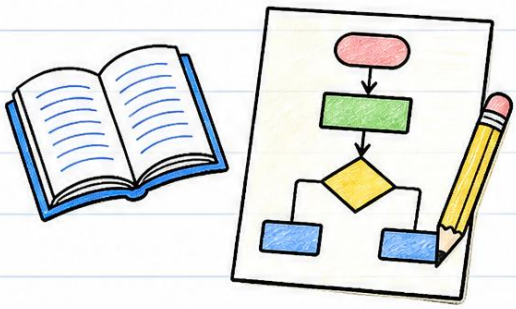
# S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

## Unit 2: Algorithms and Flowcharts to Solve Problems

### Flowchart Symbols and Processing Types

Flowchart किसी Algorithm या process का graphical representation हैं। Standard symbols और arrows की सहायता से problem-solving steps दिखाए जाते हैं।

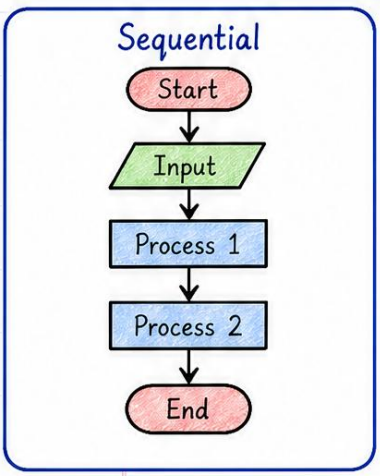


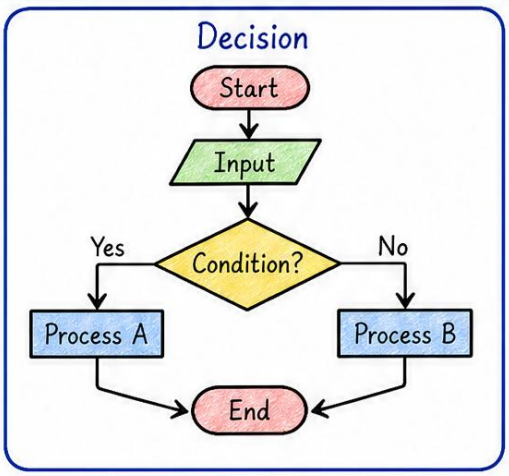
**मुख्य Flowchart Symbols:**

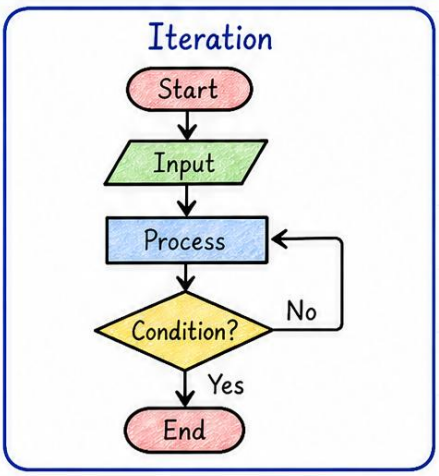
- **Oval:** Start या End
- **Parallelogram:** Input या Output
- **Rectangle:** Processing या calculation
- **Diamond:** Decision या condition
- **Arrow:** Control flow की दिशा

**Processing के प्रकार:**

1. **Sequential Processing:** Instructions एक निश्चित क्रम में एक-एक करके execute होते हैं।
2. **Decision-Based Processing:** Condition के आधार पर अलग-अलग path चुना जाता हैं।
3. **Iterative Processing:** किसी step या group of steps को बार-बार repeat किया जाता हैं।

# S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

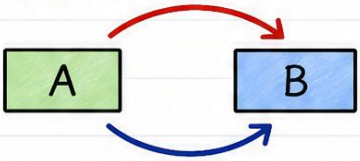
## Unit 2: Algorithms and Flowcharts to Solve Problems

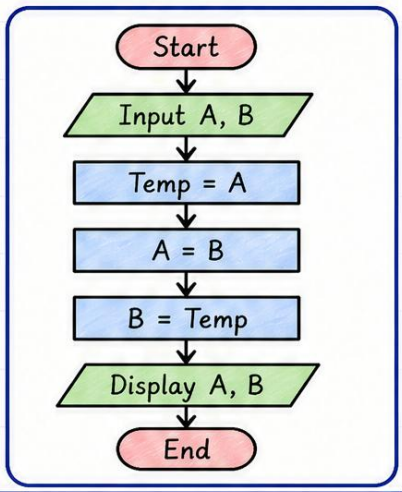
### Sequential Processing: Basic Algorithms

