

Python Programming Lab (ACS-552)

List of Programs

1. Program to Implement a Sequential Search	2
2. Program to Create a Calculator	3
3. Program to explore String Functions in Python	5
4. Program to implement Selection Sort in Python	7
5. Program to implement Stack in Python	8
6. Program to implement File Read & Write Operations in Python	12
7. Program to Demonstrate Usages of Basic Regular Expression in Python	13
8. Program to Demonstrate Use of List	16
9. Program to Demonstrate Use of Dictionaries	18

1. Program to Implement a Sequential Search

- **Code**

```
# Program for Implementing Sequential Search in Python
def sequential_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i # Return the index of the target element
    return -1 # Return -1 if the target is not found

# Get the size of the list from the user
size = int(input("Enter the size of the list: "))

# Get the elements of the list from the user
my_list = []
elements_input = input("Enter the elements separated by spaces: ")
elements = elements_input.split()
for element in elements:
    my_list.append(int(element))

# Get the target number from the user
target_number = int(input("Enter the target number: "))

# Perform sequential search
result = sequential_search(my_list, target_number)

if result != -1:
    print(f"{target_number} found at index {result}")
else:
    print("Target not found in the list")
```

- **Output**

```
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\K
\sequentialSearch.py"
Enter the size of the list: 5
Enter the elements separated by spaces: 62 87 17 66 59
Enter the target number: 66
66 found at index 3
```

2. Program to Create a Calculator

- Code

```
# Program to Implement Simple Calculator in Python

# Function to add two numbers
def add(num1, num2):
    return num1 + num2

# Function to subtract two numbers
def subtract(num1, num2):
    return num1 - num2

# Function to multiply two numbers
def multiply(num1, num2):
    return num1 * num2

# Function to divide two numbers
def divide(num1, num2):
    return num1 / num2

# Display calculator menu
print("Calculator Menu:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

# Get user's choice
choice = input("Enter your choice (1-4): ")

# Perform selected operation based on user's choice
if choice in ['1', '2', '3', '4']:
    # Get user input for two numbers
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    if choice == '1':
        # Addition
        result = add(num1, num2)
        print("Result:", result)
    elif choice == '2':
        # Subtraction
        result = subtract(num1, num2)
        print("Result:", result)
    elif choice == '3':
        # Multiplication
        result = multiply(num1, num2)
        print("Result:", result)
    elif choice == '4':
        if num2 != 0:
            # Division
```

```
        result = divide(num1, num2)
        print("Result:", result)
    else:
        # Division by zero error handling
        print("Cannot divide by zero!")
else:
    # Invalid choice error handling
    print("Invalid choice!")
```

- **Output**

```
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\calculator.py"
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter your choice (1-4): 1
Enter the first number: 62
Enter the second number: 87
Result: 149.0
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC>
```

3. Program to explore String Functions in Python

- **Code**

```
# Program to Implement and Demonstrate String Functions in Python

# String functions demonstration

# Length of a string
def string_length(string):
    return len(string)

# Convert a string to uppercase
def uppercase(string):
    return string.upper()

# Convert a string to lowercase
def lowercase(string):
    return string.lower()

# Count the occurrences of a substring in a string
def count_substring(string, substring):
    return string.count(substring)

# Check if a string starts with a specific prefix
def starts_with(string, prefix):
    return string.startswith(prefix)

# Check if a string ends with a specific suffix
def ends_with(string, suffix):
    return string.endswith(suffix)

# Main program
input_string = input("Enter a string: ")

print("Length of the string:", string_length(input_string))
print("Uppercase:", uppercase(input_string))
print("Lowercase:", lowercase(input_string))

substring = input("Enter a substring to count: ")
print("Occurrences of the substring:", count_substring(input_string,
substring))

prefix = input("Enter a prefix to check: ")
if starts_with(input_string, prefix):
    print("The string starts with the prefix.")
else:
    print("The string does not start with the prefix.")

suffix = input("Enter a suffix to check: ")
if ends_with(input_string, suffix):
    print("The string ends with the suffix.")
else:
```

```
print("The string does not end with the suffix.")
```

