The Center of Applied Data Science

Problem Solving in Data Science

Introduction to Algorithms

May 2019

Outline

1. Introduction
2. What is an Algorithm?
3. Describing Algorithms with Flowcharts
4. Introduction to Programming

Outline

1. Introduction
2. What is an Algorithm?
3. Describing Algorithms with Flowcharts
4. Introduction to Programming

What is an Algorithm?

Definition and Structure

The Center of Applied Data Science

Definition

A set of step-by-step instructions to solve a problem.

Requirements:

- Can be described in a formal language
- Consist of a finite number of steps
- Operate on zero or more inputs
- Result in an output
- Individual steps are sufficiently basic and can be executed in finite time

What is an Algorithm?

Examples of Algorithms

Calculate the sum of the first 10 positive numbers

Recipe for baking a cake

Recommend products to consumers based on previous transactions

Exercise 1

Write an algorithm to find the page number of the chapter *Little Em'ly* of the book *David Copperfield* by Charles Dickens.

What is an Algorithm?

Examples of Algorithms

Write an algorithm to find the page number of the chapter *Little Em’ly* of the book *David Copperfield* by Charles Dickens.

Solution A:

- Open book
- Flip through pages we see the chapter title *Little Em’ly* at the top of the page
- Write down page number

Solution B:

- Open book
- Turn to the table of contents
- Write down page number of chapter *Little Em’ly*

What is an Algorithm?

Examples of Algorithms

Write an algorithm to find the page number of the chapter *Little Em'ly* of the book *David Copperfield* by Charles Dickens.

Solution A:

- Open book
- Flip through pages we see the chapter title *Little Em'ly* at the top of the page
- Write down page number

Solution B:

- Open book
- Turn to the table of contents
- Write down page number of chapter *Little Em'ly*

Many different solutions are possible!

What is an Algorithm?

Examples of Algorithms

Exercise 2

Write an algorithm that takes two numbers and adds their squares.

Examples of Algorithms

Write an algorithm that takes two numbers and adds their squares.

Solution A:

- Input first number
- Compute square of first number
- Input second number
- Compute square of second number
- Add the squares of the numbers
- Output the sum of the squares

Solution B:

- $f(x, y) = x^2 + y^2$

What is an Algorithm?

Examples of Algorithms

Write an algorithm that takes two numbers and adds their squares.

Solution A:

- Input first number
- Compute square of first number
- Input second number
- Compute square of second number
- Add the squares of the numbers
- Output the sum of the squares

Solution B:

- $f(x, y) = x^2 + y^2$

Outline

1 Introduction

2 What is an Algorithm?

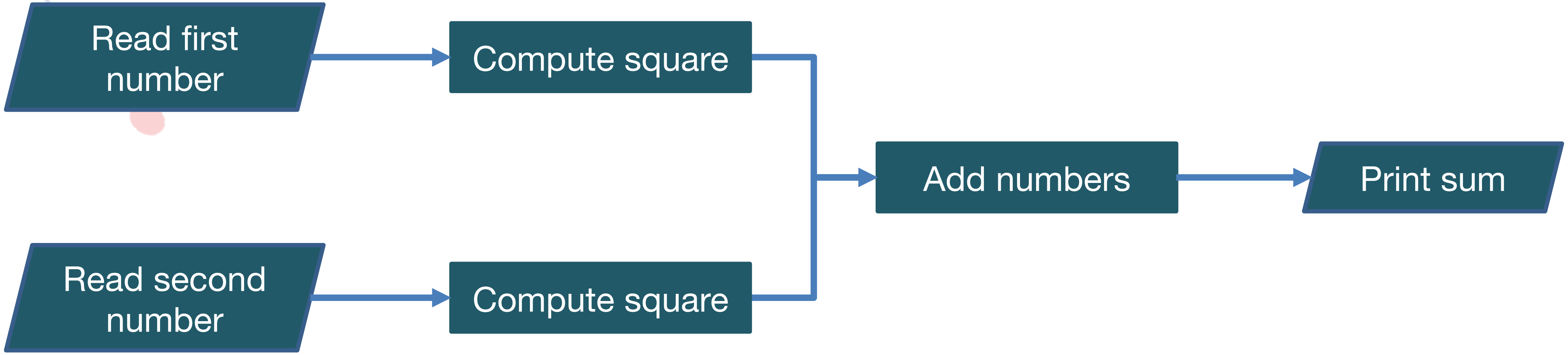
3 Describing Algorithms with Flowcharts

4 Introduction to Programming

Describing Algorithms with Flowcharts

What are Flowcharts?

Flowcharts are connected sequences of instructions



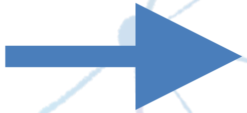
Allows for easy visualization of algorithmic logic

Provides a common language for algorithmic logic

Typically go from top to bottom and left to right

Describing Algorithms with Flowcharts

Common Flowchart Symbols



Flowline: Shows the flow of the algorithm

Input/Output: Data read or produced by the algorithm. Represented by a parallelogram.

Process: An action, e.g. addition. Represented by a rectangle.

Decision: A conditional query that determines the path the program will take. Commonly a yes/no question. Represented by a diamond.