Sequential Processing एक ऐसी प्रक्रिया हैं जिसमें instructions को निश्चित क्रम में, एक-एक करके execute किया जाता है। इसमें हर step तभी पूरा होता है जब उससे पहले वाला step पूरा हो जाता है।

**उदाहरण 1:** दो चरों के मानों का आदान-प्रदान करना (Exchanging values of two variables)

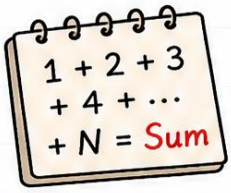
1. Input: दो variables A और B के मान पढ़िए।
2. Temp = A करें।
3. A = B करें।
4. B = Temp करें।
5. A और B के मान display कीजिए।
6. End।

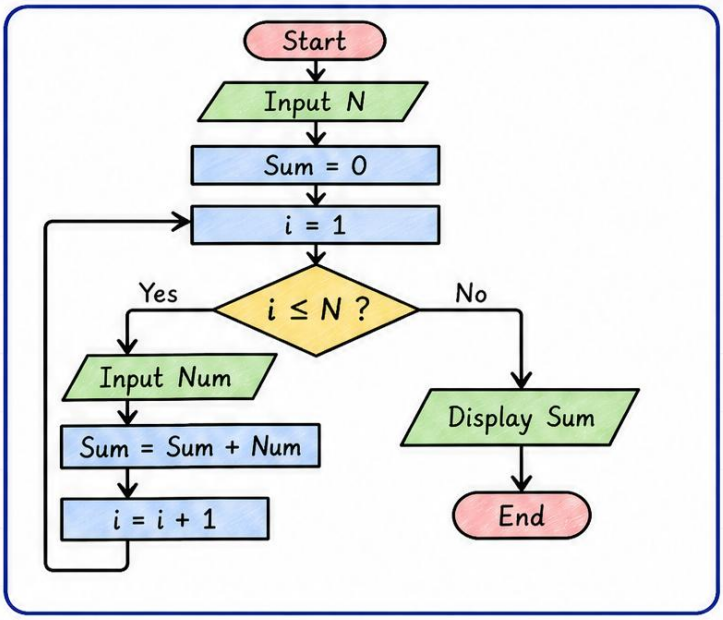



**उदाहरण 2:** संख्याओं के एक समूह का योग (Summation of a set of numbers)

1. Input: N (संख्याओं की कुल संख्या) पढ़िए।
2. Sum = 0 कर दीजिए।
3. i = 1 से N तक दोहराइए:
   - एक संख्या पढ़िए (Num)।
   - Sum = Sum + Num कीजिए।
4. Sum का मान display कीजिए।
5. End।



# S.P Group of Institute

## Programming and Problem Solving Through Python Language Notes

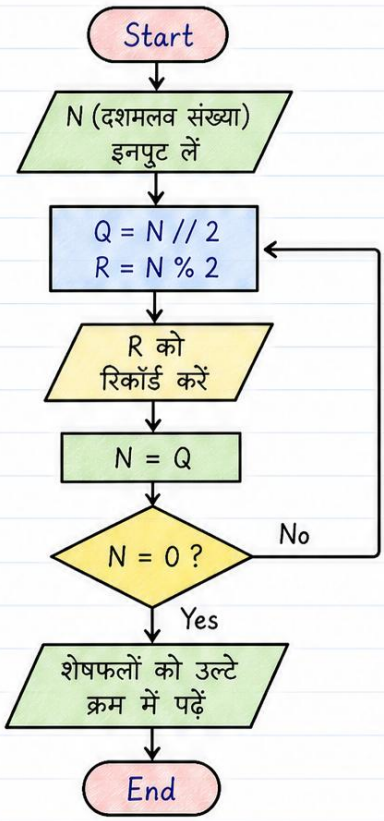
## Unit 2: Algorithms and Flowcharts to Solve Problems

### Number Conversion Algorithms

### 1. दशमलव (Decimal) से द्विआधारी (Binary) में रूपांतरण

किसी भी दशमलव संख्या को द्विआधारी में बदलने के लिए:

1. संख्या को 2 से भाग दें।
2. भागफल (Quotient) और शेषफल (Remainder) प्राप्त करें।
3. शेषफल (0 या 1) को लिखते जाएँ।
4. भागफल को पुनः 2 से भाग दें।
5. जब भागफल 0 हो जाए, तब रूक जाएँ।
6. सभी शेषफलों को उल्टे क्रम में पढ़ें; यही द्विआधारी संख्या है।




