

# COMPUTER ASSIGNMENT

Savar Manchanda

24BCH007

DIV-1( F1)

PYTHON PROGRAMMING

Branch :- ECE

## Assignment 1

```
>>> A=6
```

```
>>> B=4
```

1)

```
>>> A+B
```

10

2)

```
>>> A-B
```

2

3)

```
>>> A*B
```

24

4)

```
>>> A/B
```

1.5

5)

```
>>> A**B
```

```
1296
```

```
>>> A//B
```

```
1
```

```
>>> a=7
```

```
>>> b=5
```

```
>>> addition=a+b
```

```
>>> addition
```

```
12
```

```
>>> subtraction=a-b
```

```
>>> subtraction
```

```
2
```

```
>>> multiplication=a*b
```

```
>>> multiplication
```

```
35
```

```
>>> division=a/b
```

```
>>> division
```

```
1.4
```

6)

```
>>> hours=int(input("enter the hours:"))
```

```
enter the hours:7
```

```
>>> minutes=hours*60
```

```
>>> minutes
```

```
420
```

7)

```
>>> minutes=int(input("enter the minutes:"))
enter the minutes:120
>>> hours=minutes/60
>>> hours
2.0
```

```
8)
>>> dollar=int(input("enter the dollar:"))
enter the dollar:10
>>> rupees=dollar*85
>>> rupees
850
```

```
9)
>>> rupees=int(input("enter the rupees: "))
enter the rupees: 510
>>> dollar=rupees/85
>>> dollar
6.0
```

```
10)
>>> dollar=int(input("enter the dollar: "))
enter the dollar: 100
>>> pound=dollar*0.82
>>> pound
82.0
```

```
11)
>>> kgs=int(input("enter the kgs: "))
```

enter the kgs: 7

```
>>> grams=kgs*1000
```

```
>>> grams
```

7000

12)

```
>>> grams=int(input("enter the grams: "))
```

enter the grams: 3000

```
>>> kgs=grams/1000
```

```
>>> kgs
```

3.0

13)

```
>>> byte=int(input("enter the byte: "))
```

enter the byte: 100000

```
>>> KB=byte/1000
```

```
>>> KB
```

100.0

```
>>> MB=byte/1000000
```

```
>>> MB
```

0.1

```
>>> GB=byte/1000000000
```

```
>>> GB
```

0.0001

14)

```
>>> celcius=int(input("enter the calcius: "))
```

enter the calcius: 79

```
>>> fahrenheit=(9/5*celcius)+32
```

```
>>> fahrenheit
174.20000000000002
```

15)

```
>>> fahrenheit=int(input("enter the fahrenheit: "))
enter the fahrenheit: 264
>>> celcius=5/9*(fahrenheit-32)
>>> celcius
128.88888888888888
```

16)

```
>>> p=float(input("enter the principal: "))
enter the principal: 2000
>>> r=float(input("enter the rate: "))
enter the rate: 2
>>> n=float(input("enter the time in years: "))
enter the time in years: 1
>>> interest=(p*r*n)/100
>>> interest
40.0
```

17)

```
>>> side=int(input("enter side: "))
enter side: 4
>>> area=side*side
>>> area
16
>>> perimeter=4*side
>>> perimeter
```

```
>>> area=l*b
```

```
>>> area
```

```
40
```

```
>>> perimeter=2*(l+b)
```

```
>>> perimeter
```

```
26
```

```
19)
```

```
>>> R=int(input("enter Radius: "))
```

```
enter Radius: 8
```

```
>>> area=22/7*R*R
```

```
>>> area
```

```
201.14285714285714
```

```
20)
```

```
>>> b=int(input("enter base: "))
```

```
enter base: 2
```

```
>>> h=int(input("enter height: "))
```

```
enter height: 5
```

```
>>> area=(b*h)/2
```

```
>>> area
```

```
5.0
```

```
21)
```

```
>>> GrossSalay=int(input("enter gross Salay: "))
```

```
enter gross Salay: 30000
```

```
>>> allowance=GrossSalay*0.10
```

```
>>> deduction=GrossSalay*0.03
```

```
>>> NatSalay=GrossSalay+allowance-deduction
```

```
>>> NatSalay
```

```
32100.0
```

```
22)
```

```
>>> GrossSalay=int(input("enter gross Salay: "))
```

```
enter gross Salay: 50000
```

```
>>> discount=GrossSalay*0.10
```

```
>>> NatSalay=GrossSalay-discount
```

```
>>> NatSalay
```

```
45000.0
```

```
>>> average=total/3
```

```
>>> average
```

```
84.0
```

```
24)
```

```
>>> a=input("enter first value: ")
```

```
enter first value: 5
```

```
>>> b=input("enter second value: ")
```

```
enter second value: 8
```

```
>>> temp=a
```

```
>>> a=b
```

```
>>> b=temp
```

```
>>> a
```

```
'8'
```

```
>>> b
```

```
'5'
```

## Assignment 2

1# Print largest and smallest values out of two using if else statement.

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
if num1 > num2:
    largest = num1
    smallest = num2
else:
    largest = num2
    smallest = num1
print("The largest number is: {largest}")
print("The smallest number is: {smallest}")
```

output:-

Enter the first number: 43

Enter the second number: 42

The largest number is: 43.0

The smallest number is: 42.0

2# Print largest and smallest values out of three using if else statement.

```
a = float(input("Enter the first number: "))
b = float(input("Enter the second number: "))
c = float(input("Enter the third number: "))
if a >= b and a >= c:
    largest = a
elif b >= a and b >= c:
    largest = b
else:
    largest = c
if a <= b and a <= c:
    smallest = a
elif b <= a and b <= c:
```



```
    smallest = b
else:
    smallest = c
print("Largest number: {largest}")
print("Smallest number: {smallest}")
```

Output:-

Enter the first number: 34

Enter the second number: 35

Enter the third number: 37

Largest number: 37.0

Smallest number: 34.0

3# Check whether a given number is odd or even using if else statement.

```
number = int(input("Enter a number: "))
```

```
if number % 2 == 0:
```

```
    print(f"{number} is Even.")
```

```
else:
```

```
    print(f"{number} is Odd.")
```

Output:-

Enter a number: 75

75 is Odd.

4# Check whether a given number is divisible by 10 or not.

```
number = int(input("Enter a number: "))
```

```
if number % 10 == 0:
```

```
    print("The number {number} is divisible by 10.")
```

```
else:
```

```
    print("The number {number} is not divisible by 10.")
```

Output:-

Enter a number: 6

The number 6 is not divisible by 10.

5# Accept age of a person. If age is less than 18, print minor otherwise Major.

```
age = int(input("Enter your age: "))
```

```
if age < 18:
```

```
    print("Minor")
```

```
else:
```

```
    digit count = len(number)
```

```
print("The number of digits in the number is: {digit count}")
```

Output:-

Enter a number: 36

The number of digits in the number is: 2

7# Accept a year value from the user. Check whether it is a leap year or not.