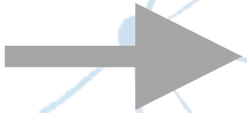
Terminal: The beginning and end of an algorithm. Represented by a stadium (rectangle with half-circles on either side).

Symbols are defined by the International Standards Organization (ISO). More info at https://en.wikipedia.org/wiki/Flowchart#Common_symbols

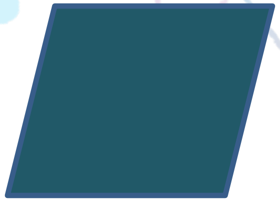
Describing Algorithms with Flowcharts

Common Flowchart Symbols

The Center of Applied Data Science



Flowline: Shows the flow of the algorithm



Input/Output: Data read or produced by the algorithm. Represented by a parallelogram.

Process: An action, e.g. addition. Represented by a rectangle.

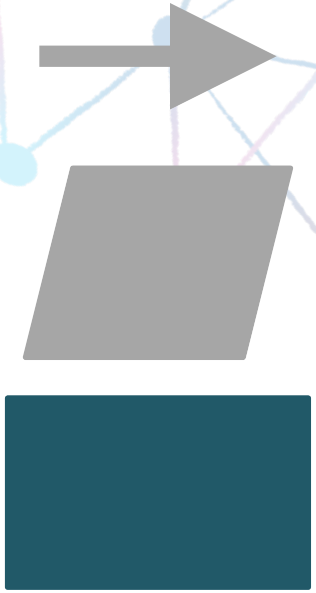
Decision: A conditional query that determines the path the program will take. Commonly a yes/no question. Represented by a diamond.

Terminal: The beginning and end of an algorithm. Represented by a stadium (rectangle with half-circles on either side).

Symbols are defined by the International Standards Organization (ISO). More info at https://en.wikipedia.org/wiki/Flowchart#Common_symbols

Describing Algorithms with Flowcharts

Common Flowchart Symbols



Flowline: Shows the flow of the algorithm

Input/Output: Data read or produced by the algorithm. Represented by a parallelogram.

Process: An action, e.g. addition. Represented by a rectangle.

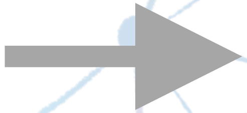
Decision: A conditional query that determines the path the program will take. Commonly a yes/no question. Represented by a diamond.

Terminal: The beginning and end of an algorithm. Represented by a stadium (rectangle with half-circles on either side).

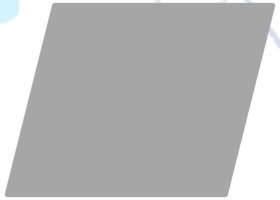
Symbols are defined by the International Standards Organization (ISO). More info at https://en.wikipedia.org/wiki/Flowchart#Common_symbols

Describing Algorithms with Flowcharts

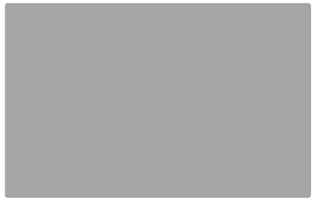
Common Flowchart Symbols



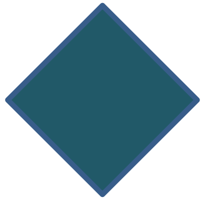
Flowline: Shows the flow of the algorithm



Input/Output: Data read or produced by the algorithm. Represented by a parallelogram.



Process: An action, e.g. addition. Represented by a rectangle.



Decision: A conditional query that determines the path the program will take. Commonly a yes/no question. Represented by a diamond.

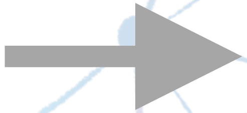
Terminal: The beginning and end of an algorithm. Represented by a stadium (rectangle with half-circles on either side).

Symbols are defined by the International Standards Organization (ISO). More info at https://en.wikipedia.org/wiki/Flowchart#Common_symbols

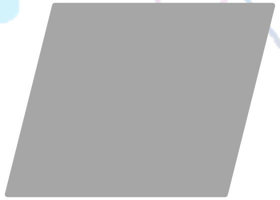
Describing Algorithms with Flowcharts

Common Flowchart Symbols

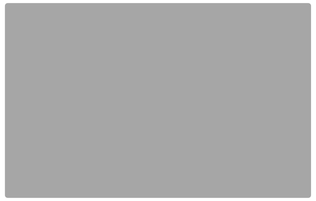
The Center of Applied Data Science



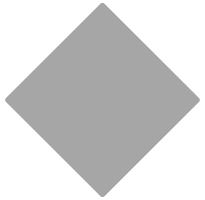
Flowline: Shows the flow of the algorithm



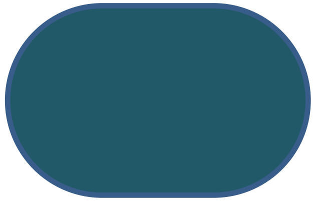
Input/Output: Data read or produced by the algorithm. Represented by a parallelogram.



Process: An action, e.g. addition. Represented by a rectangle.



Decision: A conditional query that determines the path the program will take. Commonly a yes/no question. Represented by a diamond.

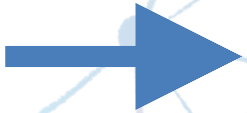


Terminal: The beginning and end of an algorithm. Represented by a stadium (rectangle with half-circles on either side).

