Data Science

Assignment 1 (15% of the final grade)

Due on February 19, 2024, 11:59 PM

This is a Jupyter Notebook document that extends a simple formatting syntax for authoring HTML and PDF. Review this website for more details on using Jupyter Notebooks.

Consider using a Jupyter Notebook platform to complete this assignment. Ensure using **Python 3.9 release or higher**, then complete the assignment by inserting your Python code wherever seeing the string `#STRAHINJA NAKIC.` `

You are expected to submit the notebook file (in IPYNB format) and the exported version (either in PDF or HTML) in the same Assignment link in D2L. Use these guidelines to submit **both** the IPYNB and the exported file (HTML or PDF). Failing to submit both files will be subject to mark deduction.

Please be advised that you cannot get more than 100% in this assignment, and the **BONUS** question (if there is any) will only be graded if all other questions have been submitted.

Coverage:

1. Data Types and Expressions
2. Repetition Statements
3. Selection Statements
4. Strings, Lists, Tuples and Text Files

Question 1 [40 pts]:

a) [20 pts] Write code to reverse an integer number entered by the user, and to give the square root of the reversed number. For example, if the user enters `144`, the output should be `441` and `21.0` (output on separate lines).

Note that the program should not use the `[::-1]` method to reverse a number or string.

```
def reverse_integer(number):
    reversed_number = 0
    while number != 0:
        digit = number % 10
        reversed_number = reversed_number * 10 + digit
        number //= 10
    return reversed_number

def main():
    try:
        num = int(input("Enter an integer number: "))
        reversed_num = reverse_integer(num)
        print(reversed_num)
        sqrt_reversed_num = reversed_num ** 0.5
        print(sqrt_reversed_num)
    except ValueError:
        print("Please enter a valid integer.")

if __name__ == "__main__":
    main()
```

```
 Enter an integer number: 5
 5
 2.23606797749979
```

b) [20 pts] There exists a property called **Reversible Square Root** for certain numbers. These numbers are such that when you square them,

then reverse the resulting digits, take the square root of that number and reverse the digits again, the result is the same as the original number.

For example, the number 12 is a Reversible Square Root number because if you square it (12^2=144), reverse the digits of the result (which gives 441), take the square root of 441 (which is $\sqrt{441}$=21), and reverse the digits again, you get 12.

Write a program to find all the Reversible Square Root numbers between 1 and 5000, inclusively. The program should display these numbers on the same line, separated by a comma. The last number should not have a comma following it.

Note that the program should not use the `[::-1]` method to reverse a number or string.

```
def reverse_number(number):
    reversed_num = 0
    while number > 0:
        reversed_num = reversed_num * 10 + number % 10
        number //= 10
    return reversed_num

def is_reversible_square_root(num):
    squared = num * num
    reversed_squared = reverse_number(squared)
    sqrt_reversed_squared = reversed_squared ** 0.5
    reversed_sqrt_reversed_squared = reverse_number(int(sqrt_reversed_squared))
    return num == reversed_sqrt_reversed_squared

def main():
    reversible_numbers = []
    for num in range(1, 5001):
        if is_reversible_square_root(num):
            reversible_numbers.append(num)
    print(','.join(map(str, reversible_numbers)))

if __name__ == "__main__":
    main()
```

```
1,2,3,11,12,13,21,22,31,101,102,103,111,112,113,121,122,201,202,211,212,221,301,311,1001,1002,1003,1011,1012,1013,1021,1022,1
```

Question 2 [40 pts]:

Create a text file named `books.txt`. Each line in this file represents a book with its title and price, separated by a comma. For example:

```
"Python Data Science Handbook, $82.61",
"Automate the Boring Stuff with Python, $59.99",
"Python for Data Analysis, $107.25",
"Think Python, $65.69",
"Fluent Python, $98.53",
"Effective Python, $62.18",
"Learning Python, $49.99",
"Python Crash Course, $65.99",
"Introduction to Machine Learning with Python, $80.43",
"Data Science from Scratch, $88.48"
```

Guidelines and Hints

- Use string methods for parsing data.
- Utilize lists to store and manipulate book data.
- Implement loops for iterating over the list of books.
- Apply conditional statements to perform checks and comparisons.
- Remember to close the file after reading or writing.