- **Output**

```
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\KMCLU_ACADEMIC\5 semester\Python\stringFunctions.py"
Enter a string: saysky2
Length of the string: 7
Uppercase: SAYSKY2
Lowercase: saysky2
Enter a substring to count: sky
Occurrences of the substring: 1
Enter a prefix to check: S
The string does not start with the prefix.
Enter a suffix to check: 2
The string ends with the suffix.
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC>
```

4. Program to implement Selection Sort in Python

- **Code**

```
# Implement Selection Sort in Python

def selection_sort(arr):
    # Traverse through all array elements
    for i in range(len(arr)):
        # Find the minimum element in the remaining unsorted array
        min_index = i
        for j in range(i+1, len(arr)):
            if arr[j] < arr[min_index]:
                min_index = j

        # Swap the found minimum element with the first element
        arr[i], arr[min_index] = arr[min_index], arr[i]

# Get the size of the list from the user
size = int(input("Enter the size of the list: "))

# Get the elements of the list from the user
my_list = []
elements_input = input("Enter the elements separated by spaces: ")
elements = elements_input.split()
for element in elements:
    my_list.append(int(element))

print("Original list:", my_list)

selection_sort(my_list)
print("Sorted list:", my_list)
```

- **Output**

```
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\KMCLU_ACADEMIC\5 semester\Python\selectionSort.py"
Enter the size of the list: 5
Enter the elements separated by spaces: 62 87 17 66 59
Original list: [62, 87, 17, 66, 59]
Sorted list: [17, 59, 62, 66, 87]
```

5. Program to implement Stack in Python

- **Code**

```
# Implementation of the Stack in Python

# Code to implement the stack data structure
class Stack:
    def __init__(self):
        self.stack = []

    def is_empty(self):
        return len(self.stack) == 0

    def push(self, item):
        self.stack.append(item)

    def pop(self):
        if self.is_empty():
            return None
        return self.stack.pop()

    def peek(self):
        if self.is_empty():
            return None
        return self.stack[-1]

    def size(self):
        return len(self.stack)

    def display(self):
        print("Stack content:", self.stack)

# Code to Perform Stack Operations
stack = Stack()

while True:
    print("Menu:")
    print("1. Push an item")
    print("2. Pop an item")
    print("3. Check if the stack is empty")
    print("4. Peek at the top item")
    print("5. Get the size of the stack")
    print("6. Print the entire content of the stack")
    print("7. Exit")

    choice = input("Enter your choice (1-7): ")

    if choice == "1":
        item = input("Enter an item to push onto the stack: ")
        stack.push(item)
```

```
        print("Item pushed onto the stack:", item)
elif choice == "2":
    item = stack.pop()
    if item is None:
        print("Stack is empty. Cannot pop an item.")
    else:
        print("Popped item:", item)
elif choice == "3":
    if stack.is_empty():
        print("Stack is empty.")
    else:
        print("Stack is not empty.")
elif choice == "4":
    item = stack.peek()
    if item is None:
        print("Stack is empty. No item to peek.")
    else:
        print("Top item:", item)
elif choice == "5":
    if stack.is_empty():
        print("Stack is empty.")
    else:
        print("Size of the stack:", stack.size())
elif choice == "6":
    if stack.is_empty():
        print("Stack is empty.")
    else:
        stack.display()
elif choice == "7":
    print("Exiting the program.")
    break
else:
    print("Invalid choice. Please try again.")

print()
```


- Output

```
\stack.py"
Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
Enter your choice (1-7): 1
Enter an item to push onto the stack: 62
Item pushed onto the stack: 62

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
Enter your choice (1-7): 1
Enter an item to push onto the stack: 87
Item pushed onto the stack: 87

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
```

```
Enter your choice (1-7): 3
Stack is not empty.

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
Enter your choice (1-7): 4
Top item: 87

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
Enter your choice (1-7): 17
Invalid choice. Please try again.

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
```

```
Enter your choice (1-7): 6
Stack content: ['62', '87']

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
Enter your choice (1-7): 5
Size of the stack: 2

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
Enter your choice (1-7): 4
Top item: 87

Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
```

```
Menu:
1. Push an item
2. Pop an item
3. Check if the stack is empty
4. Peek at the top item
5. Get the size of the stack
6. Print the entire content of the stack
7. Exit
Enter your choice (1-7): 7
Exiting the program.
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC>
```