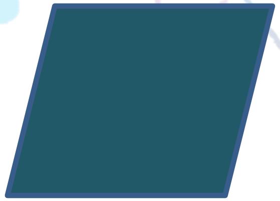
Symbols are defined by the International Standards Organization (ISO). More info at https://en.wikipedia.org/wiki/Flowchart#Common_symbols

Describing Algorithms with Flowcharts

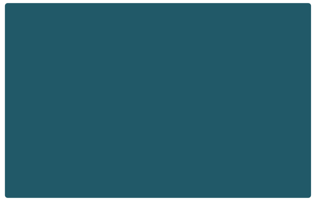
Common Flowchart Symbols



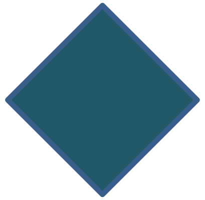
Flowline: Shows the flow of the algorithm



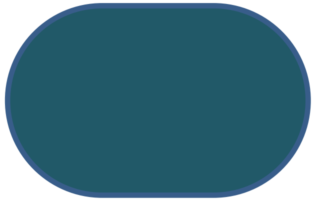
Input/Output: Data read or produced by the algorithm. Represented by a parallelogram.



Process: An action, e.g. addition. Represented by a rectangle.



Decision: A conditional query that determines the path the program will take. Commonly a yes/no question. Represented by a diamond.



Terminal: The beginning and end of an algorithm. Represented by a stadium (oval) (rectangle with half-circles on either side).

Symbols are defined by the International Standards Organization (ISO). More info at https://en.wikipedia.org/wiki/Flowchart#Common_symbols

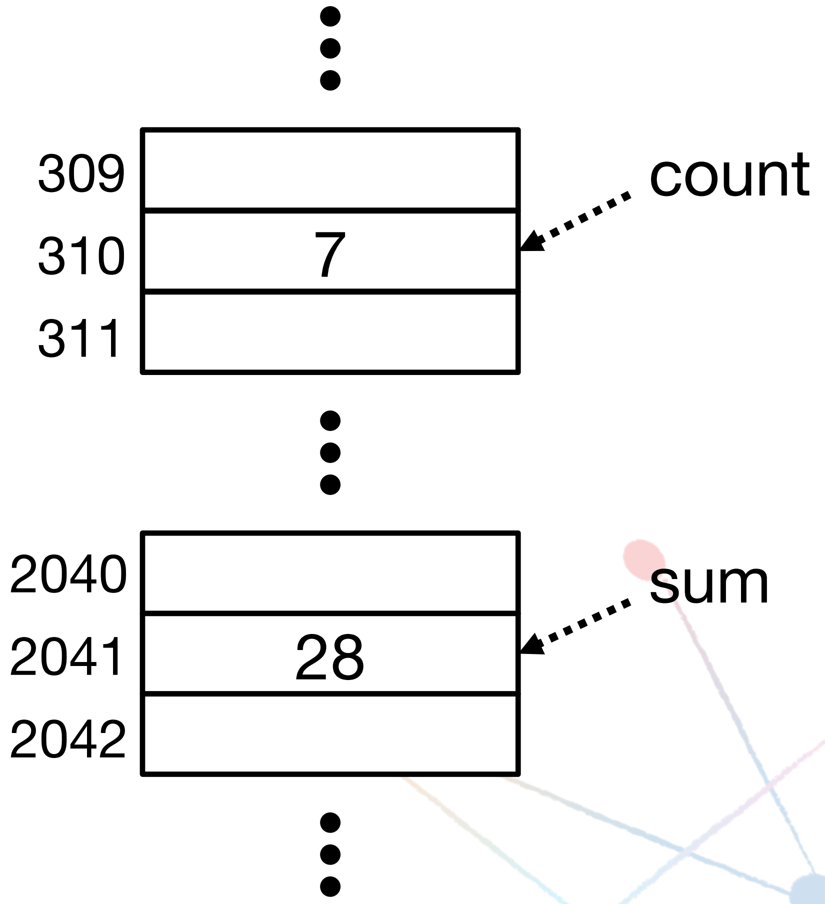
Definition

Variables are symbols that represent underlying, changeable values, e.g.

