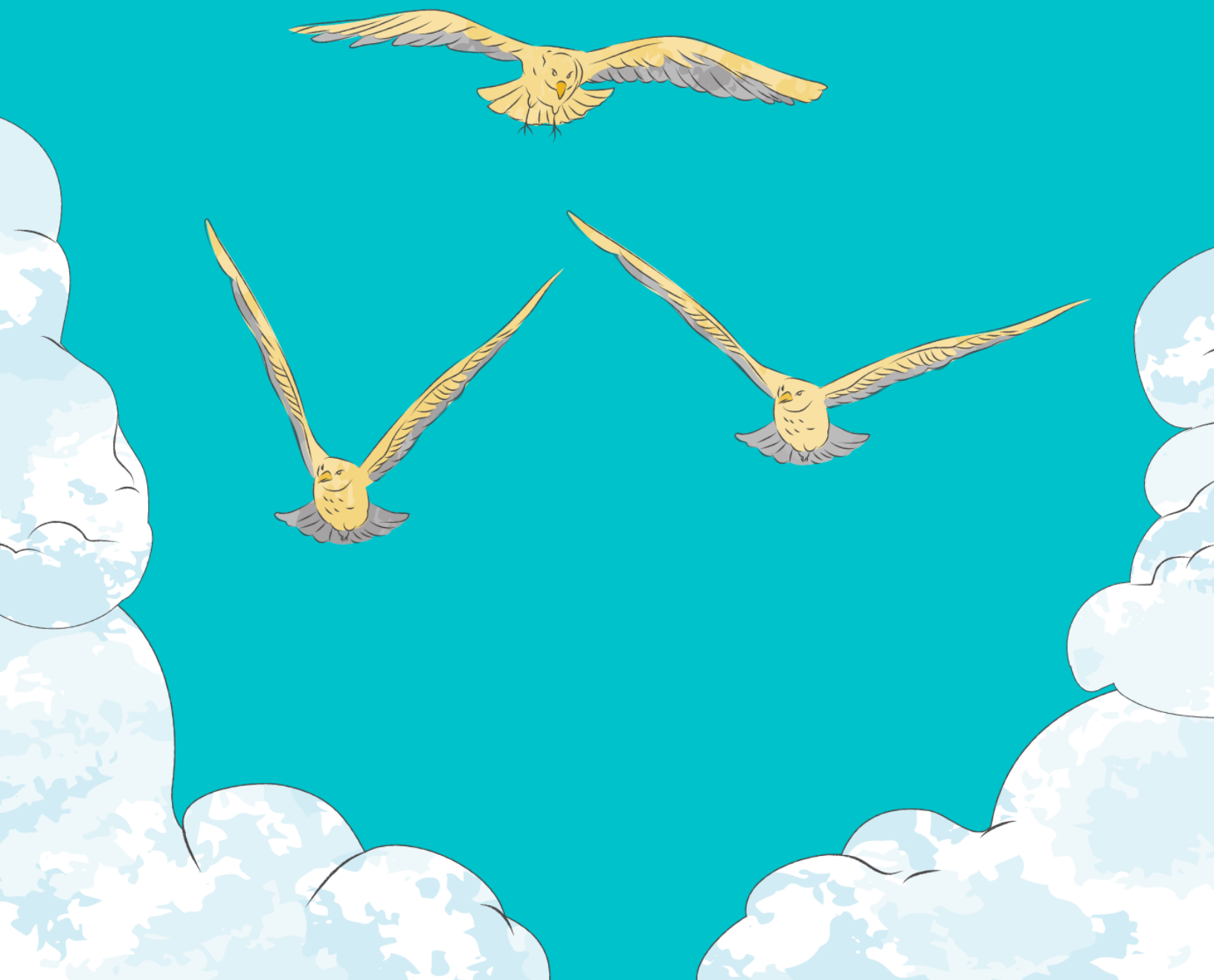
ALL IMP PROGRAMS

PPS SEM-1

by Mohd Yasir Sheikh



Q) Algorithm and flowchart to add natural numbers in python

Algorithm:

Start:

1. Initialise variables: sum to 0, counter to 1.
2. Input the value of n (the last natural number to be added).
3. Repeat Until counter is greater than n:
4. Add counter to sum.
5. Increment counter by 1.
6. Display the value of sum.

Stop.

Python Code:

```
# Step 1: Start
# Step 2: Initialize variables
sum_result = 0
counter = 1

# Step 3: Input the value of n
n = int(input("Enter the value of n: "))

# Step 4: Repeat until counter is greater than n
while counter <= n:
    # Step 4a: Add counter to sum
    sum_result += counter

    # Step 4b: Increment counter
    counter += 1

# Step 5: Display the result
print(f"The sum of the first {n} natural numbers is: {sum_result}")

# Step 6: Stop
```

Q)Flowchart & algorithm to find whether person is eligible to vote or not

Algorithm:

1. Start:
2. Input the age of the person.
3. Check if the age is greater than or equal to the minimum voting age.
4. If true, output "Person is eligible to vote."
5. If false, output "Person is not eligible to vote."
6. Stop.

