9	**a) Write a Python program to check if a specified element presents in a tuple of tuples.** **Original list:** **(('Red' ,'White' , 'Blue'),('Green', 'Pink' , 'Purple'), ('Orange', 'Yellow', 'Lime'))** ❖ **Check if White present in said tuple of tuples** ❖ **Check if Olive present in said tuple of tuples** **b) Write a Python program to remove an empty tuple(s) from a list of tuples.** **Sample data: [(), (), (''), ('a', 'b'), ('a', 'b'. 'c'), ('d')]** **Expected Output:[(''),('a', 'b'),('a', 'b', 'c'), 'd']**		
10	**Write a Program in Python to Find the Differences Between Two Lists using sets.**		
11.	**(a)Write a python program to remove duplicate values across dictionary values.** **Input: test_dict={"Manjeet":[1,1,1],"Akash":[1,1,1]}** **Output: {"Manjeet":[], "Akash":[1,1,1]}** **(b) Write a Python program to Count the frequencies in a list using dictionary in Python.** **Input: [1, 1, 1, 5, 5, 3, 1, 3, 3, 1,4, 4, 4, 2, 2, 2, 2]** **Output:** **1: 5** **2: 4** **3: 3** **4: 3** **5: 2** **Explanation: Here 1 occurs 5 times, 2 occurs 4 times and so on...**		
12.	**(a) Write a Python Program to Capitalize First Letter of Each Word in a File.** **(b) Write a Python Program to Print the Contents of File in Reverse Order.**		
13.	**WAP to catch an exception and handle it using try and except code blocks**		
14.	**Write a Python Program to Append, Delete and Display Elements of a List using Classes.**		
15.	**Write a Python Program to find the Area and Perimeter of the Circle using class.**		
16	**Create an interactive application using Python's Tkinter library for graphics programming.**		

Program 9: a) Write a Python program to check if a specified element presents in a tuple of
tuples.
Original list:
((‘Red’ ,’White’ , ‘Blue’),(‘Green’, ’Pink’ , ‘Purple’), (‘Orange’, ‘Yellow’, ‘Lime’))
Check if White present in said tuple of tuples!
True
Check if Olive present in said tuple of tuples!
False

Solution:

```
original_tuples = (('Red', 'White', 'Blue'), ('Green', 'Pink', 'Purple'), ('Orange', 'Yellow', 'Lime'))


elements_to_check = ['White', 'Olive']


for element in elements_to_check:
    found = False
    for sub_tuple in original_tuples:
        if element in sub_tuple:
            found = True
            break
    print(f"Check if {element} present in said tuple of tuples: {found}")
```

Output:

```
Check if White present in said tuple of tuples: True
Check if Olive present in said tuple of tuples: False
```

Program 9: b) Write a Python program to remove an empty tuple(s) from a list of tuples.
Sample data: [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]
Expected output: [('',), ('a', 'b'), ('a', 'b', 'c'), 'd']

Solution:

```
sample_data = [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]

# Remove empty tuples from the list using list comprehension
filtered_data = [tup for tup in sample_data if tup and any(tup)]
# Explanation: Keep tuples that are not empty and have at least one non-empty element

# Print the expected output
print("Expected output:", filtered_data)
```

Output:

```
Expected output : [('',), ('a', 'b'), ('a', 'b', 'c'), 'd']
```

Program 10: Write a Program in Python to Find the Differences Between Two Lists Using Sets.

Solution:

```
list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]

# Convert lists to sets
set1 = set(list1)
set2 = set(list2)

# Find the differences between the sets
difference1 = set1 - set2  # Elements in list1 but not in list2
difference2 = set2 - set1  # Elements in list2 but not in list1

# Convert the differences back to lists (if needed)
difference_list1 = list(difference1)
difference_list2 = list(difference2)

# Print the differences
print("Elements in list1 but not in list2:", difference_list1)
print("Elements in list2 but not in list1:", difference_list2)
```

Output:

```
Elements in list1 but not in list2: [1, 2]
Elements in list2 but not in list1: [6, 7]
```

Program 11(a): a) Write a Python program Remove duplicate values across Dictionary Values.

Input : test_dict = {'Manjeet': [1], 'Akash': [1, 8, 9]}
Output : {'Manjeet': [], 'Akash': [8, 9]}
Input : test_dict = {'Manjeet': [1, 1, 1], 'Akash': [1, 1, 1]}
Output : {'Manjeet': [], 'Akash': []}

Solution:

```
def remove_duplicates_from_dict_values(input_dict):
    result_dict = {}

    for key, value in input_dict.items():
        result_dict[key] = list(set(value))

    return result_dict

# Example usage:
test_dict1 = {'Manjeet': [1], 'Akash': [1, 8, 9]}
result1 = remove_duplicates_from_dict_values(test_dict1)
print(result1)

test_dict2 = {'Manjeet': [1, 1, 1], 'Akash': [1, 1, 1]}
result2 = remove_duplicates_from_dict_values(test_dict2)
print(result2)
```

Output:

```
{'Manjeet': [], 'Akash': [8, 9]}

{'Manjeet': [], 'Akash': []}
```

Program 11(b): b) Write a Python program to Count the frequencies in a list using dictionary in Python.