- words: Name = “Alex”
- numbers: Age = 50

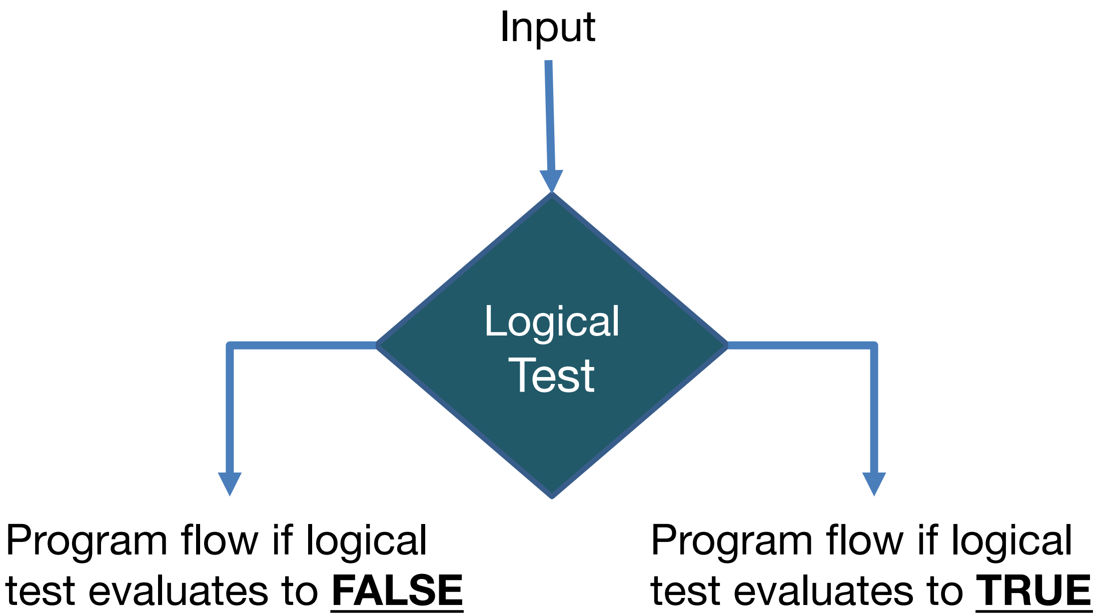
In computers:

- Variables reference blocks of memory at which data is stored



Definition

Conditional statements, also called if-else statements, perform a logical test and direct algorithm flow depending on the output.

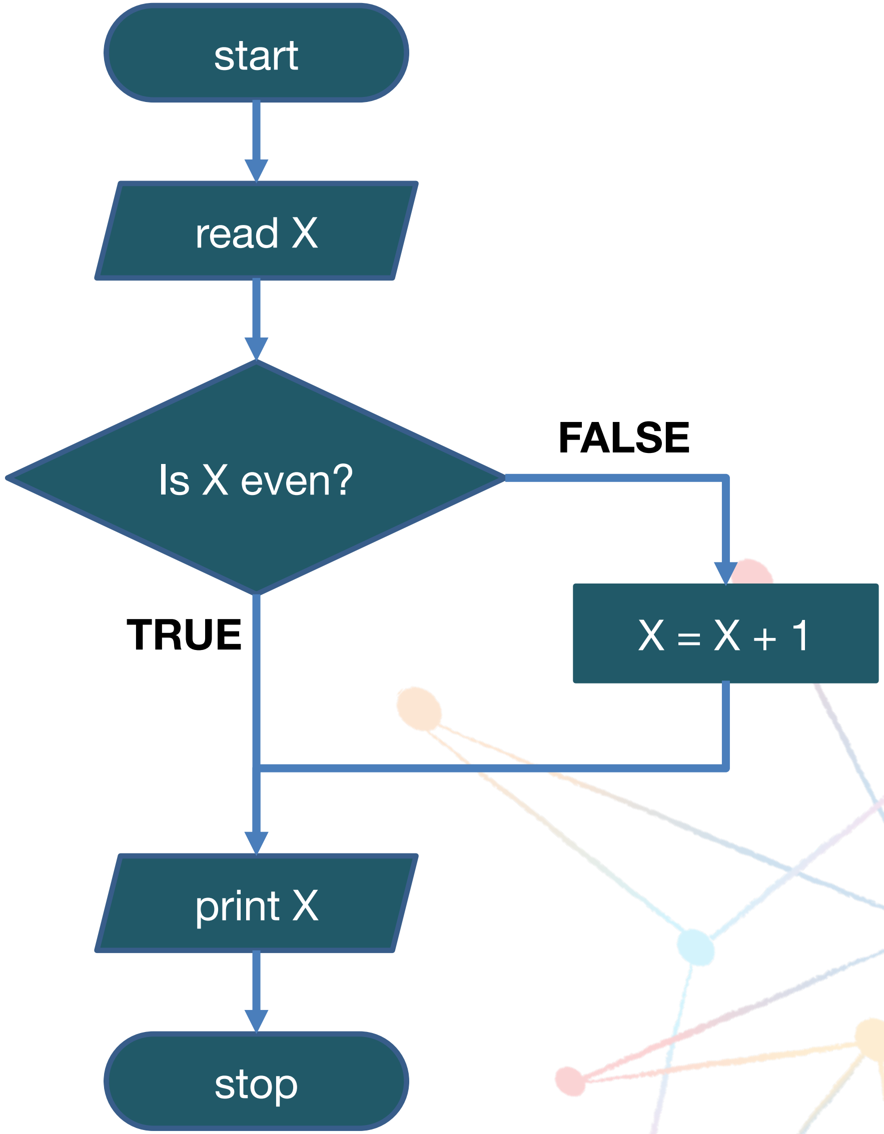

Input
Logical Test
Program flow if logical test evaluates to **FALSE**
Program flow if logical test evaluates to **TRUE**

Programming Elements

Conditional Statements

Conditional statements can also be used to skip over an action,

e.g. an algorithm that adds 1 to odd inputs but leaves even inputs unchanged.