उदाहरण: $13_{10}$ को द्विआधारी में बदलें।

| चरण | भाग | भागफल (Q) | शेषफल (R) |
|---|---|---|---|
| 1 | 13 ÷ 2 | 6 | 1 |
| 2 | 6 ÷ 2 | 3 | 0 |
| 3 | 3 ÷ 2 | 1 | 1 |
| 4 | 1 ÷ 2 | 0 | 1 |

नीचे से ऊपर की ओर शेषफलों को पढ़ें:

| |
|---|
| 1 |
| 0 |
| 1 |
| 1 |

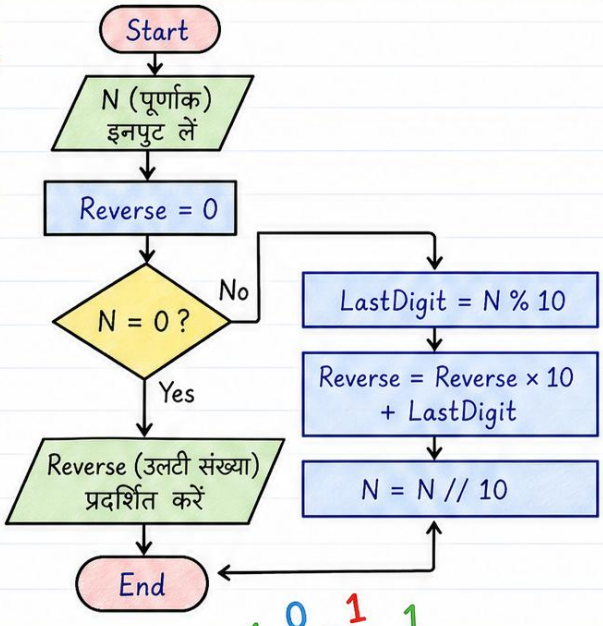
अतः $13_{10} = 1101_2$

---

### 2. किसी पूर्णांक (Integer) की अंकों को उलट देना

किसी भी पूर्णांक की अंकों को उलटने के लिए:

1. LastDigit = N % 10 (आखिरी अंक प्राप्त करें)
2. Reverse = Reverse × 10 + LastDigit (अंक को जोड़ें)
3. N = N // 10 (आखिरी अंक हटा दें)
4. जब तक N = 0 न हो जाए, चरण 1 से 3 दोहराएँ।




उदाहरण: 1234 की अंकों को उलटें।

| चरण | N | LastDigit (N % 10) | Reverse (अद्यतन) | N (अद्यतन) N // 10 |
|---|---|---|---|---|
| प्रारंभ | 1234 | – | 0 | 1234 |
| 1 | 1234 | 4 | 4 | 123 |
| 2 | 123 | 3 | 43 | 12 |
| 3 | 12 | 2 | 432 | 1 |
| 4 | 1 | 1 | 4321 | 0 |

अतः 1234 → 4321

# S.P Group of Institute

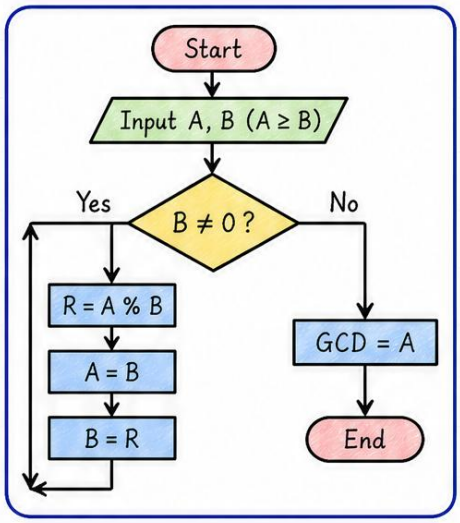
Programming and Problem Solving Through Python Language Notes

## Unit 2: Algorithms and Flowcharts to Solve Problems

## GCD और Prime Number

### 1. GCD (Greatest Common Divisor) of two numbers

यूक्लिडियन विधि (Euclidean method) का उपयोग करके दो संख्याओं का महत्तम समापवर्तक (GCD) ज्ञात किया जाता है।




**एल्गोरिथ्म:**

1. दो संख्याएँ A और B इनपुट करें। (A ≥ B)
2. जब तक B ≠ 0 हो, तब तक दोहराएँ:
   - R = A % B
   - A = B
   - B = R
3. GCD = A