6. Program to implement File Read & Write Operations in Python

- **Code**

```
# Perform read and write operations on files in Python

import os

# Write data to a file
def write_to_file(file_name, data):
    try:
        with open(file_name, 'w') as file:
            file.write(data)
        print("Data written to the file successfully.")
    except IOError:
        print("Error: Failed to write data to the file.")

# Read data from a file
def read_from_file(file_name):
    try:
        with open(file_name, 'r') as file:
            data = file.read()
            print("Data read from the file:", data)
    except IOError:
        print("Error: Failed to read data from the file.")

# Check if the file exists
def check_file_exists(file_name):
    return os.path.isfile(file_name)

# Get file name from the user
file_name = input("Enter the file name: ")

# Check if the file exists
if check_file_exists(file_name):
    # File exists, prompt for operation choice
    while True:
        print("Choose an operation:")
        print("1. Read from the file")
        print("2. Write to the file")
        print("3. Exit")

        choice = input("Enter your choice (1, 2, or 3): ")

        if choice == "1":
            read_from_file(file_name)
        elif choice == "2":
            data_to_write = input("Enter the data to write to the file: ")
            write_to_file(file_name, data_to_write)
        elif choice == "3":
            print("Exiting the program.")
            break
        else:
```

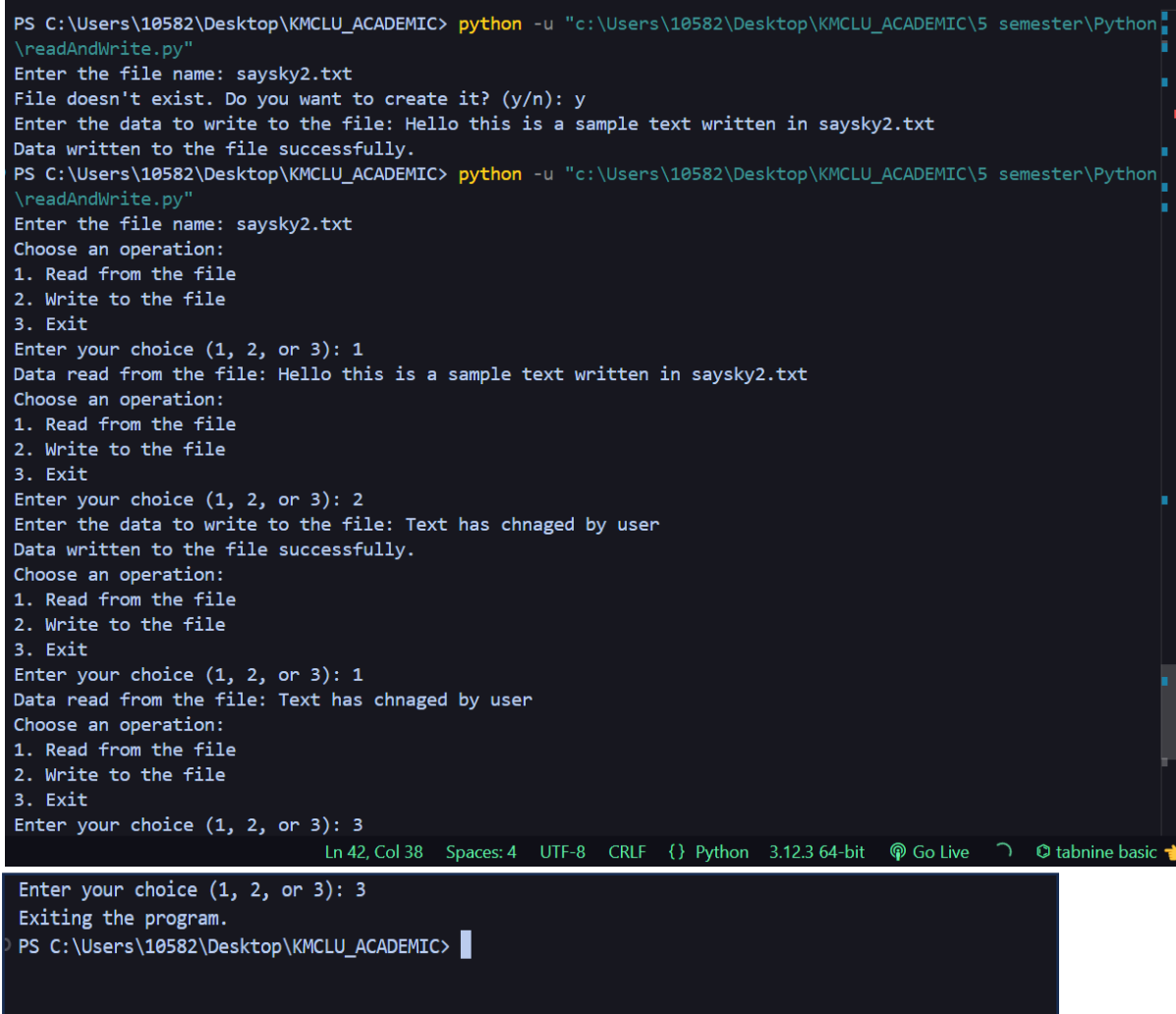
```

        print("Invalid choice. Please try again.")
    else:
        # File doesn't exist, ask if the user wants to create it
        create_file = input("File doesn't exist. Do you want to create it? (y/n): ")

        if create_file.lower() == "y":
            data_to_write = input("Enter the data to write to the file: ")
            write_to_file(file_name, data_to_write)
        else:
            print("Exiting the program.")