Programming Elements
Conditional Statements

- Conditional statements require logical operations
  - Logical operations should result in a **Boolean value**:
    - **TRUE**, i.e. ‘yes’
    - **FALSE**, i.e. ‘no’

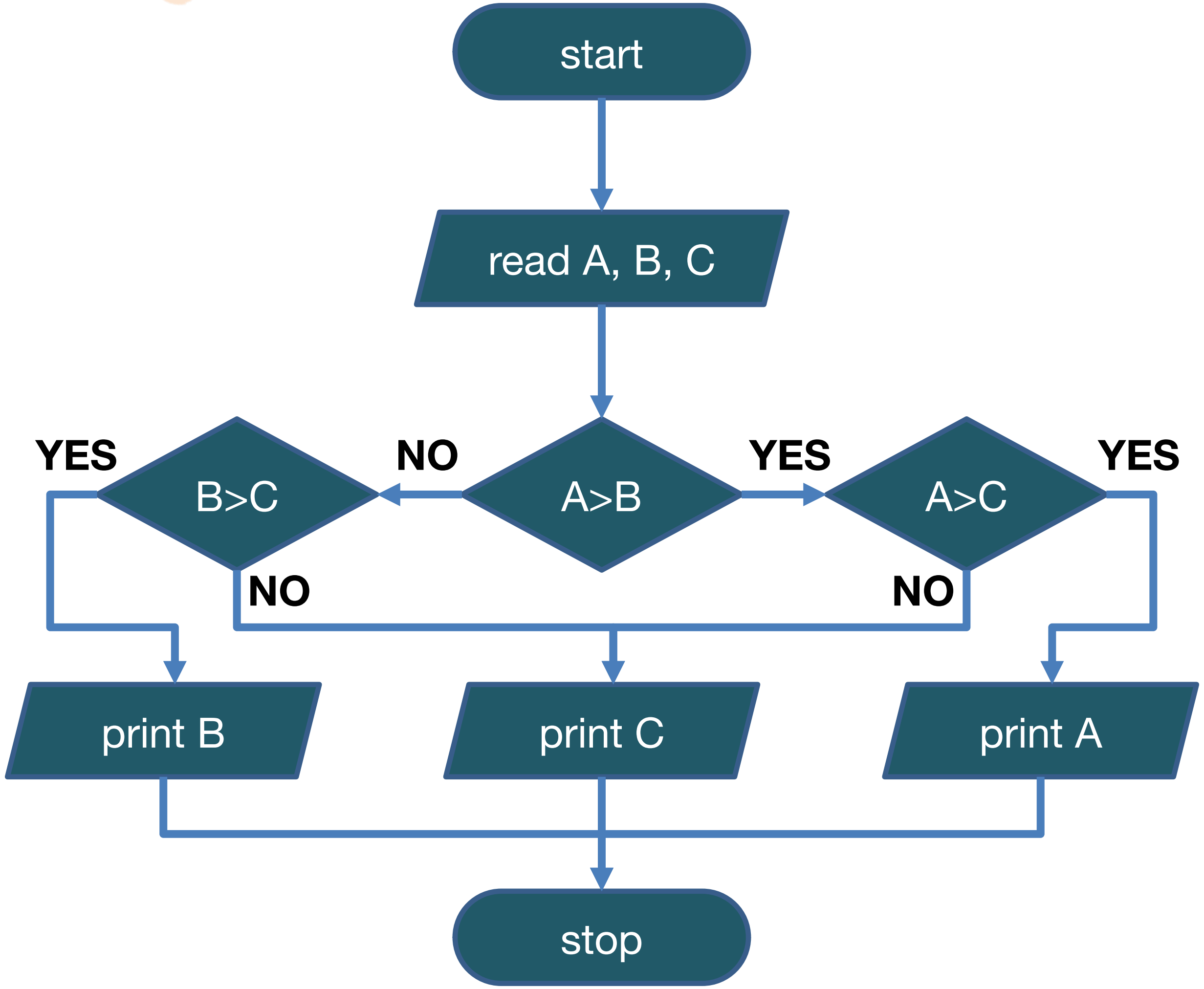
Relational Operators (for numerical values)	
x == y	Is x equal to y?
x ≠ y (x != y)	Is x not equal to y?
x < y	Is x less than y?
x > y	Is x greater than y?
x ≤ y (x <= y)	Is x less than or equal to y?
x ≥ y (x >= y)	Is x greater than or equal to y?

- Programming languages have different rules for how they interpret non-Boolean values
  - e.g. R and Python interpret 0 as FALSE and any non-zero number as TRUE

