#A) Session 1 & 2

#Input/ Output#1. Accept Empid,EmpName,Monthly_Salary,Tot_Deductions, Tot_Allowances and Display Employee Name and Salary in hand

```
empid = input("Enter Employee ID: ")
emp_name = input("Enter Employee Name: ")
monthly_salary = float(input("Enter Monthly Salary: "))
tot_deductions = float(input("Enter Total Deductions: "))
tot_allowances = float(input("Enter Total Allowances: "))

salary_in_hand = monthly_salary - tot_deductions + tot_allowances
print(f"Employee Name: {emp_name}, Salary in hand: {salary_in_hand:.2f}")
```

#if Conditions :

#1. Accept 3 integers from the User and Display Maximum

```
a = int(input("Enter first integer: "))
b = int(input("Enter second integer: "))
c = int(input("Enter third integer: "))
maximum = a
if b > maximum:
    maximum = b
if c > maximum:
    maximum = c
print(f"The maximum of the three numbers is: {maximum}")
```

#2. Accept 3 integers from USer and display Minimum

```
a = int(input("Enter first integer: "))
b = int(input("Enter second integer: "))
```

```
c = int(input("Enter third integer: "))
minimum = a
if b < minimum:
    minimum = b
if c < minimum:
    minimum = c
print(f"The minimum of the three numbers 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
"""
integers = []
while True:
    num = int(input("Enter an integer (or type '0' to stop): "))
    if num == 0:
        break
    integers.append(num)

sum_of_integers = sum(integers)
average_of_integers = sum_of_integers / len(integers)
max_integer = max(integers)
min_integer = min(integers)

print(f"Sum: {sum_of_integers}")
print(f"Average: {average_of_integers}")
print(f"Maximum: {max_integer}")
print(f"Minimum: {min_integer}")
```

```
# Accept a string from the user
user_string = input("Enter a string: ")

# 1. Find the length of the string
length_of_string = len(user_string)
print(f"Length of the string: {length_of_string}")

# 2. Display the string in reverse
reverse_string = user_string[::-1]
print(f"Reversed string: {reverse_string}")

# 3. Display every alternate character in upper case
alternate_upper = ''.join([char.upper() if i % 2 == 0 else char for i, char in enumerate(user_string)])
print(f"String with alternate characters in upper case: {alternate_upper}")

# 4. Find the number of vowels in the string
vowels = "aeiouAEIOU"
vowel_count = sum(1 for char in user_string if char in vowels)
print(f"Number of vowels in the string: {vowel_count}")

# 5. Accept username and date of birth from the user
username = input("Enter your username: ")
dob = input("Enter your date of birth (dd-mon-yy): ")

# Create password by combining the first 4 letters of the username,
# the last 2 digits of the birth year, and a '$' sign
password = username[:4] + dob[-2:] + "$"
print(f"Generated password: {password}")

# 6. Simple encryption function (Caesar Cipher with shift of 3)
```

```
password = input("Enter the password to encrypt: ")
encrypted_password = ""
# Perform encryption (shift by 3)
for char in password:
    encrypted_char = chr(ord(char) + 3)
    encrypted_password += encrypted_char
# Display the encrypted password
print("Encrypted password:", encrypted_password)

#3 write a python program to do the following:
import math

  # 1. Area of a Circle
radius = float(input("Enter the radius of the circle: "))
area_circle = math.pi * radius * radius
print(f"Area of the circle: {area_circle:.2f}")

  # 2. Area of a 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:.2f}")

#4. Accept Integer and find Square root of Integer

import math
number = int(input("Enter a positive integer: "))
if number < 0:
    print("Error: Cannot compute the square root of a negative number.")
else:
    square_root = math.sqrt(number)
```

```
    print(f"The square root of {number} is: {square_root:.2f}")



#B) 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
"""
fruit_price_list = []
num_fruits = int(input("How many fruits do you want to enter? "))
for i in range(num_fruits):
    fruit_name = input(f"Enter the name of fruit {i + 1}: ")
    price = float(input(f"Enter the price per kg of {fruit_name}: "))

    # Append price to the list (even index)
    fruit_price_list.append(price)
    # Append fruit name to the list (odd index)
    fruit_price_list.append(fruit_name)



print("Fruits and their prices (per kg):")
print(fruit_price_list)

# Displaying the formatted output
for index in range(len(fruit_price_list)):
    if index % 2 == 0:
        print(f"Price: {fruit_price_list[index]:.2f} per kg")
    else:
```

```
        print(f"Fruit: {fruit_price_list[index]}")
"""
2. Customer will buy fruits from you (Show him the Fruits Menu)
Write a Program to
a. Calculate Total Price of Fruits Bought .
(Assume price for 1 kg )
b. Add New Fruits in the List
c. Show Total Fruits in the List
"""
fruit_menu = {