```

- **Output**



```

PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\KMCLU_ACADEMIC\5 semester\Python\readAndWrite.py"
Enter the file name: saysky2.txt
File doesn't exist. Do you want to create it? (y/n): y
Enter the data to write to the file: Hello this is a sample text written in saysky2.txt
Data written to the file successfully.
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\KMCLU_ACADEMIC\5 semester\Python\readAndWrite.py"
Enter the file name: saysky2.txt
Choose an operation:
1. Read from the file
2. Write to the file
3. Exit
Enter your choice (1, 2, or 3): 1
Data read from the file: Hello this is a sample text written in saysky2.txt
Choose an operation:
1. Read from the file
2. Write to the file
3. Exit
Enter your choice (1, 2, or 3): 2
Enter the data to write to the file: Text has chnaged by user
Data written to the file successfully.
Choose an operation:
1. Read from the file
2. Write to the file
3. Exit
Enter your choice (1, 2, or 3): 1
Data read from the file: Text has chnaged by user
Choose an operation:
1. Read from the file
2. Write to the file
3. Exit
Enter your choice (1, 2, or 3): 3
Exiting the program.
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC>

```

7. Program to Demonstrate Usages of Basic Regular Expression in Python

- **Code**

```
# Demonstrate Usage of some basic Regular Expressions in Python

import re

# Example string
text = "The quick brown fox jumps over the lazy dog."

# Regular expression patterns
patterns = [
    r"fox",                                # Matches "fox"
    r"\b\w{5}\b",                          # Matches words with exactly 5 characters
    r"\b[a-z]{3}\b",                       # Matches lowercase words with exactly 3
characters
    r"[A-Z][a-z]+",                        # Matches capitalized words
    r"\b[a-zA-Z]+\b",                      # Matches any word
    r"\b[a-zA-Z]{4,}\b",                   # Matches words with 4 or more characters
    r"\b[a-zA-Z]+s\b",                     # Matches words ending with 's'
    r"\b[a-zA-Z]{3}s\b",                   # Matches words with 3 characters ending
with 's'
    r"\b[a-zA-Z]{4,}s\b",                   # Matches words with 4 or more characters
ending with 's']

# Perform regex matching and print the results
for pattern in patterns:
    matches = re.findall(pattern, text)
    print(f"Pattern: {pattern}")
    print("Matches:", matches)
    print()
```

