Sri Siddhartha Institute of Technology, Tumakuru
Department of Computer Science and Engineering

CS4LB3 : Python Programming Laboratory

Cycle-2 Experiments

5 a) Write a Python program that reads two integer lists from the user and checks whether list2 is a sub list of list1 or not.

Program:

```
def is_sub_list(list1, list2):
    if list2 == [] or list2 == list1:
        sub_list = True
    elif len(list2) > len(list1):
        sub_list = False
    else:
        sub_list=False
        for i in range(len(list1)-len(list2)+1):
            if list1[i] == list2[0]:
                j = 1
                while (j < len(list2)) and i+j < len(list1) and list1[i+j] == list2[j]:
                    j += 1
                if j == len(list2):
                    sub_list = True
                    break
    return sub_list

list1 = [int(item) for item in input("Enter a list of integers: ").split()]

list2 = [int(item) for item in input("Enter a sub list of integers to be searched in the main list: ").split()]

print("Main List:",list1)
print("Sub list to be searched:",list2)

if is_sub_list(list1, list2):
    print(list2, "is a sub list of", list1)
else:
    print(list2, "is not a sub list of", list1)
```

Output 1:

```
Enter a list of integers: 10 20 30 40 50

Enter a sub list of integers to be searched in the main list: 20 30 40

Main List: [10, 20, 30, 40, 50]

Sub list to be searched: [20, 30, 40]

[20, 30, 40] is a sub list of [10, 20, 30, 40, 50]
```

Output 2:

Enter a list of integers: 10 20 30 40

Enter a sub list of integers to be searched in the main list: 40 70

Main List: [10, 20, 30, 40]

Sub list to be searched : [40, 70]

[40, 70] is not a sub list of [10, 20, 30, 40]

5 b) Write a Python program that groups a list of key-value pairs into a dictionary of lists

Program:

```
def grouping_dictionary(list1):
    result = {}
    for key, value in list1:
        result.setdefault(key, []).append(value)
    return result

colors = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
print("List of key-value pairs:")
print(colors)
print("\nDictionary of lists:")
print(grouping_dictionary(colors))
```

Output :

List of key-value pairs:

[('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]

Dictionary of lists:

{'yellow': [1, 3], 'blue': [2, 4], 'red': [1]}

6 a) Write a Python program that generates and prints a list of prime numbers less than the given number.

Program:

```
def is_prime(n):
    """
    This function returns True if n is a prime number else returns False
    """
    flag=True                       # Assume that n is a prime number
    for f in range(2,n//2+1):       # The possible factors for n lies between 2 and n/2
        if n%f==0:                  # n has a factor, so it is not a prime number
            flag=False
            break
    return flag
```

```
def list_of_prime_numbers(num):
    """
    This function generates and returns a list of prime numbers less than the given number
    """
    prime_list=[]
    for n in range(2,num):
        if is_prime(n):
            prime_list.append(n)
    return prime_list


num=int(input("Enter a number to generate a list of prime numbers less than the number:"))
print(list_of_prime_numbers(num))
```

Output :

Enter a number to generate a list of prime numbers less than the number:20
[2, 3, 5, 7, 11, 13, 17, 19]

6 b) Write a Python program to display the lost elements in a duplicate list using set difference operation.

Program:

```
def lost_elements(original_list,duplicate_list):

        # convert lists into sets
        original_set = set(original_list)
        duplicate_set=set(duplicate_list)

        # Use the set difference operation to find the lost elements in the duplicate list
        return list(original_set-duplicate_set)

list1 = [1, 4, 5, 7, 9]        # Original list of elements
print("Original list:", list1)
list2= [1,5,7]                 # Duplicate list of elements with some elements lost
print("Duplicate list:", list2)
print("Lost elements in the duplicate list:",lost_elements(list1,list2))
```

Output :

Original list: [1, 4, 5, 7, 9]

Duplicate list: [1, 5, 7]

Lost elements in the duplicate list: [9, 4]

7 a) Write a Python program to copy the contents of a source file to a destination file by omitting comment lines (lines that start with #).