```
year = int(input("Enter a year: "))
```

```
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
```

```
    print(f"{year} is a leap year.")
```

```
else:
```

```
    print(f"{year} is not a leap year.")
```

Output:-

Enter a year: 2078

2078 is not a leap year.

8# Check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard a triangle is valid if the sum of all the three angles is equal to 180 degrees.

```
angle1 = float(input("Enter the first angle of the triangle: "))
```

```
angle2 = float(input("Enter the second angle of the triangle: "))
```

```
angle3 = float(input("Enter the third angle of the triangle: "))
```

```
if angle1 + angle2 + angle3 == 180:
```

```
    print("The triangle is valid.")
```

```
else:
```

```
    print("The triangle is not valid.")
```

Output:-

Enter the first angle of the triangle: 75

Enter the second angle of the triangle: 90

Enter the third angle of the triangle: 15

The triangle is valid.

9# Print absolute value of a given number using if else statement.

```
number = float(input("Enter a number: "))
```

```
if number < 0:
```

```
    absolute value = -number
```

```
else:
```

```
    absolute value = number
```

```
print( "The absolute value of {number} is {absolute value}")
```

Output:-

Enter a number: 65

The absolute value of 65.0 is 65.0

10# Given the length and breadth of a rectangle, write a program to find whether the area of the rectangle is greater than its perimeter using if else statement.

```
length = float(input("Enter the length of the rectangle: "))
```

```
breadth = float(input("Enter the breadth of the rectangle: "))
```

```
area = length * breadth
```

```
perimeter = 2 * (length + breadth)
```

```
if area > perimeter:
```

```
    print("The area of the rectangle ({area}) is greater than its perimeter ({perimeter}).")
```

```
else:
```

```
    print("The area of the rectangle ({area}) is not greater than its perimeter ({perimeter}).")
```

Output:-

Enter the length of the rectangle: 30

Enter the breadth of the rectangle: 15

The area of the rectangle (450.0) is greater than its perimeter (90.0).

11# Given three points (x1,y1), (x2,y2) and (x3,y3), check if all the three points fall on one straight line. using if else statement.

```
x1, y1 = float(input("Enter x1, y1: ")), float(input("Enter y1: "))
```

```
x2, y2 = float(input("Enter x2, y2: ")), float(input("Enter y2: "))
```

```
x3, y3 = float(input("Enter x3, y3: ")), float(input("Enter y3: "))
```

```
if (y2 - y1) * (x3 - x2) == (y3 - y2) * (x2 - x1):
```

```
    print("The points lie on a straight line.")
```

```
else:
```

```
    print("The points do not lie on a straight line.")
```

Output:-

Enter x1, y1: 10

Enter y1: 12

Enter x2, y2: 15

Enter y2: 9

Enter x3, y3: 2

Enter y3: 4

The points do not lie on a straight line.

12# Given the coordinates (x, y) of centre of a circle and its radius, determine whether a point lies inside the circle, on the circle or outside the circle using if else statement

```
x = float(input("Enter the x-coordinate of the circle's centre: "))
```

```
y = float(input("Enter the y-coordinate of the circle's centre: "))
```

```
r = float(input("Enter the radius of the circle: "))
```

```
px = float(input("Enter the x-coordinate of the point: "))
```

```
py = float(input("Enter the y-coordinate of the point: "))
```

```
distance squared = (px - x) ** 2 + (py - y) ** 2
```

```
radius squared = r ** 2
```

```
if distance squared < radius squared:
```

```
    print("The point lies inside the circle.")
elif distance squared == radius squared:
    print("The point lies on the circle.")
else:
    print("The point lies outside the circle.")
```

Output:-

Enter the x-coordinate of the circle's centre: 5

Enter the y-coordinate of the circle's centre: 2

Enter the radius of the circle: 3

Enter the x-coordinate of the point: 8

Enter the y-coordinate of the point: 1

The point lies outside the circle.

13# Convert number 0 to 19 to its equivalent words. E.g. 0 à zero, 19ànineteen.

```
number = int(input("Enter a number between 0 and 19: "))
```

```
if number == 0:
```

```
    print("zero")
```

```
elif number == 1:
```

```
    print("one")
```

```
elif number == 2:
```

```
    print("two")
```

```
elif number == 3:
```

```
    print("three")
```

```
elif number == 4:
```

```
    print("four")
```

```
elif number == 5:
```

```
    print("five")
```

```
elif number == 6:
```

```
    print("six")
```

```
elif number == 7:
```

```
    print("seven")
elif number == 8:
    print("eight")
elif number == 9:
    print("nine")
elif number == 10:
    print("ten")
elif number == 11:
    print("eleven")
elif number == 12:
    print("twelve")
elif number == 13:
    print("thirteen")
elif number == 14:
    print("fourteen")
elif number == 15:
    print("fifteen")
elif number == 16:
    print("sixteen")
elif number == 17:
    print("seventeen")
elif number == 18:
    print("eighteen")
elif number == 19:
    print("nineteen")
else:
    print("Invalid input! Please enter a number between 0 and 19.")
```

Output:-

Enter a number between 0 and 19: 5

Five

14# Accept marks of three subjects. Print total and average along with whether a candidate has passed or fail. If student secures  $\leq 39$  marks in any subject, consider him as fail. Also assigned a subject wise grade based on the following table: -

```
subject1 = int(input("Enter marks for Subject 1: "))
```

```
subject2 = int(input("Enter marks for Subject 2: "))
```

```
subject3 = int(input("Enter marks for Subject 3: "))
```

```
pass status = True
```

```
if subject1  $\leq$  39 or subject2  $\leq$  39 or subject3  $\leq$  39:
```

```
    pass status = False
```

```
total marks = subject1 + subject2 + subject3
```

```
average marks = total marks / 3
```

```
if subject1  $\geq$  90:
```

```
    grade1 = "A"
```

```
elif subject1  $\geq$  75:
```

```
    grade1 = "B"
```

```
elif subject1  $\geq$  50:
```

```
    grade1 = "C"
```

```
elif subject1  $\geq$  40:
```

```
    grade1 = "D"
```

```
else:
```

```
    grade1 = "Fail"
```

```
if subject2  $\geq$  90:
```

```
    grade2 = "A"
```

```
elif subject2  $\geq$  75:
```

```
    grade2 = "B"
```

```
elif subject2  $\geq$  50:
```

```
    grade2 = "C"
```

```
elif subject2 >= 40:
    grade2 = "D"
else:
    grade2 = "Fail"

if subject3 >= 90:
    grade3 = "A"
elif subject3 >= 75:
    grade3 = "B"
elif subject3 >= 50:
    grade3 = "C"
elif subject3 >= 40:
    grade3 = "D"
else:
    grade3 = "Fail"
print("\n Results:")
print( "Total Marks: {total marks}")
print( "Average Marks: {average_marks:.2f}")

if pass status:
    print("Status: Pass")
else:
    print("Status: Fail")

print( "Grade in Subject 1: {grade1}")
print("Grade in Subject 2: {grade2}")
print( "Grade in Subject 3: {grade3}")
Output:-
```

Enter marks for Subject 1: 55



Enter marks for Subject 2: 86

Enter marks for Subject 3: 24

Results:

Total Marks: 165

Average Marks: 55.00

Status: Fail

Grade in Subject 1: C

Grade in Subject 2: B

Grade in Subject 3: Fail

## Assignment 3

Q-1