a- [8 pts]

1. Write a Python script to open books.txt.
2. Read each line of the file.
3. Split each line into the book title and its price.
4. Store these in a list of tuples, where each tuple contains the title and price of a book.

```
## I created a file first in order to test my code

## Opening the file in write mode
with open('books.txt', 'w') as file:
    # Writing each book and its price to the file
    file.write("Python Data Science Handbook, $82.61\n")
    file.write("Automate the Boring Stuff with Python, $59.99\n")
    file.write("Python for Data Analysis, $107.25\n")
    file.write("Think Python, $65.69\n")
    file.write("Fluent Python, $98.53\n")
    file.write("Effective Python, $62.18\n")
    file.write("Learning Python, $49.99\n")
    file.write("Python Crash Course, $65.99\n")
    file.write("Introduction to Machine Learning with Python, $80.43\n")
    file.write("Data Science from Scratch, $88.48\n")

## Opening the file in read mode
with open('books.txt', 'r') as file:
    ## Initializing an empty list to store book data
    books = []

    ## Reading each line of the file
    for line in file:
        ## Removing leading/trailing whitespaces and newline characters
        line = line.strip()

        ## Spliting each line into title and price using comma as delimiter
        title, price = line.split(", ")

        ## Removing quotation marks from title
        title = title.strip('"')

        ## Removing dollar sign from price and convert to float
        price = float(price.strip('$'))

        ## Appending the tuple (title, price) to the list of books
        books.append((title, price))

## Printing the list of books
print(books)
```

```
[('Python Data Science Handbook', 82.61), ('Automate the Boring Stuff with Python', 59.99), ('Python for Data Analysis', 107.
```

b- [32 pts]

Each of the following questions should be addressed in a separate Python script.

1. Inventory Summary

- How many books are in the inventory?
- What is the total value of the inventory?

2. Pricing Analysis

- What is the average price of a book?
- List all books priced above the average price.

3. Search and Selection

- Find and display the title of the most expensive book.
- Display all books with a title containing the word "The"

- Display all books with a title containing the word "The".

4. Data Manipulation

- Increase the price of all books by 10%.
- Create a new file updated_books.txt with the updated prices.

Double-click (or enter) to edit

Book data

books_data = ["Python Data Science Handbook, $82.61", "Automate the Boring Stuff with Python, $59.99", "Python for Data Analysis, $107.25", "Think Python, $65.69", "Fluent Python, $98.53", "Effective Python, $62.18", "Learning Python, $49.99", "Python Crash Course, $65.99", "Introduction to Machine Learning with Python, $80.43", "Data Science from Scratch, $88.48"]