Exercise 3

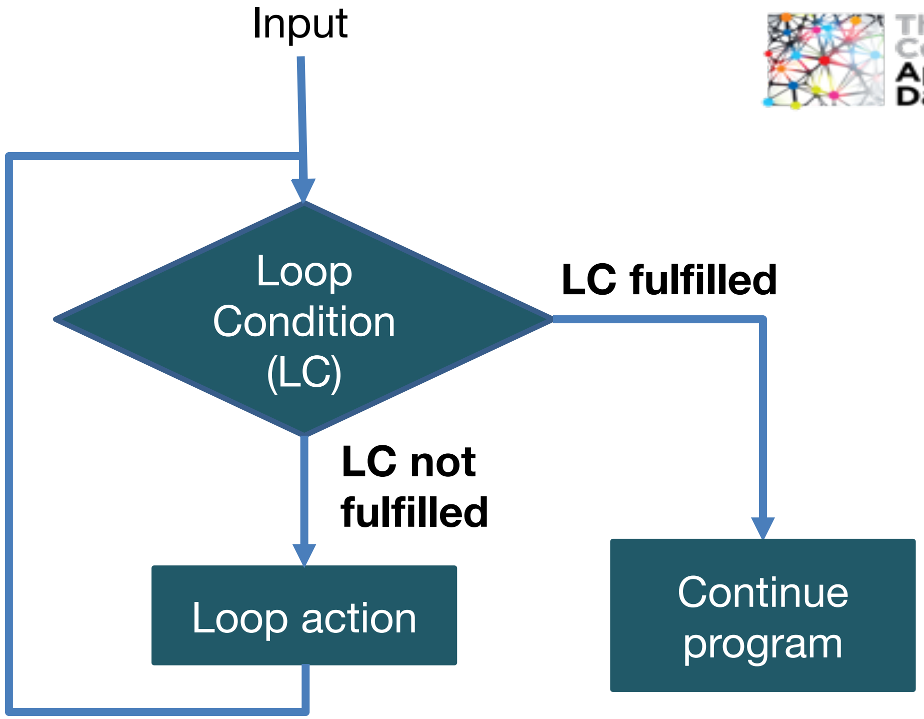
Design an algorithm as a flowchart that takes three numbers as input and prints the largest of them.

Design an algorithm as a flowchart that takes three numbers as input and prints the largest of them.



Definition

Loops are sequences of instructions that are executed repeatedly.



Two general types of loops exist:

- **count-controlled loops**, i.e. loops that execute a pre-defined number of times

  e.g. a loop that adds the first 10 positive numbers.

- **dynamically terminated loops**, i.e. loops that only terminate once a condition, evaluated within the loop, is met.

  e.g. a loop that continues until a user guesses a number

Exercise 4

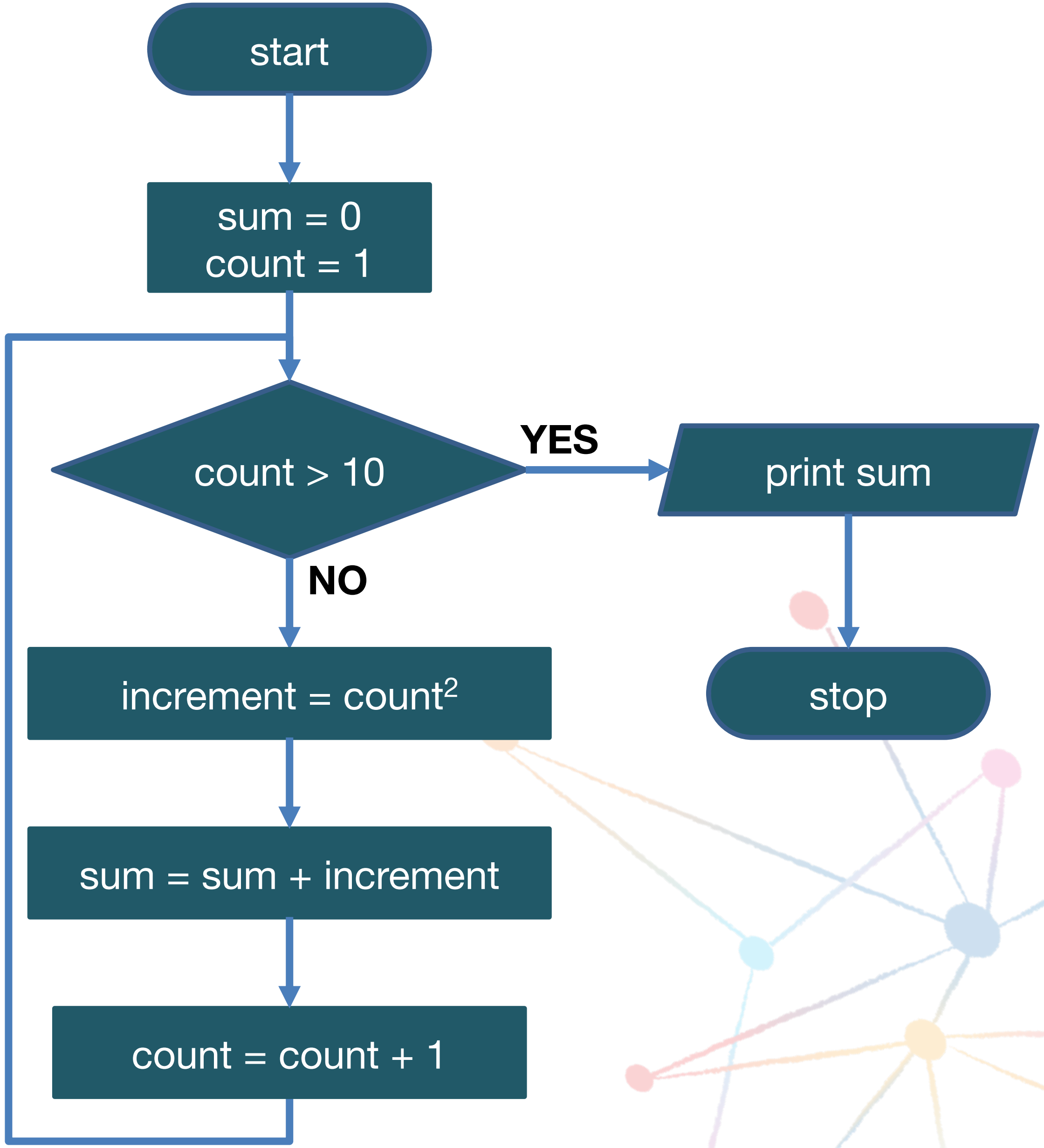
Design an algorithm as a flowchart that adds the squares of the first 10 numbers.