```
s = input("Enter a string: ")
```

```
vowels = "aeiouAEIOU"
```

```
count = sum(1 for char in s if char in vowels)
```

```
print("Number of vowels in the string:", count)
```

OUTPUT:

Enter a string: aehbjbh

Number of vowels in the string: 2

Q-2

```
def to_lower(char):
```

```
    if 'A' <= char <= 'Z':
```

```
        return chr(ord(char) + 32)
```

```
    return char
```

```
def to_upper(char):
```

```
    if 'a' <= char <= 'z':
```

```
        return chr(ord(char) - 32)
```

```
    return char
```

```
def toggle_case(char):
```

```
    if 'a' <= char <= 'z':
```

```
        return chr(ord(char) - 32)
```

```
    elif 'A' <= char <= 'Z':
```

```
        return chr(ord(char) + 32)
```

```
    return char
```

```
def to_lower_string(string):
```

```
    result = ""
```

```
    for char in string:
```

```
        result += to_lower(char)
```

```
    return result
```

```
def to_upper_string(string):
```

```
    result = ""
```

```
    for char in string:
```

```
        result += to_upper(char)
```

```
    return result
```

```
def toggle_case_string(string):
```

```
    result = ""
```

```
    for char in string:
```

```
        result += toggle_case(char)
```

```
    return result
```

```
string = input("Enter a string: ")
```

```
print("Lowercase:", to_lower_string(string))
print("Uppercase:", to_upper_string(string))
print("Toggle Case:", toggle_case_string(string))
```

OUTPUT:

Enter a string: TUSHAR

Lowercase: tushar

Uppercase: TUSHAR

Toggle Case: tushar

Q-3

```
def is_substring(str1, str2):
    len1, len2 = len(str1), len(str2)

    if len2 > len1:
        return False

    for i in range(len1 - len2 + 1):
        match = True
        for j in range(len2):
            if str1[i + j] != str2[j]:
                match = False
                break
        if match:
            return True
    return False
```

```
str1 = input("Enter the main string: ")
```

```
str2 = input("Enter the substring: ")
```

```
if is_substring(str1, str2):
```

```
    print("Yes, the substring is present.")
```

```
else:
```

```
    print("No, the substring is not present.")
```

OUTPUT:

Enter the main string: tushar

Enter the substring: tushar

Yes, the substring is present.

Q-4

```
def remove_substring(onestring, removestring):
```

```
    len1, len2 = len(onestring), len(removestring)
```

```
    if len2 > len1:
```

```
        return onestring
```

```
    i = 0
```

```
    while i <= len1 - len2:
```

```
        match = True
```

```
        for j in range(len2):
```

```
            if onestring[i + j] != removestring[j]:
```

```
                match = False
```

```
                break
```

```
        if match:
```

```
            onestring = onestring[:i] + onestring[i + len2:]
```

```
            len1 = len(onestring)
```

```
            i -= 1
```

```
            i += 1
```

```
return onestring
```

```
onestring = input("Enter the main string: ")
removestring = input("Enter the substring to remove: ")
finalstring = remove_substring(onestring, removestring)
print("Final string:", finalstring)
```

output:

Enter the main string: tuhuirb

Enter the substring to remove: hui

Final string: turb

## Assignment 4

1# Print all alphabets in upper case and in lower case.

```
for i in range(65, 91):
```

```
    print(chr(i), end=" ")
```

```
print()
```

```
for i in range(97, 123):
```

```
    print(chr(i), end=" ")
```

# Output:-

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

2# Print a multiplication table of a given number.

```
num = int(input("Enter a number: "))
```

```
for i in range(1, 11):
```

```
    print(num, "x", i, "=", num * i)
```

# Output :-

Enter a number: 7

7 x 1 = 7  
7 x 2 = 14  
7 x 3 = 21  
7 x 4 = 28  
7 x 5 = 35  
7 x 6 = 42  
7 x 7 = 49  
7 x 8 = 56  
7 x 9 = 63  
7 x 10 = 70

3# Count no. of alphabets and no. of digits in any given string.

```
s = input("Enter a string: ")
alphabet count = 0
digit count = 0
for char in s:
    if ('A' <= char <= 'Z') or ('a' <= char <= 'z'):
        alphabet count += 1
    elif '0' <= char <= '9':
        digit count += 1
print("Number of alphabets:", alphabet count)
print("Number of digits:", digit count)
# Output:-
Enter a string: 24BCH000@sptdpuacin
Number of alphabets: 14
Number of digits: 5
```

4# Check whether a given number is prime, is perfect, is Armstrong, is palindrome, is automorphic.

```
num = int(input("Enter a number: "))
is prime = True
if num < 2:
    is prime = False
else:
    for i in range(2, num):
        if num % i == 0:
            is prime = False
            break
sum factors = 0
for i in range(1, num):
    if num % i == 0:
        sum factors += i
is perfect = (sum factors == num)
sum digits = 0
temp = num
order = len(str(num))
while temp > 0:
    digit = temp % 10
    sum digits += digit ** order
    temp //= 10
is armstrong = (sum digits == num)
temp = num
rev = 0
while temp > 0:
    rev = rev * 10 + temp % 10
    temp //= 10
is palindrome = (num == rev)
square = num * num
```

```
is automorphic = (str(square).ends with(str(num)))
print("Prime:", "Yes" if is prime else "No")
print("Perfect:", "Yes" if is perfect else "No")
print("Armstrong:", "Yes" if is armstrong else "No")
print("Palindrome:", "Yes" if is palindrome else "No")
print("Automorphic:", "Yes" if is automorphic else "No")
```

#Output :-

Enter a number: 37

Prime: Yes

Perfect: No

Armstrong: No

Palindrome: No

Automorphic: No

5# Generate all Pythagorean Triplets with side length <= 30.