**उदाहरण:**

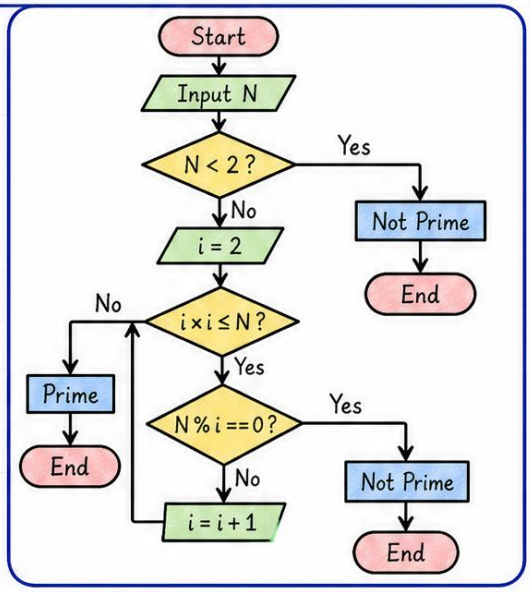
GCD(48, 18) = ?

| चरण | A | B | R = A % B |
|---|---|---|---|
| शुरुआत | 48 | 18 | – |
| चरण 1 | 48 | 18 | 12 |
| चरण 2 | 18 | 12 | 6 |
| चरण 3 | 12 | 6 | 0 |
| रुकें | 6 | 0 | – |

अत: GCD(48, 18) = 6

### 2. Test whether a number is prime

किसी संख्या के अभाज्य (Prime) होने की जाँच करने के लिए:




**एल्गोरिथ्म:**

1. यदि N < 2, तो संख्या अभाज्य (Prime) नहीं है।
2. i को 2 से लेकर $\sqrt{N}$ तक जाँचें।
3. यदि N, i से पूरी तरह विभाजित हो जाता है (N % i == 0), तो संख्या अभाज्य नहीं है।
4. यदि कोई भी i, N को विभाजित नहीं करता, तो संख्या अभाज्य (Prime) है।

**उदाहरण:**

29 संख्या की जाँच करें।

- N = 29 (N ≥ 2)
- $\sqrt{N} = \sqrt{29} \approx 5.38$, इसलिए i = 2, 3, 4, 5 तक जाँच करेंगे।
- 29 % 2 ≠ 0, 29 % 3 ≠ 0, 29 % 4 ≠ 0, 29 % 5 ≠ 0
- कोई भी भाजक नहीं मिला, इसलिए 29 अभाज्य (Prime) है।

29 is a Prime number

# S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

## Unit 2: Algorithms and Flowcharts to Solve Problems

## Factorial और Fibonacci Sequence

### 1. Factorial computation:

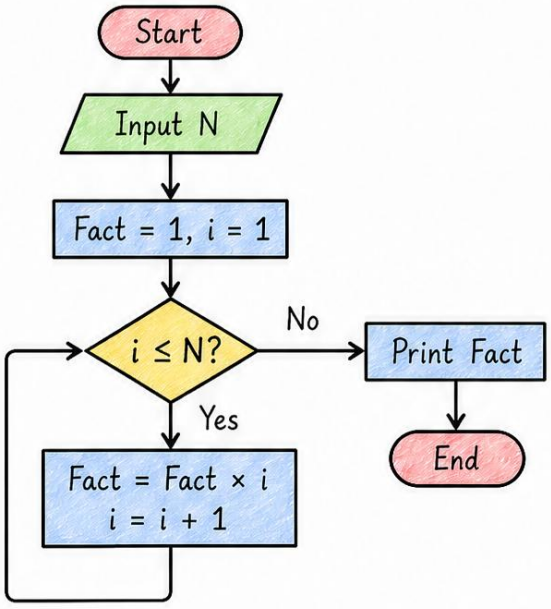
- N ≥ 0 के लिए, Fact = 1 सेट करें।
- 1 से N तक प्रत्येक पूर्णांक से Fact को गुणा करें।
- $N! = 1 \times 2 \times \ldots \times N$
- $0! = 1$