- **Output**

```
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\KMCLU_ACADEMIC\5 semester\Python\useBasicRegularExp.py"
Pattern: saysky2
Matches: ['saysky2']

Pattern: \b\w{6}\b
Matches: ['author', 'Python']

Pattern: \b[a-z]{3}\b
Matches: ['the', 'the']

Pattern: [A-Z][a-z]+
Matches: ['The', 'Python']

Pattern: \b[a-zA-Z]+\b
Matches: ['The', 'author', 'of', 'the', 'program', 'is', 'the', 'codes', 'is', 'wrtiten', 'in', 'Python']

Pattern: \b[a-zA-Z]{4,}\b
Matches: ['author', 'program', 'codes', 'wrtiten', 'Python']

Pattern: \b[a-zA-Z]+\s\b
Matches: ['is', 'codes', 'is']

Pattern: \b[a-zA-Z]{3}s\b
Matches: []

Pattern: \b[a-zA-Z]{4,}s\b
Matches: ['codes']

PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> 
```

8. Program to Demonstrate Use of List

- **Code**

```
# Demonstrate the usage of the List in Python

# Create an empty list
my_list = []

# Add elements to the list
my_list.append("apple")
my_list.append("banana")
my_list.append("orange")

# Access elements in the list
print("First element:", my_list[0])
print("Second element:", my_list[1])
print("Last element:", my_list[-1])

# Modify elements in the list
my_list[1] = "grape"
print("Modified list:", my_list)

# Remove elements from the list
removed_item = my_list.pop(0)
print("Removed item:", removed_item)
print("Updated list:", my_list)

# Get the length of the list
list_length = len(my_list)
print("Length of the list:", list_length)

# Check if an element exists in the list
if "orange" in my_list:
    print("The element 'orange' exists in the list.")
else:
    print("The element 'orange' does not exist in the list.")

# Iterate over the elements in the list
print("Elements in the list:")
for item in my_list:
    print(item)

# Clear the list
my_list.clear()
print("Cleared list:", my_list)
```

- **Output**

```
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\KMCLU_ACADEMIC\5 semester\Python\useOfList.py"
First element: apple
Second element: banana
Last element: orange
Modified list: ['apple', 'grape', 'orange']
Removed item: apple
Updated list: ['grape', 'orange']
Length of the list: 2
The element 'orange' exists in the list.
Elements in the list:
grape
orange
Cleared list: []
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> 
```

9. Program to Demonstrate Use of Dictionaries

- **Code**

```
# Demonstrate the usage of the Dictionary in Python

# Create an empty dictionary
my_dict = {}

# Add key-value pairs to the dictionary
my_dict["name"] = "John Doe"
my_dict["age"] = 25
my_dict["city"] = "New York"

# Access values in the dictionary
print("Name:", my_dict["name"])
print("Age:", my_dict["age"])
print("City:", my_dict["city"])

# Modify values in the dictionary
my_dict["age"] = 30
print("Modified dictionary:", my_dict)

# Remove a key-value pair from the dictionary
removed_value = my_dict.pop("city")
print("Removed value:", removed_value)
print("Updated dictionary:", my_dict)

# Check if a key exists in the dictionary
if "name" in my_dict:
    print("The key 'name' exists in the dictionary.")
else:
    print("The key 'name' does not exist in the dictionary.")

# Get the keys and values in the dictionary
keys = my_dict.keys()
values = my_dict.values()
print("Keys:", keys)
print("Values:", values)

# Iterate over key-value pairs in the dictionary
print("Key-Value Pairs:")
for key, value in my_dict.items():
    print(key, ":", value)

# Clear the dictionary
my_dict.clear()
print("Cleared dictionary:", my_dict)
```


- **Output**

```
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> python -u "c:\Users\10582\Desktop\KMCLU_ACADEMIC\5 semester\Python\useOfDictionaries.py"
Name: saysky2
Age: 21
City: Ballia
Modified dictionary: {'name': 'saysky2', 'age': 20, 'city': 'Ballia'}
Removed value: Ballia
Updated dictionary: {'name': 'saysky2', 'age': 20}
The key 'name' exists in the dictionary.
Keys: dict_keys(['name', 'age'])
Values: dict_values(['saysky2', 20])
Key-Value Pairs:
name : saysky2
age : 20
Cleared dictionary: {}
PS C:\Users\10582\Desktop\KMCLU_ACADEMIC> 
```