Practical-7

February 20, 2025

1 Practical 7 :-

Name: Sarthak Sanay

Enrollment No: 230031101611051

1.1 Problem Statement 1 :-

Stock Price Variation Tracker: Write a function `track_price_variation` that takes a list of daily stock prices and returns a list of tuples. Each tuple should contain the day number (starting from 0), the price for that day, and the difference in price from the previous day. For the first day, the difference can be listed as `None`.

[15]:
```
def track_price_variation(stock_price):
    variation = [(0, stock_price[0], None)]
    for i in range(1, len(stock_price)):
        daily = (i, stock_price[i], (stock_price[i] - stock_price[i-1]))
        variation.append(daily)
    return variation

# Runner code
stock_price = [80, 88, 90, 94, 86, 91, 100, 105, 96, 98]
variation = track_price_variation(stock_price)
print(f"Variation in Stock price is as follows :-\n{variation}")
```

```
Variation in Stock price is as follows :-
[(0, 80, None), (1, 88, 8), (2, 90, 2), (3, 94, 4), (4, 86, -8), (5, 91, 5), (6,
100, 9), (7, 105, 5), (8, 96, -9), (9, 98, 2)]
```

1.2 Problem Statement 2 :-

Todo List Manager: Create a function `manage_todo` that takes a list of to-do items and a command ('add', 'remove', 'complete') and modifies the list based on the command.

[16]:
```
def manage_todo(todo_list, command, task):
    if command == 'add':
        todo_list.append(task)
        print(f'\nTask "{task}" added successfully in To-Do List!')
        return todo_list
```

```
    elif command == 'remove':
        if task in todo_list:
            todo_list.remove(task)
            print(f'\nTask "{task}" removed successfully from To-Do List!')
        else:
            print(f'\nTask "{task}" does not exist in the To Do List!')
        return todo_list

    elif command == 'complete':
        if task in todo_list:
            index = todo_list.index(task)
            todo_list[index] = task + " -> (Completed!)"
            print(f'\nTask "{task}" marked as completed!')
            return todo_list
        else:
            print(f'\nTask "{task}" does not exist in the To Do List!')
            return todo_list

# Runner code
todo_list = ['Finish assignment', 'Buy snacks', 'Workout']
print(manage_todo(todo_list, 'add', 'Clean the room'))
print(manage_todo(todo_list, 'remove', 'Buy snacks'))
print(manage_todo(todo_list, 'complete', 'Finish assignment'))
```

```
Task "Clean the room" added successfully in To-Do List!
['Finish assignment', 'Buy snacks', 'Workout', 'Clean the room']

Task "Buy snacks" removed successfully from To-Do List!
['Finish assignment', 'Workout', 'Clean the room']

Task "Finish assignment" marked as completed!
['Finish assignment -> (Completed!)', 'Workout', 'Clean the room']
```

1.3 Problem Statement 3 :-

Book Collection Sorter: Write a function to sort books. It should take a list of book titles and sort them by the length of the title, with the shortest title first.

[17]:
```
def sort_booklist(booklist):
    # Bubble sort implementation for manual sorting, based on title length
    for i in range(len(booklist)):
        for j in range(i+1, len(booklist)):
            if len(booklist[i]) > len(booklist[j]):
                booklist[i], booklist[j] = booklist[j], booklist[i] # Swap␣
 ↪operation
    return booklist
```

```
# Runner code
booklist = ['Software Engineering: a practitioners approach',
            'Python Programming: Using Problem Solving Approach',
            'Introduction to ALgorithms',
            'Algorithm Design',
            'The Design and Analysis of Computer Algorithms',
            'Introduction to modern cryptography',
            'Handbook of Applied Cryptography',
            'Python Programming - Learn & Practice']

i = 0
print("Books before sorting :-\n")
for book in booklist:
    print(f"{i+1}. {book}")
    i += 1

sorted_len_booklist = sort_booklist(booklist)

i = 0
print("\n\nBooks after sorting them by length of their title :-\n")
for book in booklist:
    print(f"{i+1}. {book}")
    i += 1
```

```
Books before sorting :-

1. Software Engineering: a practitioners approach
2. Python Programming: Using Problem Solving Approach
3. Introduction to ALgorithms
4. Algorithm Design
5. The Design and Analysis of Computer Algorithms
6. Introduction to modern cryptography
7. Handbook of Applied Cryptography
8. Python Programming - Learn & Practice


Books after sorting them by length of their title :-

1. Algorithm Design
2. Introduction to ALgorithms
3. Handbook of Applied Cryptography
4. Introduction to modern cryptography
5. Python Programming - Learn & Practice
6. The Design and Analysis of Computer Algorithms
7. Software Engineering: a practitioners approach
8. Python Programming: Using Problem Solving Approach
```

1.4 Problem Statement 4 :-

Duplicate Finder: Implement a function to find duplicates that take a list of numbers and returns a list of all numbers that appear more than once.

[18]:
```
def duplicate_finder(num_list):
    duplicates = []
    for num in num_list:
        if num not in duplicates:
            count = 0
            temp = num
            for num in num_list:
                if num == temp:
                    count += 1
            if count > 1:
                duplicates.append(temp)
    return duplicates

# Runner code
num_list = [1, 2, 3, 4, 5, 6, 3, 7, 8, 9, 10, 1, 5]
print(f"Original List :-\n{num_list}")

duplicates = duplicate_finder(num_list)
print(f"\nDuplicates in the List are :-\n{duplicates}")
```

```
Original List :-
[1, 2, 3, 4, 5, 6, 3, 7, 8, 9, 10, 1, 5]

Duplicates in the List are :-
[1, 3, 5]
```

1.5 Problem Statement 5 :-

Data Normalizer: Write a function to normalize data that takes a list of numbers and normalizes them so that the numbers range from 0 to 1 (inclusive).
Use the following formula :-
`XNew = (X - X_min) / (X_max - X_min)`

[19]:
```
def data_normalizer(num_list):
    num_max = max(num_list)
    num_min = min(num_list)
    output = []
    for num in num_list:
        num = (num - num_min) / (num_max - num_min)
        output.append(f"{num:.4f}")
    return output

# Runner code
num_list = [1, 2, 3, 4, 5, 6, 3, 7, 8, 9, 10, 1, 5]
```

```
print(f"Original Data :-\n{num_list}")

normalized_num_list = data_normalizer(num_list)
print(f"\nNormalized Data :-\n{normalized_num_list}")
```

```
Original Data :-
[1, 2, 3, 4, 5, 6, 3, 7, 8, 9, 10, 1, 5]

Normalized Data :-
['0.0000', '0.1111', '0.2222', '0.3333', '0.4444', '0.5556', '0.2222', '0.6667',
'0.7778', '0.8889', '1.0000', '0.0000', '0.4444']
```

1.6 Problem Statement 6 :-

Reversing a Sentence: You are building a text editor application, and one of the features you want to implement is sentence reversal.
Example:
Input: `hello python.`
Output: `olleh .nohtyp`

[20]:
```
def sentence_reversal(sentence):
    words = sentence.split()
    rev_word = ''
    for word in words:
        for i in range(len(word)-1, -1, -1):
            rev_word += word[i]
        rev_word += ' '
    return rev_word.rstrip() # Remove trailing blankspace at end of last word

# Runner code
sentence = input("Enter a sentence: ")
reverse = sentence_reversal(sentence)
print(reverse)
```

```
Enter a sentence:  hello python.

olleh .nohtyp
```