उदाहरण: $5! = 120$

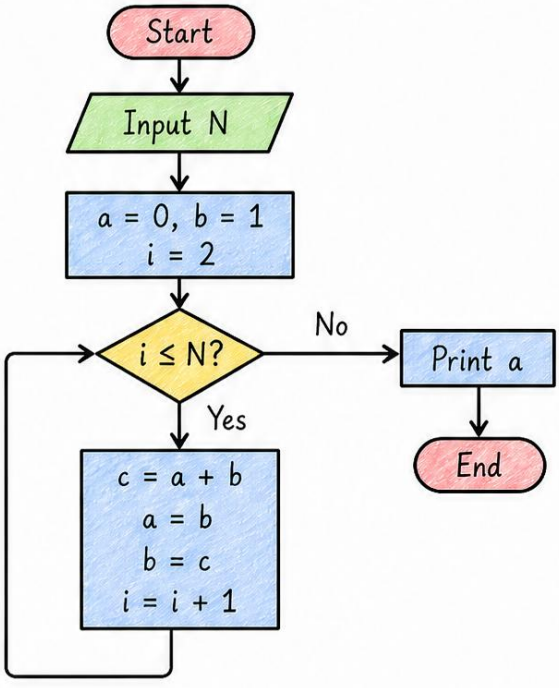
### 2. Fibonacci sequence:

- Fibonacci sequence के पहले दो पद 0 और 1 होते हैं।
- अगला पद पिछले दो पदों के योग के बराबर होता हैं।
- अनुक्रम: 0, 1, 1, 2, 3, 5, 8, 13

Factorial computation Flowchart




Fibonacci sequence Flowchart



# S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

## Unit 2: Algorithms and Flowcharts to Solve Problems

### Decision-Based Processing

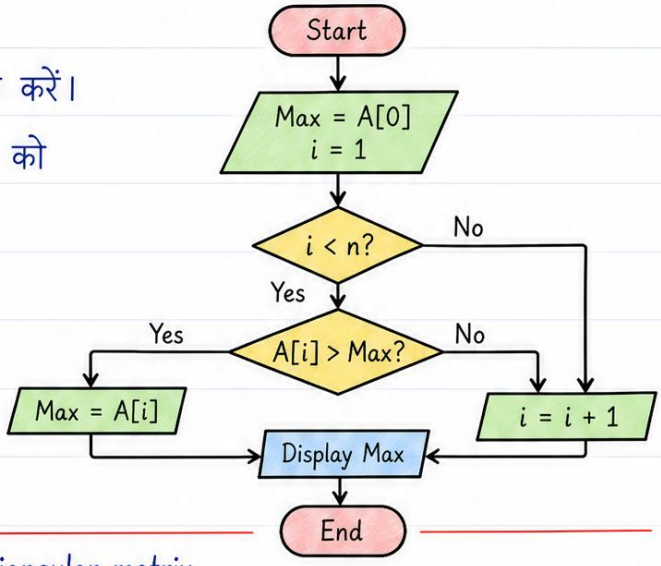
निर्णय-आधारित प्रोसेसिंग में, किसी शर्त (Condition) के आधार पर दो या अधिक पाथ (Paths) में से एक की चुना जाता हैं।

#### उदाहरण 1: Find largest number in an array

A

| 12 | 7 | 25 | 9 | 18 | ... |
|---|---|---|---|---|---|

- Max = A[0] सेट करें।
- प्रत्येक अगले तत्त की तुलना Max से करें।
- यदि तत्त Max से बड़ा हो, तो Max को उससे अपडेट करें।
- अंत में Max को प्रदर्शित करें।