Q) WAP to access the marks of the students and find the sum and % of marks obtained by the student.

```python
# Function to calculate sum and percentage of marks
def calculate_marks():
    # Input the number of subjects
    num_subjects = int(input("Enter the number of subjects: "))

    # Initialize variables
    total_marks = 0

    # Input marks for each subject
    for i in range(1, num_subjects + 1):
        subject_marks = float(input(f"Enter marks for Subject {i}: "))

        # Add marks to total
        total_marks += subject_marks

    # Calculate percentage
    percentage = (total_marks / (num_subjects * 100)) * 100

    # Display results
    print("\nResults:")
    print(f"Total Marks: {total_marks}")
    print(f"Percentage: {percentage:.2f}%")

# Call the function
calculate_marks()
```

Q) Program to generate the different patterns

Example 1: Square Pattern

```
def square_pattern(rows):
    for i in range(rows):
        print('* ' * rows)

# Example usage with a square of size 5
square_pattern(5)
```

Example 2: Right Triangle Pattern

```
def right_triangle_pattern(rows):
    for i in range(1, rows + 1):
        print('* ' * i)

# Example usage with a right triangle of height 5
right_triangle_pattern(5)
```

Example 3: Pyramid Pattern

```
def pyramid_pattern(rows):
    for i in range(1, rows + 1):
        print(' ' * (rows - i) + '* ' * i)

# Example usage with a pyramid of height 5
pyramid_pattern(5)
```

Example 4: Inverted Pyramid Pattern

```
def inverted_pyramid_pattern(rows):
    for i in range(rows, 0, -1):
        print(' ' * (rows - i) + '* ' * i)

# Example usage with an inverted pyramid of height 5
inverted_pyramid_pattern(5)
```

Q) WAP to show reverse of a given number using while Loop

```
def reverse_number():
    # Input a number
    num = int(input("Enter a number: "))

    # Initialize variables
    reversed_num = 0

    # Reverse the number using a while loop
    while num > 0:
        digit = num % 10
        reversed_num = reversed_num * 10 + digit
        num = num // 10

    # Display the reversed number
    print(f"Reverse of the given number: {reversed_num}")

# Call the function
reverse_number()
```

Q) Snippets on List indexing & slicing

List Indexing:

1. Accessing an Element by Index:

```
my_list = [10, 20, 30, 40, 50]
element = my_list[2]  # Access the element at index 2 (zero-based)
print(element)
```

2. Negative Indexing:

```
my_list = [10, 20, 30, 40, 50]
last_element = my_list[-1]  # Access the last element using negative
print(last_element)
```

List Slicing:

1. Basic Slicing:

```
my_list = [10, 20, 30, 40, 50]
sublist = my_list[1:4]  # Slice elements from index 1 to 3 (exclusive)
print(sublist)
```

2. Omitting Start or End Index:

```
my_list = [10, 20, 30, 40, 50]
sublist1 = my_list[:3]  # Slice from the beginning to index 2
sublist2 = my_list[2:]  # Slice from index 2 to the end
print(sublist1)
print(sublist2)
```

3. Slicing with a Step:

```
my_list = [10, 20, 30, 40, 50]
step_slice = my_list[1:5:2]  # Slice elements from index 1 to 4 with
print(step_slice)
```

4. Reverse a List using Slicing:

```
my_list = [10, 20, 30, 40, 50]
reversed_list = my_list[::-1]  # Reverse the entire list
print(reversed_list)
```

Q) Addition and subtraction using function

Addition Function:

```
def add_numbers(a, b):
    """
    Adds two numbers and returns the result.
    """
    result = a + b
    return result

# Example usage:
num1 = 5
num2 = 3
sum_result = add_numbers(num1, num2)
print(f"The sum of {num1} and {num2} is: {sum_result}")
```

Subtraction Function:

```python
def subtract_numbers(a, b):
    """
    Subtracts the second number from the first and returns the result.
    """
    result = a - b
    return result

# Example usage:
num1 = 8
num2 = 3
difference_result = subtract_numbers(num1, num2)
print(f"The difference between {num1} and {num2} is: {difference_result
```

Q) Factorial of a number using recursion

```python
def factorial(n):
    """
    Calculates the factorial of a number using recursion.
    """
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Example usage:
number = int(input("Enter a number to calculate its factorial: "))
result = factorial(number)
print(f"The factorial of {number} is: {result}")
```