```
for a in range(1, 31):
    for b in range(a, 31):
        for c in range(b, 31):
            if a**2 + b**2 == c**2:
                print(a, b, c)
```

#Output:-

3 4 5

5 12 13

6 8 10

7 24 25

8 15 17

9 12 15

10 24 26

12 16 20



15 20 25

18 24 30

20 21 29

6# Print 24 hours of day with suitable suffixes like AM, PM, Noon and Midnight.

for hour in range(24):

if hour == 0:

print("12 Midnight")

elif hour == 12:

print("12 Noon")

elif hour < 12:

print(hour, "AM")

else:

print(hour - 12, "PM")

#Output :-

12 Midnight

1 AM

2 AM

3 AM

4 AM

5 AM

6 AM

7 AM

8 AM

9 AM

10 AM

11 AM

12 Noon

1 PM

2 PM

3 PM

4 PM

5 PM

6 PM

7 PM

8 PM

9 PM

10 PM

11 PM

7#Print n Cr and nPr.

```
n = int(input("Enter n: "))
```

```
r = int(input("Enter r: "))
```

```
fact n = 1
```

```
for i in range(1, n + 1):
```

```
    fact n *= i
```

```
fact n r = 1
```

```
for i in range(1, n - r + 1):
```

```
    fact n r *= i
```

```
fact r = 1
```

```
for i in range(1, r + 1):
```

```
    fact r *= i
```

```
nPr = fact n // fact n r
```

```
n Cr = fact n // (fact r * fact n r)
```

```
print("nPr (Permutation):", nPr)
```

```
print("n Cr (Combination):", n Cr)
```

#Output :-

Enter n: 7

Enter r: 4

nPr (Permutation): 840

n Cr (Combination): 35

8# Print factorial of a given number.

```
n = int(input("Enter a number: "))
```

```
fact = 1
```

```
for i in range(1, n + 1):
```

```
    fact *= i
```

```
print("Factorial of", n, "is:", fact)
```

#Output:-

Enter a number: 6

Factorial of 6 is: 720

9# Print N natural nos. in reverse

```
N = int(input("Enter a number: "))
```

```
for i in range(N, 0, -1):
```

```
    print(i, end=" ")
```

#Output:-

Enter a number: 13

13 12 11 10 9 8 7 6 5 4 3 2 1

10# Generate N numbers of Fibonacci series.

```
N = int(input("Enter the number of Fibonacci terms: "))
```

```
a = 0
```

```
b = 1
```

```
print("Fibonacci Series:", end=" ")
```

```
for i in range(N):
```

```
    print(a, end=" ")
```

```
temp = a + b
```

```
a = b
```

```
b = temp
```

#Output :-

Enter the number of Fibonacci terms: 15

Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

11# Calculate  $\sin(x)$ ; x is a radian value. The formula is as under:

```
x = float(input("Enter the value of x in radians: "))
```

```
terms = 10
```

```
sin x = 0
```

```
for n in range(terms):
```

```
    term = ((-1) ** n) * (x ** (2 * n + 1))
```

```
    fact = 1
```

```
    for i in range(1, (2 * n + 1) + 1):
```

```
        fact *= i
```

```
    sin x += term / fact
```

```
print("sin(", x, ") ≈", sin x)
```

#Output:-

Enter the value of x in radians: 2.64

$\sin(2.64) \approx 0.48082261497486406$

## Assignment 5

```
import random
```

```
odd_integers = [random.choice(range(1, 100, 2)) for _ in range(5)]
```

```
print("List of 5 odd integers:", odd_integers)
```

```
even_integers = [random.choice(range(2, 100, 2)) for _ in range(4)]  
print("List of 4 even integers:", even_integers)
```

```
odd_integers[2] = even_integers  
print("Updated list of odd integers with 3rd element replaced:", odd_integers)
```

```
flattened_list = []  
for item in odd_integers:  
    if isinstance(item, list):  
        flattened_list.extend(item)  
    else:  
        flattened_list.append(item)  
print("Flattened list:", flattened_list)
```

```
flattened_list.sort()  
print("Sorted flattened list:", flattened_list)
```

OUTPUT:

List of 5 odd integers: [53, 37, 89, 67, 89]

List of 4 even integers: [78, 14, 74, 70]

Updated list of odd integers with 3rd element replaced: [53, 37, [78, 14, 74, 70], 67, 89]

Flattened list: [53, 37, 78, 14, 74, 70, 67, 89]

Sorted flattened list: [14, 37, 53, 67, 70, 74, 78, 89]

Q-2

```
import random  
random_integers = [random.randint(1, 100) for _ in range(20)]  
print("Generated list of 20 random integers:", random_integers)  
user_number = int(input("Enter a number to find its positions in the list: "))  
positions = [index for index, value in enumerate(random_integers) if value ==  
user_number]
```

if positions:

```
    print(f"The number {user_number} is found at the following position(s):  
{positions}")
```

else:

```
    print(f"The number {user_number} is not found in the list.")
```

OUTPUT:

Generated list of 20 random integers: [10, 33, 73, 6, 93, 100, 64, 73, 41, 26, 36, 78, 10, 82, 50, 38, 55, 23, 36, 41]

Enter a number to find its positions in the list: 100

The number 100 is found at the following position(s): [5]

Q-3

```
import random
```

```
random_numbers = [random.randint(1, 30) for _ in range(50)]
```

```
print("Generated list of 50 random numbers:", random_numbers)
```

```
unique_numbers = list(set(random_numbers))
```

```
print("List after removing duplicates:", unique_numbers)
```

OUTPUT:

Generated list of 50 random numbers: [8, 5, 7, 25, 6, 16, 27, 28, 3, 26, 4, 16, 13, 23, 8, 30, 23, 20, 1, 19, 9, 20, 6, 15, 16, 3, 8, 11, 3, 15, 2, 11, 17, 14, 4, 9, 28, 17, 17, 26, 30, 7, 11, 12, 27, 3, 26, 18, 29, 2]

List after removing duplicates: [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23, 25, 26, 27, 28, 29, 30]

Q-4

```
import random
```

```
random_numbers = [random.randint(-100, 100) for _ in range(30)]
```

```
print("Generated list of 30 random numbers:", random_numbers)
```

```
positive_numbers = [num for num in random_numbers if num > 0]
```

```
negative_numbers = [num for num in random_numbers if num < 0]
```

```
print("List of positive numbers:", positive_numbers)
```

```
print("List of negative numbers:", negative_numbers)
```

OUTPUT:

Generated list of 30 random numbers: [-92, 41, -29, -25, 95, 36, 78, -89, -27, 47, 14, -24, -73, 10, 87, -19, 26, 1, 97, -47, 80, -88, 14, -46, 27, 48, 11, 82, 97, 98]

List of positive numbers: [41, 95, 36, 78, 47, 14, 10, 87, 26, 1, 97, 80, 14, 27, 48, 11, 82, 97, 98]

List of negative numbers: [-92, -29, -25, -89, -27, -24, -73, -19, -47, -88, -46]

Q-5