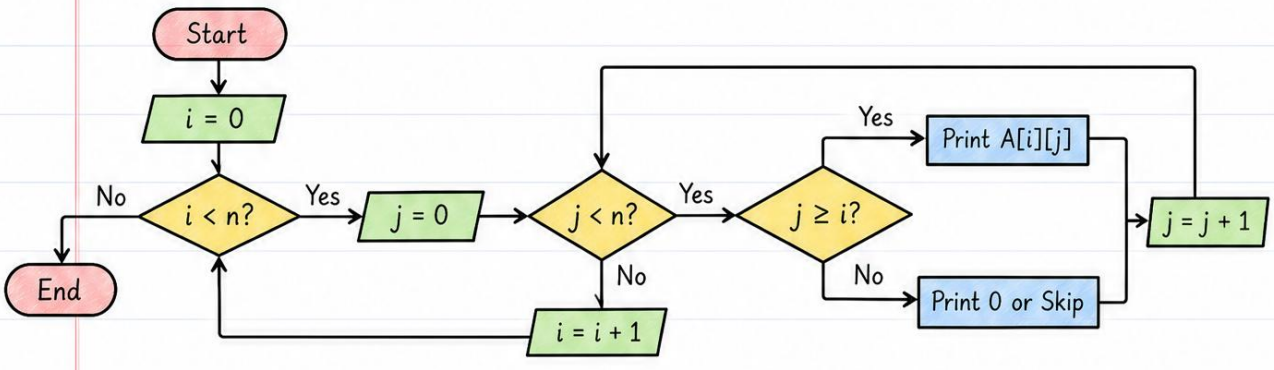

#### उदाहरण 2: Print elements of upper triangular matrix

- $i = 0$ से $n - 1$ तक दोहराएँ।
- प्रत्येक $i$ के लिए $j = 0$ से $n - 1$ तक दोहराएँ।
- यदि $j \geq i$, तो $A[i][j]$ प्रिंट करें; अन्यथा 0 प्रिंट करें या स्किप करें।
- अगली पंक्ति पर जाएँ।

A =

| 1 | 2 | 3 |
|---|---|---|
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Upper triangle cells



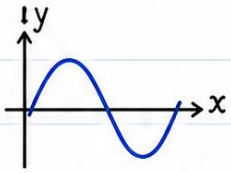
# S.P Group of Institute

## Programming and Problem Solving Through Python Language Notes

## Unit 2: Algorithms and Flowcharts to Solve Problems

### Iterative Processing: Series और Array Algorithms

Iterative Processing में loop के द्वारा steps को बार-बार दोहराया जाता हैं जब तक condition पूरी न हो।



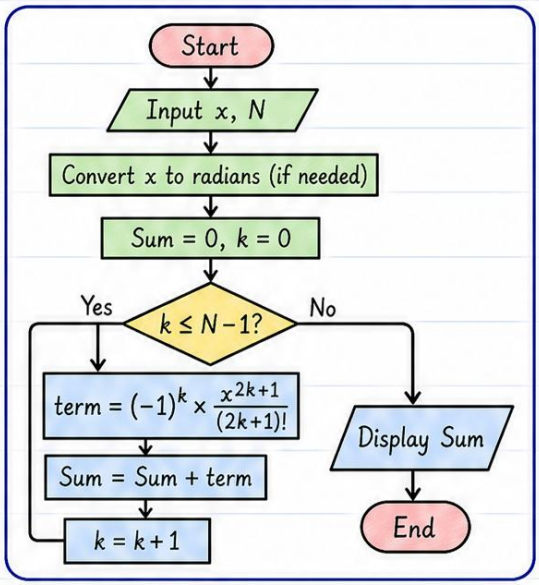

**5!**

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

### 1. Series का योग करके 'sin x' का मान ज्ञात करें

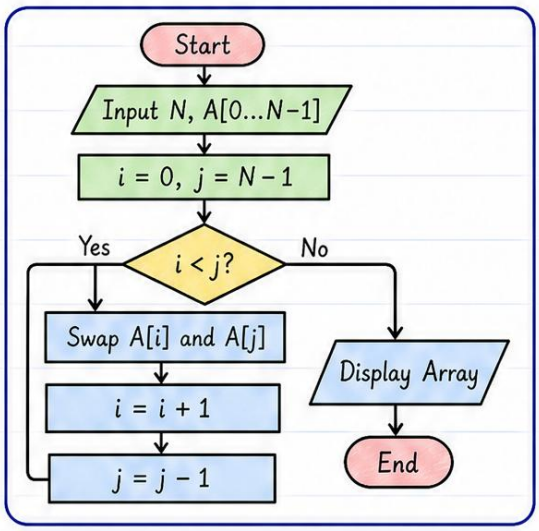
$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

1. x को radians में बदलें (यदि आवश्यक हो)।
2. Sum = 0 सेट करें।
3. k = 0 से N − 1 तक के लिए दोहराएँ:
   - term = $(-1)^k \times x^{2k+1}/(2k+1)!$ निकालें।
   - Sum = Sum + term करें।
4. Sum प्रदर्शित करें।




### 2. Array के elements का reverse order में प्रबंध करें

1. N और A[0...N−1] इनपुट करें।
2. i = 0 और j = N − 1 सेट करें।
3. जब तक i < j हो, निम्न करें:
   - A[i] और A[j] को आपस में बदलें (swap करें)।
   - i = i + 1 करें।
   - j = j − 1 करें।
4. Array प्रदर्शित करें।




**उदाहरण:**

A = [1, 2, 3, 4, 5] → [5, 4, 3, 2, 1]