Program:

```
def filter_comment_lines(src_file, dest_file):
    """
    This function accepts the names of source file and destination file as parameters,
    reads the source file line by line and writes only non comment lines from it
    (lines that donot start with #) to the destination file.
    """
    with open(src_file, "r") as infile, open(dest_file, "w") as outfile:
        for line in infile:
            if not line.startswith('#'):
                outfile.write(line)
    print("Filtering of comment lines succesful, check the contents of destination file")

src=input("Enter the name of the source file:")
dest=input("Enter the name of the destination file:")
filter_comment_lines(src,dest)
```

Procedure to create the source file:

- Create a new text file in notepad
- Type the following python code

```
#Program to find the area of a circle
#Get the radius from the user
r=float(input("Enter the radius")
#Compute area of the circle
area=3.142*r*r
#Print area of the circle
print("Area of the circle is:",area)
```

- Save the file as source.txt in the same directory (working directory)

Output:

```
Enter the name of the source file:source.txt
Enter the name of the destination file:destination.txt
Filtering of comment lines successful, check the contents of destination file
```

Procedure to check the contents of the destination file:

- A destination file with the name destination.txt will be created in the same directory(working directory)
- Open the destination.txt file to see its contents

```
r=float(input("Enter the radius")
area=3.142*r*r
print("Area of the circle is:",area)
```

7 b) Write a Python program to draw a Polygon of given number of sides. The program should raise appropriate exception when the number of sides is negative or zero.

Program:

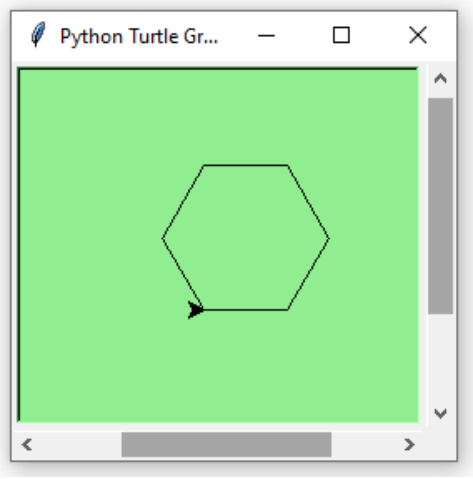
```
import turtle
import time
def draw_poly(sides):

    try:
      win = turtle.Screen()        # Creates a window
      win.bgcolor("lightgreen")
      tess = turtle.Turtle()       # Creates a turtle
      if sides<0:
        raise ValueError           # Raises ValueError if number of sides is negative
      else:
        angle = 360 / sides
        for i in range(sides):     # Draws a polygon with given number of sides
          tess.forward(50)
          tess.left(angle)
          time.sleep(1)
    except ValueError:
        print("Sorry, number of sides cannot be negative")
    except ZeroDivisionError:
        print("Sorry, number of sides cannot be zero, it leads to zero division error")
    finally:
      win.mainloop()               # Waits till the user closes the window


n=int(input("Enter the number of sides to draw a polygon:"))
draw_poly(n)
```

Output :

Enter the number of sides to draw a polygon:6



Error case 1 (Error manually raised and handled by the programmer):

Enter the number of sides to draw a polygon:-4
Sorry, number of sides cannot be negative

<u>Error case 2</u> (Error automatically raised by the runtime environment, but manually handled by the programmer):

Enter the number of sides to draw a polygon:0
Sorry, number of sides cannot be zero, it leads to zero division error

<u>Error case 3</u> (Error automatically raised and handled by the runtime environment):

Enter the number of sides to draw a polygon:abc

Traceback (most recent call last):
  File "C:\Users\SHESHADRI\Desktop\temp.py", line 25, in <module>
    n=int(input("Enter the number of sides to draw a polygon:"))
ValueError: invalid literal for int() with base 10: 'abc'