Count the number of books using a regular loop

num_books = 0 for book in books_data: num_books += 1

print("Number of books in inventory:", num_books)

```
## Opening the file in read mode
with open('books.txt', 'r') as file:
    ## Initializing a variable to count the number of books
    num_books = 0
    ## Iterating over each line in the file
    for line in file:
        # Incrementing the count for each line
        num_books += 1

print("Number of books in the inventory:", num_books)

## Initializing the total value of the inventory
total_value = 0

## Opening the file in read mode
with open('books.txt', 'r') as file:
    ## Iterating over each line in the file
    for line in file:
        ## Splitting the line by comma to separate title and price
        title, price = line.strip().split(', ')
        ## Extracting the price by removing the dollar sign and converting to float
        price = float(price.replace('$', ''))
        ## Adding the price to the total value
        total_value += price

## Printing the total value of the inventory
print("Total value of the inventory: $", format(total_value, '.2f'))
```

```
 Number of books in the inventory: 10
 Total value of the inventory: $ 761.14
```

```
## Initializing the total value of the inventory
total_value = 0

## Initializing the count of books
num_books = 0

## Opening the file in read mode
with open('books.txt', 'r') as file:
    # Iterating over each line in the file
    for line in file:
        ## Splitting the line by comma to separate title and price
        title, price = line.strip().split(', ')
        ## Extracting the price by removing the dollar sign and converting to float
        price = float(price.replace('$', ''))
```

```
        ## Adding the price to the total value
        total_value += price
        ## Incrementing the count of books
        num_books += 1

## Calculating the average price of a book
average_price = total_value / num_books

## Printing the average price of a book
print("Average price of a book: $", format(average_price, '.2f'))

## Initializing variables
total_value = 0
num_books = 0
books_above_average = []

## Opening the file in read mode
with open('books.txt', 'r') as file:
    ## Iterating over each line in the file
    for line in file:
        ## Splitting the line into title and price
        title, price_str = line.strip().split(', ')
        ## Removing the dollar sign and convert price to float
        price = float(price_str.replace('$', ''))
        ## Accumulating total value and count of books
        total_value += price
        num_books += 1

## Calculating average price
average_price = total_value / num_books

## Opening the file again to iterate over it
with open('books.txt', 'r') as file:
    ## Iterating over each line in the file
    for line in file:
        ## Splitting the line into title and price
        title, price_str = line.strip().split(', ')
        ## Removing the dollar sign and convert price to float
        price = float(price_str.replace('$', ''))
        ## Checking if the price is above average
        if price > average_price:
            # Adding book title and price to the list
            books_above_average.append((title, price))

## Printing the list of books priced above the average price
print("Books priced above the average price:")
for book_title, book_price in books_above_average:
    print(book_title, "- $", book_price)
```

```
Average price of a book: $ 76.11
Books priced above the average price:
Python Data Science Handbook - $ 82.61
Python for Data Analysis - $ 107.25
Fluent Python - $ 98.53
Introduction to Machine Learning with Python - $ 80.43
Data Science from Scratch - $ 88.48
```

```
## Initializing variables
max_price = 0
most_expensive_title = ""

## Opening the file in read mode
with open('books.txt', 'r') as file:
    ## Iterating over each line in the file
    for line in file:
        ## Splitting the line into title and price
        title, price_str = line.strip().split(', ')
        ## Removing the dollar sign and convert price to float
        price = float(price_str.replace('$', ''))
        ## Checking if the price is greater than the current maximum price
        if price > max_price:
```

```
        if price > max_price:
            # Updating the maximum price and most expensive title
            max_price = price
            most_expensive_title = title

## Printing the title of the most expensive book
print("The title of the most expensive book is:", most_expensive_title)

## Opening the file again to iterate over it
with open('books.txt', 'r') as file:
    ## Printing a separator for readability
    print("\nBooks with title containing 'The':")
    ## Iterating over each line in the file
    for line in file:
        ## Splitting the line into title and price
        title, _ = line.strip().split(', ')
        ## Checking if the title contains the word "The", but also "the"
        if "the" in title.lower():
            # Printing the title of the book containing the word "The"
            print(title)
```

```
The title of the most expensive book is: Python for Data Analysis

Books with title containing 'The':
Automate the Boring Stuff with Python
```

```
## Increasing the price of all books by 10%
books = [(title, round(price * 1.10, 2)) for title, price in books]

## Printing the updated prices
for title, price in books:
    print(f"{title}: ${price:.2f}")

## Writing the updated prices to a new file
with open('updated_books.txt', 'w') as file:
    file.writelines([f"{title}: ${price:.2f}\n" for title, price in books])
print("Updated prices have been written to updated_books.txt")
```

```
Python Data Science Handbook: $90.87
Automate the Boring Stuff with Python: $65.99
Python for Data Analysis: $117.98
Think Python: $72.26
Fluent Python: $108.38
Effective Python: $68.40
Learning Python: $54.99
Python Crash Course: $72.59
Introduction to Machine Learning with Python: $88.47
Data Science from Scratch: $97.33
Updated prices have been written to updated_books.txt
```

Question 3 [20 pts].

Create a Control Flow Diagram

1. Choose one of the following questions from Q2.b that you have already answered:
   - What is the average price of a book?
   - List all books priced above the average price.
   - Find and display the title of the most expensive book.
   - Display all books with a title containing the word "The".
2. Using pen and paper, sketch a control flow diagram that represents the logic and flow of the Python code for the chosen question. Your diagram should include:
   - Start and end points.
   - Sequential steps in the process.
   - Decision points,if any, where the flow of control depends on a condition.

- Arrows to indicate the flow of the program.

3. Once your diagram is complete, take a clear photograph or scan it.
4. Convert the image to a PNG or JPG format, if it isn't already.
5. Upload the image file along with your Python IPYNB file when submitting your assignment.

This is the end of assignment 1