ASSIGNMENT 1-PYTHON

Name: P217 KATIKA MUBEENA

A) Session 1 & 2

Input / Output:

```python
# Input Employee Details

emp_id = 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: "))


# Calculate Salary in Hand

salary_in_hand = monthly_salary - tot_deductions + tot_allowances


# Display Result

print(f"Employee Name: {emp_name}")

print(f"Salary in Hand: {salary_in_hand}")
```

output:

Enter Employee ID: 17

Enter Employee Name: ram

Enter Monthly Salary: 30000

Enter Total Deductions: 2000

Enter Total Allowances: 10000

Employee Name: ram

Salary in Hand: 38000.0

if Conditions :

```python
# Input 3 Integers
num1 = int(input("Enter first integer: "))
num2 = int(input("Enter second integer: "))
num3 = int(input("Enter third integer: "))


# Find Maximum
if num1 >= num2 and num1 >= num3:
    max_num = num1
elif num2 >= num1 and num2 >= num3:
    max_num = num2
else:
    max_num = num3


# Display Result
print(f"The maximum number is: {max_num}")


# Find Minimum
if num1 <= num2 and num1 <= num3:
    min_num = num1
elif num2 <= num1 and num2 <= num3:
    min_num = num2
else:
    min_num = num3


# Display Result
print(f"The minimum number is: {min_num}")
```

output:

Enter first integer: 2

Enter second integer: 4

Enter third integer: 5

The maximum number is: 5

The minimum number is: 2

---

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

```python
# Initialize list to store integers and variables for calculations
numbers = []
total_sum = 0
max_num = None
min_num = None
count = 0

# Loop until user chooses to stop
while True:
    num = int(input("Enter an integer: "))
    numbers.append(num)
    total_sum += num
    count += 1

    # Set initial max and min
```

```python
    if max_num is None or num > max_num:

        max_num = num

    if min_num is None or num < min_num:

        min_num = num


    choice = input("Do you want to add another number? (yes/no): ")

    if choice.lower() != 'yes':

        break


# Calculate Average

average = total_sum / count


# Display Results

print(f"Sum of all integers: {total_sum}")

print(f"Average of all integers: {average}")

print(f"Maximum integer: {max_num}")

print(f"Minimum integer: {min_num}")
```

output:

Enter an integer: 3

Do you want to add another number? (yes/no): yes

Enter an integer: 4

Do you want to add another number? (yes/no): 6

Sum of all integers: 7

Average of all integers: 3.5

Maximum integer: 4

Minimum integer: 3

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

```python
# Accept a String
user_string = input("Enter a string: ")


# Find Length of String
length = 0
for _ in user_string:
    length += 1
print(f"Length of the string: {length}")


# Display String in Reverse
reversed_string = user_string[::-1]
print(f"Reversed string: {reversed_string}")


# Display Every Alternate Character in Upper Case
alt_upper = ""
for i in range(length):
    if i % 2 == 0:
        alt_upper += user_string[i].upper()
    else:
```

```python
        alt_upper += user_string[i]
print(f"Alternate characters in uppercase: {alt_upper}")


# Count Number of Vowels

vowels = 'aeiouAEIOU'

num_vowels = 0

for char in user_string:

    if char in vowels:

        num_vowels += 1

print(f"Number of vowels: {num_vowels}")


# Create Password

username = input("Enter username: ")

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

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

print(f"Generated password: {password}")


# Encrypt the String (Simple Shift Cipher)

encrypted_string = ""

shift = 3

for char in user_string:

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

        encrypted_char = chr((ord(char) - ord('a') + shift) % 26 + ord('a'))

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

        encrypted_char = chr((ord(char) - ord('A') + shift) % 26 + ord('A'))

    else:

        encrypted_char = char

    encrypted_string += encrypted_char

print(f"Encrypted string: {encrypted_string}")
```

output:

Enter a string: raazi

Length of the string: 5

Reversed string: izaar

Alternate characters in uppercase: RaAzI

Number of vowels: 3

Enter username: hello

Enter date of birth (dd-mon-yy): 24-03-2003

Generated password: hell03$

Encrypted string: uddcl

---

3. Write Python Program to do the following :

1. Display Area of

Circle

Parallelogram

```
# Area of a Circle
radius = float(input("Enter the radius of the circle: "))
area_circle = 3.14159 * radius * radius
print(f"Area of the circle: {area_circle}")
```

```
# 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}")
```

output:

Enter the radius of the circle: 5

Area of the circle: 78.53975

Enter the base of the parallelogram: 8

Enter the height of the parallelogram: 4

Area of the parallelogram: 32.0

---

4. Accept Integer and find Square root of Integer

```python
number = int(input("Enter a number to find its square root: "))
guess = number / 2  # Start with an initial guess

# Use the Newton-Raphson method for better approximation
for _ in range(20):  # Loop enough times for precision
    guess = (guess + number / guess) / 2

print(f"Approximate square root of {number} is: {guess}")
```

output:

Enter a number to find its square root: 9

Approximate square root of 9 is: 3.0

B) Session 3 & 4

#1. Create a List for Fruits and Prices
Create a list for fruits and their prices, fruits at odd index and prices at even index

```python
fruits_prices = []

# Accept fruit names and prices from user

for i in range(3):  # Assume the user will enter 3 fruits
    fruit = input(f"Enter the name of fruit {i+1}: ")
    price = float(input(f"Enter the price per kg of {fruit}: "))
    fruits_prices.append(fruit)
    fruits_prices.append(price)
```

```python
# Display the fruits menu

print("\nFruits Menu:")
for i in range(0, len(fruits_prices), 2):
    print(f"{fruits_prices[i]}: Rs {fruits_prices[i+1]}/kg")

#2. Calculate Total Price of Fruits Bought
# Customer buys fruits

total_price = 0
while True:
    fruit_name = input("Enter the name of the fruit to buy (or 'exit' to stop): ")
    if fruit_name.lower() == 'exit':
        break
    quantity = float(input(f"Enter the quantity (in kg) for {fruit_name}: "))

    # Find fruit in the list and calculate total price

    if fruit_name in fruits_prices:
        index = fruits_prices.index(fruit_name)
        price_per_kg = fruits_prices[index + 1]
        total_price += price_per_kg * quantity
    else:
        print(f"{fruit_name} is not available.")

print(f"\nTotal price of fruits bought: Rs {total_price}")

#3. Tuple for Employee Information

# Create a tuple for employee information (EmpId - Phone Numbers)

employees = (
    (101, ["9876543210", "9123456789"]),
    (102, ["9870011223"]),
    (103, ["8796543210", "8907654321"]),
    (104, ["9812345678"]),
    (105, ["7896541230", "7891234567"])
)

# Display employee phone numbers if they exist

emp_id = int(input("Enter Employee ID to display their phone numbers: "))
found = False
for emp in employees:
    if emp[0] == emp_id:
        print(f"Phone numbers of Employee {emp_id}: {', '.join(emp[1])}")
        found = True
        break

if not found:
    print(f"Employee with ID {emp_id} not found.")
```

```python
#4. Store Information in Dictionary (Department - Employees)
# Dictionary storing department names and employee names

department_employees = {
    "HR": ["Alice", "Bob"],
    "IT": ["Charlie", "Dave"],
    "Finance": ["Eve", "Frank"]
}


# Add new department and employees

dept_name = input("Enter new department name: ")
if dept_name not in department_employees:
    employees = input(f"Enter employees for {dept_name} (comma-separated): ").split(',')
    department_employees[dept_name] = [emp.strip() for emp in employees]
    print(f"{dept_name} department added with employees: {',
'.join(department_employees[dept_name])}")
else:
    print(f"{dept_name} already exists.")

# List employees in a department

dept_name = input("\nEnter department name to list employees: ")
if dept_name in department_employees:
    print(f"Employees in {dept_name}: {', '.join(department_employees[dept_name])}")
else:
    print(f"{dept_name} not found.")

# Add a new employee to existing department
dept_name = input("\nEnter department name to add an employee: ")
if dept_name in department_employees:
    new_employee = input("Enter the name of new employee: ")
    department_employees[dept_name].append(new_employee)
    print(f"{new_employee} added to {dept_name}.")
else:
    print(f"{dept_name} does not exist.")

# Delete an employee from department

dept_name = input("\nEnter department name to remove an employee: ")
if dept_name in department_employees:
    emp_name = input("Enter employee name to delete: ")
    if emp_name in department_employees[dept_name]:
        department_employees[dept_name].remove(emp_name)
        print(f"{emp_name} removed from {dept_name}.")
    else:
        print(f"{emp_name} not found in {dept_name}.")
else:
    print(f"{dept_name} does not exist.")
```

#5. Set Operations for Fruit Salesman
Create sets for two fruit salesmen

```
salesman1_fruits = {"apple", "banana", "mango", "grapes"}
salesman2_fruits = {"banana", "orange", "mango", "kiwi"}

# Common fruits between both salesmen

common_fruits = salesman1_fruits.intersection(salesman2_fruits)
print(f"\nCommon fruits sold by both: {common_fruits}")

# Extra fruits sold by each salesman

extra_salesman1 = salesman1_fruits.difference(salesman2_fruits)
extra_salesman2 = salesman2_fruits.difference(salesman1_fruits)
print(f"Extra fruits sold by salesman1: {extra_salesman1}")
print(f"Extra fruits sold by salesman2: {extra_salesman2}")

# Total unique fruits sold by both

total_fruits = salesman1_fruits.union(salesman2_fruits)
print(f"Total fruits sold by both: {total_fruits}")
```

OUTPUT1

Enter the name of fruit 1: Mango
Enter the price per kg of Mango: 100
Enter the name of fruit 2: Orange
Enter the price per kg of Orange: 70
Enter the name of fruit 3: Apple
Enter the price per kg of Apple: 90

Fruits Menu:
Mango: Rs 100.0/kg
Orange: Rs 70.0/kg
Apple: Rs 90.0/kg
Enter the name of the fruit to buy (or 'exit' to stop): Mango
Enter the quantity (in kg) for Mango: 7
Enter the name of the fruit to buy (or 'exit' to stop): exit

Total price of fruits bought: Rs 700.0
Create sets for two fruit salesmen

Common fruits sold by both: {'banana', 'mango'}
Extra fruits sold by salesman1: {'grapes', 'apple'}
Extra fruits sold by salesman2: {'orange', 'kiwi'}
Total fruits sold by both: {'grapes', 'orange', 'kiwi', 'apple', 'banana', 'mango'}

OUTPUT2

Enter new department name: IT
IT already exists.

Enter department name to list employees: HR
Employees in HR: Alice, Bob

Enter department name to add an employee: IT
Enter the name of new employee: MUBEENA
MUBEENA added to IT.

Enter department name to remove an employee: HR
Enter employee name to delete: DIVYA
DIVYA not found in HR.