Programming Elements

Loops

Initialize two variables, placeholders for values.

- ***sum*** is the running sum of all numbers
- ***count*** keeps track of how many numbers have already been added to ***sum***
- If ***count*** is greater than 10, we've added the squares of the first 10 numbers to ***sum*** (since ***count*** started at 1). Output ***sum*** and end the algorithm.
- If ***count*** is not greater than 10, add its square to ***sum***, increase ***count*** by 1, and loop back to the conditional statement.



Programming Elements

Loops

What happens in the loop?

Iteration 1 (sum = 0; count = 1 → stay in loop)

- increment = count^2 = 1^2 = 1
- sum = sum + increment = 0 + 1 = 1
- count = count + 1 = 1 + 1 = 2

Iteration 2 (sum = 1; count = 2 → stay in loop)

- increment = count^2 = 2^2 = 4
- sum = sum + increment = 1 + 4 = 5
- count = count + 1 = 2 + 1 = 3

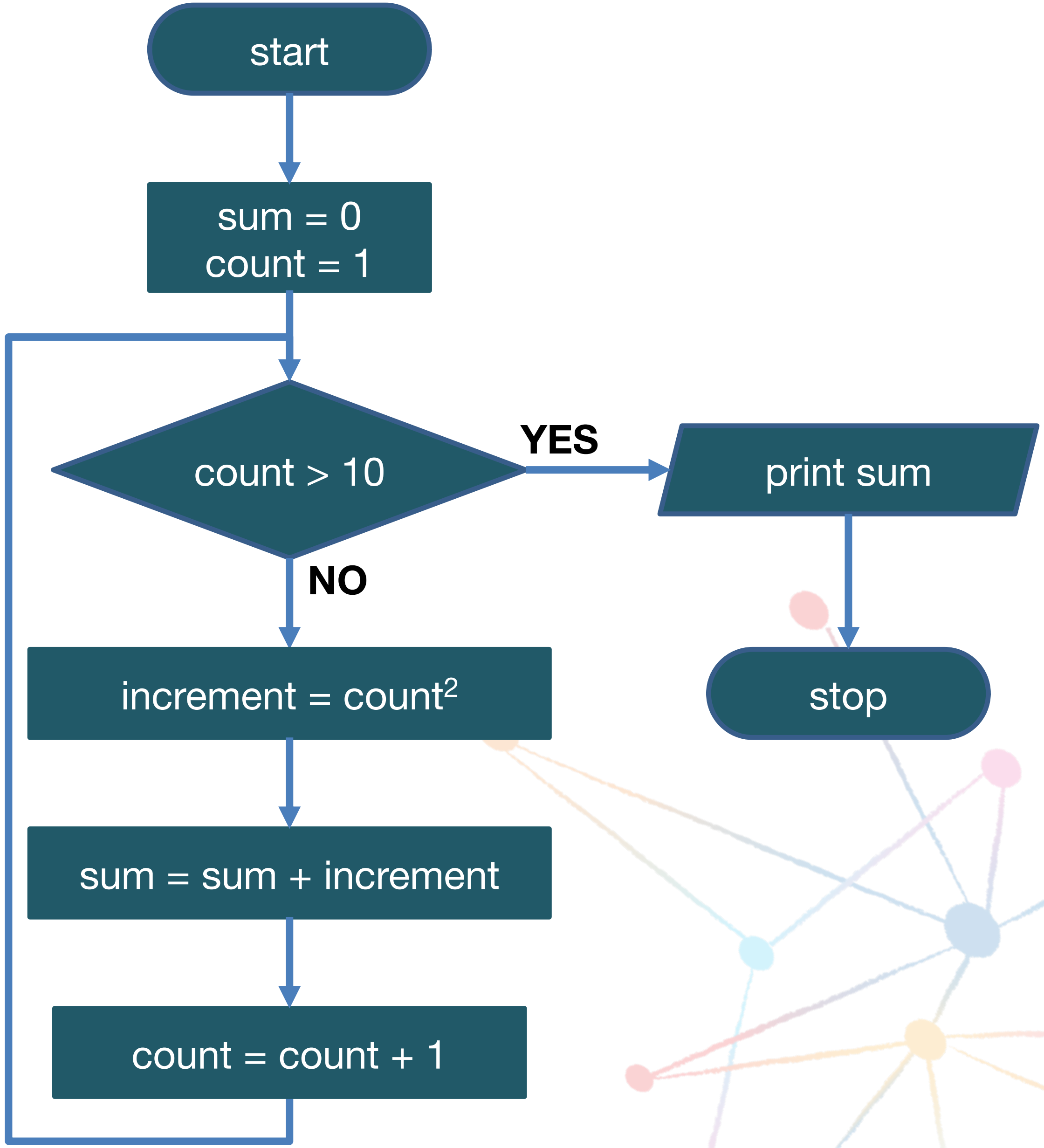
…

Iteration 10 (sum = 285; count = 10 → stay in loop)