```
strings = ["hello", "world", "tushar", "kathiriya", "patel"]
```

```
print("Original list of strings:", strings)
```

```
uppercase_strings = [string.upper() for string in strings]
```

```
print("List of strings in uppercase:", uppercase_strings)
```

OUTPUT:

Original list of strings: ['hello', 'world', 'tushar', 'kathiriya', 'patel']

List of strings in uppercase: ['HELLO', 'WORLD', 'TUSHAR', 'KATHIRIYA', 'PATEL']

Q-6

```
fahrenheit_temps = [90,87,76,25,35]
```

```
print("Temperatures in Fahrenheit:", fahrenheit_temps)
```

cels

```
if self.is_empty():
```

```
    print("Stack is empty.")
```

```
else:
```

```
    print(f"Top element is: {self.stack[-1]}")
```

```
def is_empty(self):
```

```
    return len(self.stack) == 0
```

```
def display(self):
```

```
        if self.is_empty():
            print("Stack is empty.")
        else:
            print("Stack elements:", self.stack)
def menu():
    stack = Stack()
    while True:
        print("\nMenu:")
        print("1. Push element to stack")
        print("2. Pop element from stack")
        print("3. Peek top element")
        print("4. Check if stack is empty")
        print("5. Display stack")
        print("6. Exit")
        choice = input("Enter your choice: ")
        if choice == "1":
            item = int(input("Enter the element to push onto the stack: "))
            stack.push(item)
        elif choice == "2":
            stack.pop()
        elif choice == "3":
            stack.peak()
        elif choice == "4":
            if stack.is_empty():
                print("Stack is empty.")
            else:
                print("Stack is not empty.")
        elif choice == "5":
            stack.display()
```



```
elif choice == "6":
    print("Exiting program.")
    break
else:
    print("Invalid choice, please try again.")
menu()
```

OUTPUT:

Menu:

1. Push element to stack
2. Pop element from stack
3. Peek top element
4. Check if stack is empty
5. Display stack
6. Exit

Q-8

class Queue:

```
def __init__(self):
    self.queue = []

def enqueue(self, item):
    self.queue.append(item)
    print(f"{item} has been added to the queue.")

def dequeue(self):
    if self.is_empty():
        print("Queue is empty, cannot dequeue.")
    else:
        dequeued_item = self.queue.pop(0)
        print(f"{dequeued_item} has been removed from the queue.")
```

```
def peek(self):
    if self.is_empty():
        print("Queue is empty.")
    else:
        print(f"Front element is: {self.queue[0]}")
```

```
def is_empty(self):
    return len(self.queue) == 0
```

```
def display(self):
    if self.is_empty():
        print("Queue is empty.")
    else:
        print("Queue elements:", self.queue)
```

```
def menu():
    queue = Queue()
    while True:
        print("\nMenu:")
        print("1. Enqueue element to queue")
        print("2. Dequeue element from queue")
        print("3. Peek front element")
        print("4. Check if queue is empty")
        print("5. Display queue")
        print("6. Exit")

        choice = input("Enter your choice: ")
```

```
if choice == "1":
    item = int(input("Enter the element to enqueue to the queue: "))
    queue.enqueue(item)
elif choice == "2":
    queue.dequeue()
elif choice == "3":
    queue.peek()
elif choice == "4":
    if queue.is_empty():
        print("Queue is empty.")
    else:
        print("Queue is not empty.")
elif choice == "5":
    queue.display()
elif choice == "6":
    print("Exiting program.")
    break
else:
    print("Invalid choice, please try again.")

menu()
```

OUTPUT:

Menu:

1. Enqueue element to queue
2. Dequeue element from queue
3. Peek front element
4. Check if queue is empty
5. Display queue
6. Exit

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

```
list2 = [4, 5, 6, 8, 9]
```

```
list3 = [num for num in list1 if num not in list2]
```

```
print("Numbers from list1 that are not in list2:", list3)
```

OUTPUT:

Numbers from list1 that are not in list2: [1, 2, 3, 7]

## Assignment 6

Q-1

```
Names_list = [("John", "Doe"), ("James", "Smith"), "Emily", "Sophia", ("Alex", "Brown"),  
"Mia"]
```

```
Boys_count = 0
```

```
Girls_count = 0
```

```
For ele in names_list:
```

```
    If isinstance(ele, tuple):
```

```
        Boys_count += 1
```

```
    Else:
```

```
        Girls_count += 1
```

```
Print(f"Number of boys: {boys_count}")
```

```
Print(f"Number of girls: {girls_count}")
```

OUTPUT:

Number of boys: 3

Number of girls: 3

Q-2

```
Students_list = [(1, "TUSHAR", 15), (2, "ANSH", 16), (3, "NAMAN", 14)]
```

```
Roll_no_list = []
```

```
Name_list = []
```

```
Age_list = []
```

```
For student in students_list:
```

```
    Roll_no_list.append(student[0])
```

```
    Name_list.append(student[1])
```

```
    Age_list.append(student[2])
```

```
Print("Roll No. List:", roll_no_list)
```

```
Print("Name List:", name_list)
```

```
Print("Age List:", age_list)
```

OUTPUT:

Roll No. List: [1, 2, 3]

Name List: ['TUSHAR', 'ANSH', 'NAMAN']

Age List: [15, 16, 14]

Q-3

```
From datetime import datetime
```

```
Def convert_to_date(date_tuple):
```

```
    Day, month, year = date_tuple
```

```
    Return datetime(year, month, day)
```

```
Date1 = (15, 2, 2025)
```

```
Date2 = (22, 2, 2025)
```

```
Date1_obj = convert_to_date(date1)
```

```
Date2_obj = convert_to_date(date2)
```

```
Date_difference = date2_obj - date1_obj
```

```
Print(f"Number of days between {date1} and {date2}: {date_difference.days} days")
```

OUTPUT:

Number of days between (15, 2, 2025) and (22, 2, 2025): 7 days

Q-4

```
Food_items = [("Burger", 5), ("Pizza", 8), ("Pasta", 7), ("Salad", 4), ("Sushi", 12)]
```

```
Sorted_food_items = sorted(food_items, key=lambda x: x[1], reverse=True)
```

```
Print(sorted_food_items)
```

OUTPUT:

```
[('Sushi', 12), ('Pizza', 8), ('Pasta', 7), ('Burger', 5), ('Salad', 4)]
```

Q-5

```
Tuple_list = [("Apple", 3), (), ("Banana", 2), (), ("Orange", 5)]
```

```
Cleaned_list = [t for t in tuple_list if t]
```

```
Print(cleaned_list)
```

OUTPUT:

```
[('Apple', 3), ('Banana', 2), ('Orange', 5)]
```

Q-6

```
My_tuple = (1, 2, 3, 4, 5)
```

```
Temp_list = list(my_tuple)
```

```
Temp_list[2] = 10
```

```
Modified_tuple = tuple(temp_list)
```

```
Print(modified_tuple)
```

OUTPUT:

```
(1, 2, 10, 4, 5)
```

Q-7

```
My_tuple = (1, 2, 3, 4, 5)
```

```
My_tuple = my_tuple[:2] + my_tuple[3:]
```

```
Print(my_tuple)
```



OUTPUT:

(1, 2, 4, 5)

1) Lst = [ ('X', 'Y', 'Z'), 40, 50 , 60]

A = lst[0]

Print (a)

a) X                      b) 0                      c) (x, y, z)                      d) ('x', 'y', 'z')

ANS:D

## Assignment 7

Q-1

```
dict1 = {'a': 1, 'b': 2} dict2 = {'c': 3, 'd': 4} dict3 = {'e': 5, 'f': 6} dict4 = {**dict1, **dict2,
**dict3} print("Concatenated Dic onary:", dict4)
```

OUTPUT:

Concatenated Dic onary: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6} Q-2 def check\_empty(dic  
onary):    if not dic onary:

    return "The dic onary is empty."

    else:

        return "The dic onary is not empty." dict1 = {} dict2 = {'a': 1, 'b': 2}

print(check\_empty(dict1)) print(check\_empty(dict2)) OUTPUT:

The dic onary is empty.

The dic onary is not empty. Q-3 employees = {

```

101: [
    {'roll_no': 1, 'salary': 4000},
    {'roll_no': 2, 'salary': 5000},
    {'roll_no': 3, 'salary': 3000}
],
102: [
    {'roll_no': 4, 'salary': 6000},
    {'roll_no': 5, 'salary': 7000},
    {'roll_no': 6, 'salary': 5500}
],
103: [
    {'roll_no': 7, 'salary': 3500},
    {'roll_no': 8, 'salary': 4500},
    {'roll_no': 9, 'salary': 6500}
]
}

```

```

for dept_no, employee_list in employees.items():

```

```

    min_salary = min(employee_list, key=lambda x: x['salary'])['salary']    max_salary =
    max(employee_list, key=lambda x: x['salary'])['salary']

```

```

    print(f"Department {dept_no}:")    print(f" Minimum Salary: {min_salary}")    print(f"
    Maximum Salary: {max_salary}")

```

```

    print()

```

OUTPUT:

Department 101:

```

_string):

```

```

    freq_dict = {}

```

```

for char in input_string:    if char in freq_dict:    freq_dict[char] += 1
    else:
        freq_dict[char] = 1    return freq_dict
input_string = input("Enter a string: ")
freq_dict = character_frequency(input_string)
print("Character Frequency in the given string:")
for char, freq in freq_dict.items():
    print(f"{char}: {freq}")

```

OUTPUT:

Enter a string: tusjafdsjk

Character Frequency in the given string:

't': 1

'u': 1

's': 2

'j': 2

'a': 1

'f': 1

'd': 1

'k': 2

Q-5

```

prices = {
    'apple': 2.5,
    'banana': 1.2,
    'orange': 3.0,
    'milk': 1.5,
    'bread': 2.0
}

```

```

quantities = {

```

## Assignment 8

Q-1 def convert\_to\_uppercase(word\_list):

```
uppercased_set = {word.upper() for word in word_list} return uppercased_set
```

```
words = ["hello", "world", "python", "programming", "hello"]
```

```
result = convert_to_uppercase(words) print(result)
```

OUTPUT:

```
{'PYTHON', 'PROGRAMMING', 'WORLD', 'HELLO'} Q-2 import random
```

```
random_numbers = set() while len(random_numbers) < 10:
```

```
    random_numbers.add(random.randint(15, 45)) count_less_than_30 = 0 for num in random_numbers:
```

```
    if num < 30:
```

```
        count_less_than_30 += 1 random_numbers = {num for num in random_numbers if num <= 35} print("Random numbers (a er dele ng numbers greater than 35):", random_numbers) print("Count of numbers less than 30:", count_less_than_30)
```

OUTPUT:

```
Random numbers (a er dele ng numbers greater than 35): {32, 33, 15, 18, 22, 26, 28}
```

```
Count of numbers less than 30: 5
```

Q-3

```
names_set = set() names_set.add("TUSHAR") names_set.add("lavkesh")
```

```
names_set.add("dhruv") names_set.add("mihir") names_set.add("nikhil") if "mihir" in names_set:
```

```
    names_set.remove("mihir") names_set.add("subham") names_set.discard("mihir") names_set.discard("lavkesh") print("Final set of names:", names_set)
```

## Assignment 9

Q1

```

Def prime_factors(n, divisor=2):
    # Base case: If n becomes 1, return an empty list
    If n <= 1:
        Return []

    # If n is divisible by the current divisor, it's a prime factor
    If n % divisor == 0:
        Return [divisor] + prime_factors(n // divisor, divisor)

    # Otherwise, increment the divisor to check the next number
    Return prime_factors(n, divisor + 1)

# Input from the user
Try:
    Num = int(input("Enter a positive integer to find its prime factors: "))
    If num <= 0:
        Print("Please enter a positive integer.")
    Else:
        Factors = prime_factors(num)
        Print(f"Prime factors of {num} are: {factors}")
Except ValueError:

```

Q2

```

Def find_binary(n):
    # Base case: if n is 0 or 1, return it as a string
    If n <= 1:
        Return str(n)
    Else:
        # Recursive call: Divide n by 2 and find the remainder

```

```
Return find_binary(n // 2) + str(n % 2)
```

# Input from the user

Try:

```
Num = int(input("Enter a positive integer to find its binary equivalent: "))
```

```
If num < 0:
```

```
    Print("Please enter a positive integer.")
```

```
Else:
```

```
    Print(f"Binary equivalent of {num} is: {find_binary(num)}")
```

Except ValueError:

```
    Print("Invalid input. Please enter a valid integer.")
```

Output

Enter a positive integer to find its binary equivalent: 10

Binary equivalent of 10 is: 1010

Enter a positive integer to find its binary equivalent: 25

Binary equivalent of 25 is: 11001

Enter a positive integer to find its binary equivalent: 0

Binary equivalent of 0 is: 0

Enter a positive integer to find its binary equivalent: 1

Binary equivalent of 1 is: 1

Q3

Def count\_vowels(s):

```
    # Base case: if the string is empty, return 0
```

```
    If len(s) == 0:
```

```
        Return 0
```

```
# Check if the first character is a vowel
```

```
Vowels = 'aeiouAEIOU'
```

```
If s[0] in vowels:
```

```
    Return 1 + count_vowels(s[1:])
```

```
Else:
```

```
    Return count_vowels(s[1:])
```

```
# Input from the user
```

```
String = input("Enter a string: ")
```

```
# Call the recursive function
```

```
Vowel_count = count_vowels(string)
```

```
Number of vowels in the string: 3
```

```
Enter a string: Python Programming
```

```
Number of vowels in the string: 5
```

```
Q4
```

```
Def reverse_list(lst):
```

```
    # Base case: if the list is empty or has only one element
```

```
    If len(lst) <= 1:
```

```
        Return lst
```

```
    Else:
```

```
        # Recursive case: reverse the rest of the list and append the first element to the  
end
```

```
        Return reverse_list(lst[1:]) + [lst[0]]
```

```
# Example input
```

```
Numbers = [1, 2, 3, 4, 5]
```

```
Print("Original List:", numbers)
```

```
# Reversing using the recursive function
```

```
Reversed_numbers = reverse_list(numbers)
```

```
Print("Reversed List:", reversed_numbers)
```

Output

```
Original List: [1, 2, 3, 4, 5]
```

```
Reversed List: [5, 4, 3, 2, 1]
```

```
# If b is negative, convert to positive and negate the result
```

```
Else:
```

```
    Return -multiply(a, -b)
```

```
# Taking input using keyword arguments
```

```
A = int(input("Enter value for a: "))
```

```
B = int(input("Enter value for b: "))
```

```
# Calling the recursive function
```

```
Result = multiply(a=a, b=b)
```

```
# Display the result
```

```
Print(f"The product of {a} and {b} is: {result}")
```

Output

```
Enter value for a: 5
```

```
Enter value for b: 3
```

```
The product of 5 and 3 is: 15
```

Q6

```
def sanitize_list(lst, index=0):
```



```

# Base case: if index reaches the end of the list, return the list
if index == len(lst):
    return lst

# Replace negative value with 0
if lst[index] < 0:
    lst[index] = 0

# Recursive call for the next index
return sanitize_list(lst, index + 1)

# Example usage
my_list = [4, -3, 7, -1, 0, 9, -8, 6]
print("Original List:", my_list)

sanitized_list = sanitize_list(my_list)
print("Sanitized List:", sanitized_list)

Q7

Def recursive_sum(numbers):
    # Base case: if the list has only one number, return that number
    If len(numbers) == 1:
        Return numbers[0]

    # Recursive case: sum the first number with the sum of the rest of the list
    Else:
        Return numbers[0] + recursive_sum(numbers[1:])

Def find_average(numbers):
    If len(numbers) == 0:

```

```
    Return 0 # Return 0 if the list is empty to avoid division by zero
Total_sum = recursive_sum(numbers)
Return total_sum / len(numbers)
Numbers_list = [10, 20, 30, 40, 50]
Average = find_average(numbers_list)
Print(f"The average of {numbers_list} is: {average}")
```

Output

The average of [10, 20, 30, 40, 50] is: 30.0

Q8

```
Def string_length(s):
```

```
    # Base case: if the string is empty, return 0
```

```
    If s == "":
```

```
        Return 0
```

```
    Else:
```

```
        # Recursive case: reduce the string by one character and add 1
```

```
        Return 1 + string_length(s[1:])
```

```
# Input from user
```

```
User_input = input("Enter a string: ")
```

```
Length = string_length(user_input)
```

```
Print(f"The length of the string is: {length}")
```

Output

Enter a string: Hello World

The length of the string is: 11

## Assignment 10

Q1

```
Def fun():
```

```
    Print("This is function fun()")
```

```
Def disp():
```

```
    Print("This is function disp()")
```

```
Def msg():
```

```
    Print("This is function msg()")
```

```
# Storing functions in a list
```

```
Functions_list = [fun, disp, msg]
```

```
# Calling functions one by one using a loop
```

```
For func in functions_list:
```

```
    Func()
```

Output

This is function fun()

This is function disp()

This is function msg()

Q2

```
# Define the two lists
```

```
List1 = [1, 2, 3, 4, 5, 6]
```

```
List2 = [6, 5, 4, 3, 2, 1]
```

```
# Use map and lambda to add corresponding elements
```

```
Result = list(map(lambda x, y: x + y, list1, list2))
```

```
# Print the result
```

```
Print("Resultant List:", result)
```

Output

```
Resultant List: [7, 7, 7, 7, 7, 7]
```

Q3

```
Import random
```

```
# Generate a list of 10 random numbers between -15 and 15
```

```
Random_numbers = [random.randint(-15, 15) for _ in range(10)]
```

```
Print("Random Numbers:", random_numbers)
```

```
# Create a new list with the squares of these numbers
```

```
Squared_numbers = [x**2 for x in random_numbers]
```

```
Print("Squared Numbers:", squared_numbers)
```

Output

```
Random Numbers: [12, -3, 7, -15, 0, 11, -5, 6, -13, 9]
```

```
Squared Numbers: [144, 9, 49, 225, 0, 121, 25, 36, 169, 81]
```

Q4

```
# Given list
```

```
Lst = ['madam', 'Python', 'malayalam', 12321]
```

```
# Function to check palindrome
```

```
Def is_palindrome(s):
```

```
    # Convert to string to handle both strings and integers
```

```
    S = str(s)
```

```
    Return s == s[::-1]
```

```
# Print palindrome strings
Print("Palindromes in the list are:")
For item in lst:
    If is_palindrome(item):
        Print(item)
```

Output

Palindromes in the list are:

Madam

Malayalam

12321

Q5

```
# List of faculty member names
```

```
Faculty_names = ["Alexander", "John", "Catherine", "Mike", "Elizabeth", "Rajesh",  
"Jennifer", "Robert"]
```

```
# Using list comprehension to filter names with length more than 8 characters
```

```
Filtered_names = [name for name in faculty_names if len(name) <= 8]
```

```
# Display the result
```

```
Print("Names with 8 or fewer characters:", filtered_names)
```

Output

Names with 8 or fewer characters: ['John', 'Mike', 'Rajesh', 'Robert']

## Assignment 11

Q1

```
import csv
```

```
# Data to write into CSV
```

```

data = [
    ["Name", "Subject", "Score"],
    ["Alice", "Math", 90],
    ["Bob", "Science", 85],
    ["Charlie", "History", 78],
    ["Diana", "English", 92]
]

# Creating CSV file
filename = "students_scores.csv"
with open(filename, mode="w", newline="") as file:
    writer = csv.writer(file)
    writer.writerows(data)
print(f"CSV file '{filename}' created successfully.")

```

Output

CSV file 'students\_scores.csv' created successfully.

Q2

Import openpyxl

# Load the Excel file

```
Wb = openpyxl.load_workbook("students_data.xlsx")
```

```
Sheet = wb.active
```

```
Students_dict = {}
```

# Skip header and start from row 2

```
For row in sheet.iter_rows(min_row=2, values_only=True):
```

```
    Rollno, name, sub1, sub2, sub3 = row
```

```
    Total = sub1 + sub2 + sub3
```

```
    Students_dict[rollno] = {
```

```
        "Name": name,
```

```
        "Subject1": sub1,
```

```
        "Subject2": sub2,
```

```
"Subject3": sub3,
```

```
"Total": total
```

```
}
```

```
# Display the dictionary
```

```
For rollno, data in students_dict.items():
```

```
    Print(f"Roll No: {rollno}")
```

```
    For key, value in data.items():
```

```
        Print(f" {key}: {value}")
```

```
    Print()
```

Output

Roll No: 1

Name: Alice

Subject1: 78

Subject2: 82

Subject3: 91

Total: 251

Roll No: 2

Name: Bob

Subject1: 69

Subject2: 73

Subject3: 80

Total: 222

Roll No: 3

Name: Charlie

Subject1: 88

Subject2: 85

Subject3: 90

Total: 263

Q3

# Accept contact details from user

Name = input("Enter Full Name: ")

Phone = input("Enter Phone Number: ")

Email = input("Enter Email Address: ")

Address = input("Enter Address: ")

# Create vCard content



```
Vcard_data = f"""BEGIN:VCARD
```

```
VERSION:3.0
```

```
FN:{name}
```

```
TEL;TYPE=CELL:{phone}
```

```
EMAIL:{email}
```

```
ADR;TYPE=HOME;;;{address}
```

```
END:VCARD
```

```
"""
```

```
# Save to .vcf file
```

```
Filename = "contact.vcf"
```

```
With open(filename, "w") as file:
```

```
    File.write(vcard_data)
```

```
Print(f"\nvCard saved as '{filename}'. You can import it into your mobile contacts.")
```

Output

Enter Full Name: John Doe

Enter Phone Number: +1234567890

Enter Email Address: john@example.com

Enter Address: 123 Street Name, City, Country

vCard saved as 'contact.vcf'. You can import it into your mobile contacts.

Q4

Import os

Import shutil

```
# Define paths
```

```
Source_dir = "source_folder"
```

```
Destination_dir = "destination_folder"
```

```
File_to_copy = "example.txt"
```

```
# Create source and destination folders if not exist
```

```
Os.makedirs(source_dir, exist_ok=True)
```

```
Os.makedirs(destination_dir, exist_ok=True)
```

```
# Create a sample file in the source folder
```

```
Source_file_path = os.path.join(source_dir, file_to_copy)
```

```
With open(source_file_path, "w") as f:
```

```
f.write("This is an example file to be copied.")
```

```
# Copy the file to the destination folder
```

```
Destination_file_path = os.path.join(destination_dir, file_to_copy)
```

```
Shutil.copy(source_file_path, destination_file_path)
```

```
Print(f"File '{file_to_copy}' copied from '{source_dir}' to '{destination_dir}'  
successfully.")
```

Output

File 'example.txt' copied from 'source\_folder' to 'destination\_folder' successfully.

Q5

# File names

Source\_file = "input.txt"

Destination\_file = "output.txt"

# Step 1: Create and write sample content to source file

With open(source\_file, "w") as f:

f.write("This is a Sample Text with Mixed CASE.")

# Step 2: Read, convert to uppercase, and write to destination file

With open(source\_file, "r") as src:

Content = src.read()

Uppercase\_content = content.upper()

With open(destination\_file, "w") as dest:

Dest.write(uppercase\_content)

Print(f"Contents copied from '{source\_file}' to '{destination\_file}' with lowercase

Converted to uppercase.")

Output

Contents copied from 'input.txt' to 'output.txt' with lowercase converted to uppercase.

Q6

# File names

File1 = "file1.txt"

File2 = "file2.txt"

Merged\_file = "merged.txt"

# Step 1: Create two sample files

With open(file1, "w") as f1:

F1.write("Line1 from File1\nLine2 from File1\nLine3 from File1\n")

With open(file2, "w") as f2:

F2.write("Line1 from File2\nLine2 from File2\n")

# Step 2: Read lines from both files

With open(file1, "r") as f1:

Lines1 = f1.readlines()

With open(file2, "r") as f2:

```
Lines2 = f2.readlines()
```

```
# Step 3: Merge lines alternatively
```

```
Merged_lines = []
```

```
Max_len = max(len(lines1), len(lines2))
```

```
For i in range(max_len):
```

```
    If i < len(lines1):
```

```
        Merged_lines.append(lines1[i])
```

```
    If i < len(lines2):
```

```
        Merged_lines.append(lines2[i])
```

```
# Step 4: Write merged content to new file
```

```
With open(merged_file, "w") as mf:
```

```
    Mf.writelines(merged_lines)
```

```
Print(f"Lines merged alternatively into '{merged_file}' successfully.")
```

Output

Lines merged alternatively into 'merged.txt' successfully.

Q7

Import pickle

# Define Employee class

Class Employee:

Def \_\_init\_\_(self, empcode, empname, doj, salary):

Self.empcode = empcode

Self.empname = empname

Self.doj = doj

Self.salary = salary

Def display(self):

Print("Employee Code:", self.empcode)

Print("Employee Name:", self.empname)

Print("Date of Joining:", self.doj)

Print("Salary:", self.salary)

```
# Create an Employee object
```

```
Emp = Employee("E001", "John Doe", "2022-04-01", 50000)
```

```
# Serialize (save to file)
```

```
With open("employee.dat", "wb") as f:
```

```
Pickle.dump(emp, f)
```

```
Print("Employee object serialized to 'employee.dat'.")
```

```
# Deserialize (load from file)
```

```
With open("employee.dat", "rb") as f:
```

```
Emp_loaded = pickle.load(f)
```

```
Print("\nDeserialized Employee object:")
```

```
Emp_loaded.display()
```

Output

Employee object serialized to 'employee.dat'.

Deserialized Employee object:

Employee Code: E001

Employee Name: John Doe

Date of Joining: 2022-04-01

Salary: 50000

Q8

Import re

```
# File names
```

```
Source_file = "original.txt"
```

```
Cleaned_file = "cleaned.txt"
```

```
# Step 1: Create a sample source file
```

```
With open(source_file, "w") as f:
```

```
    f.write("This is a sample sentence with an article and the definite article.")
```

```
# Step 2: Read and process the content
```

```
With open(source_file, "r") as f:
```

```
    Content = f.read()
```

```
# Replace 'a', 'an', 'the' with blank space using regex (word boundaries)
```

```
Cleaned_content = re.sub(r'\b(a|an|the)\b', "", content, flags=re.IGNORECASE)
```

```
# Optional: Remove extra spaces caused by word deletion
```

```
Cleaned_content = re.sub(r'\s+', ' ', cleaned_content).strip()
```

```
# Step 3: Write the cleaned content to a new file
```

```
With open(cleaned_file, "w") as f:
```

```
    f.write(cleaned_content)
```

```
print(f"Words 'a', 'an', 'the' removed and result saved in '{cleaned_file}'.")
```

Output

Words 'a', 'an', 'the' removed and result saved in 'cleaned.txt'.