    "Apple": 120.0,
    "Banana": 80.0,
    "Cherry": 200.0,
    "Mango": 150.0,
    "Orange": 100.0
}

# Display menu
print("Fruit Menu:")
for fruit, price in fruit_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 fruit_menu:
        quantity = float(input(f"Quantity (kg) of {fruit}: "))
        total_price += fruit_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: ")
    fruit_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(fruit_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
"""

# Employee data as a tuple (EmpId, [Phone Numbers])
employee_data = (
```

```
    (101, ["123-456-7890", "098-765-4321"]),
    (102, ["234-567-8901"]),
    (103, ["345-678-9012", "876-543-2109"]),
    (104, ["456-789-0123"]),
    (105, ["567-890-1234", "678-901-2345"])
)

# Convert tuple to dictionary for easier lookup and update
employee_dict = {emp[0]: emp[1] for emp in employee_data}

# 1. Accept EmpId from User and display phone numbers
emp_id = int(input("Enter Employee ID to view phone numbers: "))
if emp_id in employee_dict:
    print(f"Phone numbers for Employee ID {emp_id}: {', '.join(employee_dict[emp_id])}")
else:
    print("Employee ID not found in the database.")

# 2. Update Employee Phone Number
update_id = int(input("Enter Employee ID to update phone number: "))
if update_id in employee_dict:
    new_number = input("Enter the new phone number: ")
    employee_dict[update_id].append(new_number)  # Add the new number to the list
    print(f"Phone number updated for Employee ID {update_id}. Current numbers: {',
'.join(employee_dict[update_id])}")
else:
    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
```

```
Perform the Following Operations :
1. Add a New Department Name and Employees in that Department
If a New Department Name doesnot Exists
2. Accept Dept Name from User and List all Employees
If Dept Name Exists in the Database
3. Add a New Employee in Existing Department
4. Delete Existing Employee From Department
"""

# Initialize the department dictionary with sample data
departments = {
    "HR": ["Alice", "Bob"],
    "IT": ["Charlie", "David"],
    "Finance": ["Eva", "Frank"],
    "Marketing": ["Grace", "Hank"]
}

# 1. Add a new department and employees
dept_name = input("Enter a new department name: ")
if dept_name not in departments:
    employees = input(f"Enter employee names for {dept_name} (comma separated): ").split(',')
    departments[dept_name] = [emp.strip() for emp in employees]
    print(f"Department '{dept_name}' added.")
else:
    print(f"Department '{dept_name}' already exists.")

# 2. List all employees in a given department
dept_name = input("Enter department name to list employees: ")
if dept_name in departments:
    print(f"Employees in {dept_name}: {', '.join(departments[dept_name])}")
```

```
else:
    print(f"Department '{dept_name}' not found.")

# 3. Add a new employee in an existing department
dept_name = input("Enter existing department name to add an employee: ")
if dept_name in departments:
    new_employee = input("Enter the name of the new employee: ")
    departments[dept_name].append(new_employee)
    print(f"Employee '{new_employee}' added to {dept_name}.")
else:
    print(f"Department '{dept_name}' not found.")

# 4. Delete an existing employee from a department
dept_name = input("Enter department name to delete an employee: ")
if dept_name in departments:
    employee_to_delete = input("Enter the name of the employee to delete: ")
    if employee_to_delete in departments[dept_name]:
        departments[dept_name].remove(employee_to_delete)
        print(f"Employee '{employee_to_delete}' removed from {dept_name}.")
    else:
        print(f"Employee '{employee_to_delete}' not found in {dept_name}.")
else:
    print(f"Department '{dept_name}' not found.")

"""
5. Create Following two Sets
1. Fruit_Salesman1
2. Fruit_Salesman2
Create Fruits for both Salesmans
Perform the Following Operations
1. Find out Common Fruits with both Salesman
```

2. List Extra Fruits with Both Salesman

3. List Total Fruits with both Salesman

"""

```
# Create sets for fruit salesmen
fruit_salesman1 = {"Apple", "Banana", "Cherry", "Mango"}
fruit_salesman2 = {"Banana", "Mango", "Orange", "Grapes"}

# 1. Find out common fruits
common_fruits = fruit_salesman1.intersection(fruit_salesman2)
print("Common fruits:", common_fruits)

# 2. List extra fruits for both salesmen
extra_fruits_salesman1 = fruit_salesman1 - fruit_salesman2
extra_fruits_salesman2 = fruit_salesman2 - fruit_salesman1
print("Extra fruits with Salesman 1:", extra_fruits_salesman1)
print("Extra fruits with Salesman 2:", extra_fruits_salesman2)

# 3. List total unique fruits from both salesmen
total_fruits = fruit_salesman1.union(fruit_salesman2)
print("Total unique fruits from both salesmen:", total_fruits)
```