Input : [1, 1, 1, 5, 5, 3, 1, 3, 3, 1,4, 4, 4, 2, 2, 2, 2]

Output :

1 : 5
2 : 4
3 : 3
4 : 3
5 : 2

Solution:

```
def count_frequencies(input_list):
    frequency_dict = {}

    for element in input_list:
        if element in frequency_dict:
            frequency_dict[element] += 1
        else:
            frequency_dict[element] = 1

    return frequency_dict
input_list = [1, 1, 1, 5, 5, 3, 1, 3, 3, 1, 4, 4, 4, 2, 2, 2, 2]
result = count_frequencies(input_list)

for key, value in result.items():
    print(f"{key} : {value}")
```

Output:

1: 5

2: 4

3: 3

4: 3

5: 2

Program 12(a): Write a Python Program to Capitalize First Letter of Each Word in a File.

Solution:

```
def capitalize_first_letter(file_path):
    try:

        with open("abc.py", 'r') as file:

            content = file.read()


            modified_content = ' '.join(word.capitalize() for word in content.split())


        with open("abc.py", 'w') as file:
            file.write(modified_content)

        print(f"Capitalization completed for file: {file_path}")

    except FileNotFoundError:
        print(f"File not found: {file_path}")
    except Exception as e:
        print(f"An error occurred: {e}")


file_path = 'abc.py'
capitalize_first_letter(file_path)
```

```
Capitalization completed for file: abc.py
```

Output:

Program 12 (b): Write a Python Program to Print the Contents of File in Reverse Order.

Solution:

```
with open("abc.py", "r") as myfile:
    my_data = myfile.read()


rev_data = my_data[::-1]


print("Reversed data = ",rev_data)
```

Output:

```
Reversed data =  .stcejbO gnitaerC roF "tnirpeulb" A rO ,rotcurtsnoC tcejbO nA ekiL sI ssalC a.sdohteM dnA seitreporP stI ht
iW ,tcejbO nA sI nohtyP nI gnihtyrevE tsomla.egaugnaL gnimmargorP detneirO tcejbO nA sI nohtyP
```

Program 13: Write a program to catch an exception and handle it using try and except code blocks.

Solution:

```
def divide_numbers(num1, num2):
    try:
        result = num1 / num2
        print(f"The result of {num1} / {num2} is: {result}")

    except ZeroDivisionError:
        print("Error: Cannot divide by zero!")

    except Exception as e:
        print(f"An error occurred: {e}")

try:
    divide_numbers(10, 2)
    divide_numbers(5, 0)
    divide_numbers("a", 2)

except Exception as e:
    print(f"Outer exception: {e}")
```

Output:

```
The result of 10 / 2 is: 5.0
Error: Cannot divide by zero!
An error occurred: unsupported operand type(s) for /: 'str' and 'int'
```

Program 14: Write a Python Program to Append, Delete and Display Elements of a List using classes.

Solution:

```
class ListOperations:
    def __init__(self):
        self.my_list = []
    def append_element(self, element):
        self.my_list.append(element)
        print(f"Element {element} appended to the list.")
    def delete_element(self, element):
        if element in self.my_list:
            self.my_list.remove(element)
            print(f"Element {element} deleted from the list.")
        else:
            print(f"Element {element} not found in the list.")
    def display_elements(self):
        if not self.my_list:
            print("The list is empty.")
        else:
            print("Elements in the list:")
            for element in self.my_list:
                print(element)
list_operations = ListOperations()

list_operations.display_elements()

list_operations.append_element(10)
list_operations.append_element(20)
list_operations.append_element(30)

list_operations.display_elements()

list_operations.delete_element(20)
list_operations.delete_element(40)

list_operations.display_elements()
```

Output:

```
The list is empty.
Element 10 appended to the list.
Element 20 appended to the list.
Element 30 appended to the list.
Elements in the list:
10
20
30
Element 20 deleted from the list.
Element 40 not found in the list.
Elements in the list:
10
30
```

Program 15: Write a Python Program to Find the Area and Perimeter of the Circle using class.

Solution:

```
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        area = math.pi * self.radius**2
        return area

    def calculate_perimeter(self):
        perimeter = 2 * math.pi * self.radius
        return perimeter
radius = float(input("Enter the radius of the circle: "))

circle_instance = Circle(radius)

area = circle_instance.calculate_area()
perimeter = circle_instance.calculate_perimeter()

print(f"Area of the circle: {area:.2f}")
print(f"Perimeter of the circle: {perimeter:.2f}")
```

Output:

```
Enter the radius of the circle: 56
Area of the circle: 9852.03
Perimeter of the circle: 351.86
```

Program 16 : Create an interactive application using Pythons Tkinter library for graphics programming

Solution:

```
import tkinter as tk
from tkinter import ttk
from random import randint

class InteractiveApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Interactive App")

        self.label = ttk.Label(root, text="Hello, Tkinter!", font=("Helvetica", 16))
        self.label.pack(pady=20)

        self.button = ttk.Button(root, text="Change Color", command=self.change_color)
        self.button.pack()

    def change_color(self):
        # Change the label's text color to a random color
        color = "#{:06x}".format(randint(0, 0xFFFFFF))
        self.label.config(foreground=color)

def main():
    root = tk.Tk()
    app = InteractiveApp(root)
    root.mainloop()

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

Output: