a) session 1 & 2

```
#1. accept empid,empname,monthly_salary,tot_deductions, tot_allowances and display employee name and salary in hand

# accepting employee details
emp_id = input("enter emp id: ")
emp_name = input("enter emp name: ")

monthly_salary = float(input("enter monthly salary: "))

tot_deductions = float(input("enter total deductions: "))

tot_allowances = float(input("enter total allowances: "))


# calculating salary in hand

salary_in_hand = monthly_salary + tot_allowances - tot_deductions

# displaying employee name and salary in hand

print(f"employee name: {emp_name}")

print(f"salary in hand: {salary_in_hand}")


#if conditions :

#1. accept 3 integers from the user and display maximum

num1 = int(input("enter the first integer: "))

num2 = int(input("enter the second integer: "))

num3 = int(input("enter the third integer: "))

if num1 >= num2 and num1 >= num3:

    maximum = num1

elif num2 >= num1 and num2 >= num3:

    maximum = num2

else:

    maximum = num3

print("the maximum value is:", maximum)


#2. accept 3 integers from user and display minimum

num1 = int(input("enter the first integer: "))

num2 = int(input("enter the second integer: "))

num3 = int(input("enter the third integer: "))

if num1 <= num2 and num1 <= num3:

    minimum = num1
```

```
elif num2 <= num1 and num2 <= num3:
    minimum = num2
else:
    minimum = num3
print("the minimum value is:", minimum)

#loops (solve without using functions if any)
#1. accept integers from user till users choice and do the following:
    #1. sum of all integers
    #2. average of all integers
    #3. maximum integer from all
    #4. minimum integer from all
a = int(input('enter the number of integers: '))
arr = []
for i in range(a):
  num = int(input('enter a number: '))
  arr.append(num)
print('sum of all integers={}'.format(sum(arr)))
print('average of all integers={}'.format(sum(arr)/a))
print('maximum integer from all={}'.format(max(arr)))
print('minimum integer from all={}'.format(min(arr)))


#2. accept a string from user an do the following :
    #1. find the length
    #2. display string in reverse
    #2. display every alternate character in upper case
    #3. find out no of vowels in the string
    #4. accept username and date of birth (dd-mon-yy) from user
    #create a password string which will be combination of
    #1st 4 letters of username and last 2digits of date of birth followed by $ sign
    #5. encrypt the string and return encrypted string
user_string = input("enter a string: ")

string_length = len(user_string)
```

```
print("length of the string: {}".format(string_length))

reversed_string = user_string[::-1]

print("reversed string: {}".format(reversed_string))

alternate_uppercase = ""

for i in range(len(user_string)):

    if i % 2 == 0:

        alternate_uppercase += user_string[i].upper()

    else:

        alternate_uppercase += user_string[i]

print("alternate characters in uppercase: {}".format(alternate_uppercase))

vowels = "aeiouaeiou"

vowel_count = 0

for char in user_string:

    if char in vowels:

        vowel_count += 1

print("number of vowels in the string: {}".format(vowel_count))

username = input("enter your username: ")

dob = input("enter your date of birth (dd-mon-yy): ")

password = username[:4] + dob[-2:] + "$"

print("generated password: {}".format(password))

encrypted_password = ""

for char in password:

    encrypted_password += chr(ord(char) + 3)  # shift each character by 3 in ascii

print("encrypted password: {}".format(encrypted_password))


#3. write python program to do the following :

    #1. display area of circle

    #parallelogram

# accepting input for circle

radius = float(input("enter the radius of the circle: "))

area_circle = 3.14 * (radius ** 2)

print(f"area of the circle: {area_circle}")
```

```
# accepting input for parallelogram
base = float(input("enter the base of the parallelogram: "))
height = float(input("enter the height of the parallelogram: "))
area_parallelogram = base * height
print(f"area of the parallelogram: {area_parallelogram}")

#4. accept integer and find square root of integer
number = int(input("enter an integer: "))
if number < 0:
    print("the square root is complex")
elif number==0:
    print("the square root of 0 is 0.")
else:
    approx_sqrt = number ** 0.5
    print(f"the square root of {number} is approximately {approx_sqrt}")
```

Session 3 / 4

List / Tuples / Dictionary / Sets

```
#1. Create a List for the following :
    #a. Accept Fruits Name and their price(per kg)
    #b. Fruits Name should be at odd index position in the List.Price at even index position
fruits_list = []
n = int(input("Enter the number of fruits: "))
for i in range(n):
    fruit_name = input(f"Enter the name of fruit {i+1}: ")
    fruit_price = float(input(f"Enter the price of {fruit_name} per kg: "))
    fruits_list.append(fruit_price)
    fruits_list.append(fruit_name)
print("Fruits and their prices are):")
print(fruits_list)

#2. Customer will buy fruits from you (Show him the Fruits Menu)
#Write a Program to
```

```
    #a. Calculate Total Price of Fruits Bought .
    #b. Add New Fruits in the List
    #c. Show Total Fruits in the List
menu = {"Strawberry": 300.0,"Pineapple": 250.0,"Blueberry": 350.0,"Kiwi": 220.0,"Watermelon": 180.0}
# Display menu
print("Fruit Menu:")
for fruit, price in menu.items():
    print(f"{fruit}: ${price:.2f} per kg")
# Customer purchases
total_price = 0.0
while True:
    fruit = input("Enter fruit to buy (or 'done' to finish): ").capitalize()
    if fruit.lower() == 'done':
        break
    if fruit in menu:
        quantity = float(input(f"Quantity (kg) of {fruit}: "))
        total_price += menu[fruit] * quantity
    else:
        print("Not available.")
# Total price
print(f"Total price: ${total_price:.2f}")
# Add new fruit
if input("Add a new fruit to the menu? (yes/no): ").lower() == 'yes':
    new_fruit = input("New fruit name: ")
    menu[new_fruit] = float(input(f"Price per kg of {new_fruit}: "))
    print(f"{new_fruit} added.")
# Total fruits in menu
print(f"Total fruits in the menu: {len(menu)}")

# 3. Create Foll. Information in the Tuple (atleast 5 Employees)
# 1. EmpId - Phone Numbers (One Employee can have Multiple Numbers )
# 2. Accept Empid from User.
# Display his Numbers only if he exists in the Database(Tuple)
# Display App. Message if not present
# 3. Update Employee phone Number
# Accept Empid from User
# Check whether he / she Exists
# Accept New Phone Number
# Update
# Display Appropriate Message for any task
employees = ([101,"+91 9876543210","+91 9876543211"],
        [102,"+91 9123456789","+91 9123456790"],
        [103,"+91 9234567890","+91 9234567891"],
        [104,"+91 9345678901","+91 9345678902"],
        [105,"+91 9456789012","+91 9456789013"])
# Accept EmpId from user and display their phone numbers
emp_id = input("\nEnter Employee ID to view phone numbers: ")
found = False
for emp in employees:
  if emp[0] == emp_id:
    print(f"Phone Numbers for {emp_id}: {', '.join(emp[1])}")
    found = True
    break
if not found:
  print("Employee ID not found in the database.")
# Update Employee phone number
update_id = input("\nEnter Employee ID to update phone number: ")
found = False
for emp in employees:
  if emp[0] == update_id:
    new_phone = input("Enter the new phone number: ")
    emp[1].append(new_phone)  # Update phone numbers
    print(f"Updated phone numbers for {update_id}: {', '.join(emp[1])}")
    found = True
    break
if not found:
  print("Employee ID not found for update.")
```

```
# 4. Store the Following info in Dictionary
# Department Name and their Employee Names
# Note : One Department can have multiple Employees
departments = {
    "CSE": ["Priya", "Ravi"],
    "IT": ["Anil", "Sita"],
    "ECE": ["Amit", "Rohini"]
}

# 1. Add a new department and employees if the department doesn't exist
dept_name = input("Enter department to add: ")
if dept_name not in departments:
    employees = input(f"Enter employees for {dept_name} (comma separated): ").split(", ")
    departments[dept_name] = employees
    print(f"Department {dept_name} added with employees: {', '.join(employees)}")
else:
    print(f"Department {dept_name} already exists.")
# 2. Accept department name and list all employees if the department exists
dept_name = input("\nEnter department to list employees: ")
if dept_name in departments:
    print(f"Employees in {dept_name}: {', '.join(departments[dept_name])}")
else:
    print(f"Department {dept_name} does not exist.")
# 3. Add a new employee to an existing department
dept_name = input("\nEnter department to add an employee: ")
if dept_name in departments:
    new_employee = input(f"Enter new employee for {dept_name}: ")
    departments[dept_name].append(new_employee)
    print(f"{new_employee} added to {dept_name}.")
else:
    print(f"Department {dept_name} does not exist.")
```

```
# 4. Delete an existing employee from a department

dept_name = input("\nEnter department to delete an employee from: ")

if dept_name in departments:

    employee_to_remove = input(f"Enter employee to remove from {dept_name}: ")

    if employee_to_remove in departments[dept_name]:

        departments[dept_name].remove(employee_to_remove)

        print(f"{employee_to_remove} removed from {dept_name}.")

    else:

        print(f"{employee_to_remove} not found in {dept_name}.")

else:

    print(f"Department {dept_name} does not exist.")


# 5. Create Following two Sets

# 1. Fruit_Salesman1

# 2. Fruit_Salesman2

# Create Fruits for both SalesmaN

# Sets for fruits sold by each salesman

Fruit_Salesman1 = {"Mango", "Guava", "Papaya", "Litchi"}

Fruit_Salesman2 = {"Papaya", "Banana", "Mango", "Chikoo"}

# 1. Find out common fruits

common_fruits = Fruit_Salesman1.intersection(Fruit_Salesman2)

print(f"Common fruits: {common_fruits}")

# 2. List extra fruits with both salesmen

extra_fruits_salesman1 = Fruit_Salesman1.difference(Fruit_Salesman2)

extra_fruits_salesman2 = Fruit_Salesman2.difference(Fruit_Salesman1)

print(f"Extra fruits with Salesman 1: {extra_fruits_salesman1}")

print(f"Extra fruits with Salesman 2: {extra_fruits_salesman2}")

# 3. List total fruits with both salesmen

total_fruits = Fruit_Salesman1.union(Fruit_Salesman2)

print(f"Total fruits with both Salesmen: {total_fruits}")
```