- increment = count^2 = 10^2 = 100
- sum = sum + increment = 285 + 100 = 385
- count = count + 1 = 10 + 1 = 11

Iteration 11 (sum = 385; count = 11 → <u>leave loop!</u>)

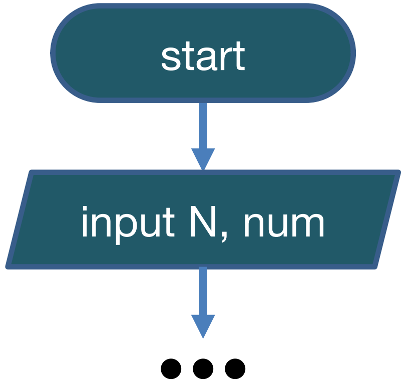
- print '385'



Exercise 5

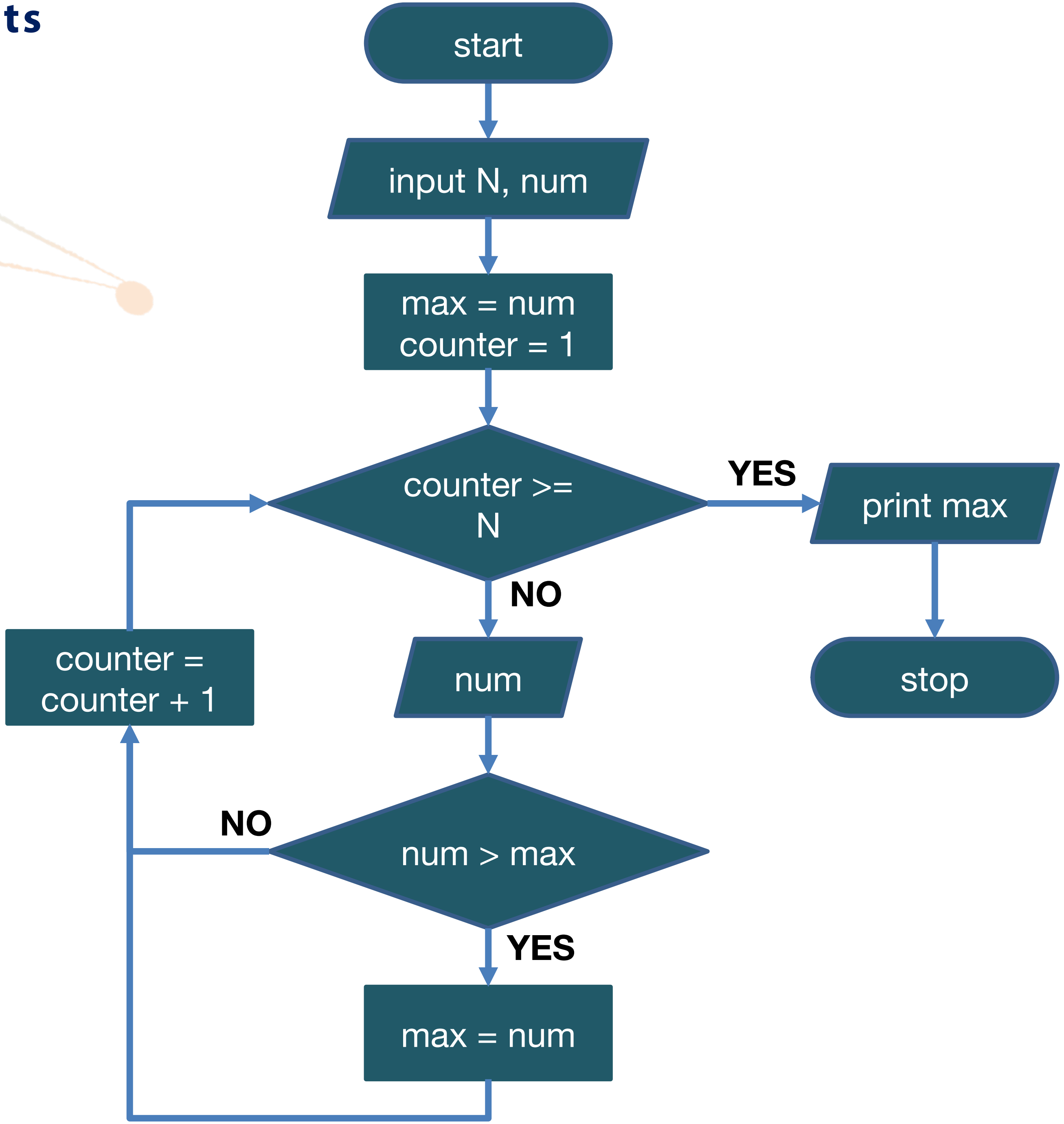
Design an algorithm as a flowchart that lets a user enter ‘N’ numbers and prints out the largest of the numbers. The algorithm should take ‘N’ as an input.

Hint:



Programming Elements

Loops



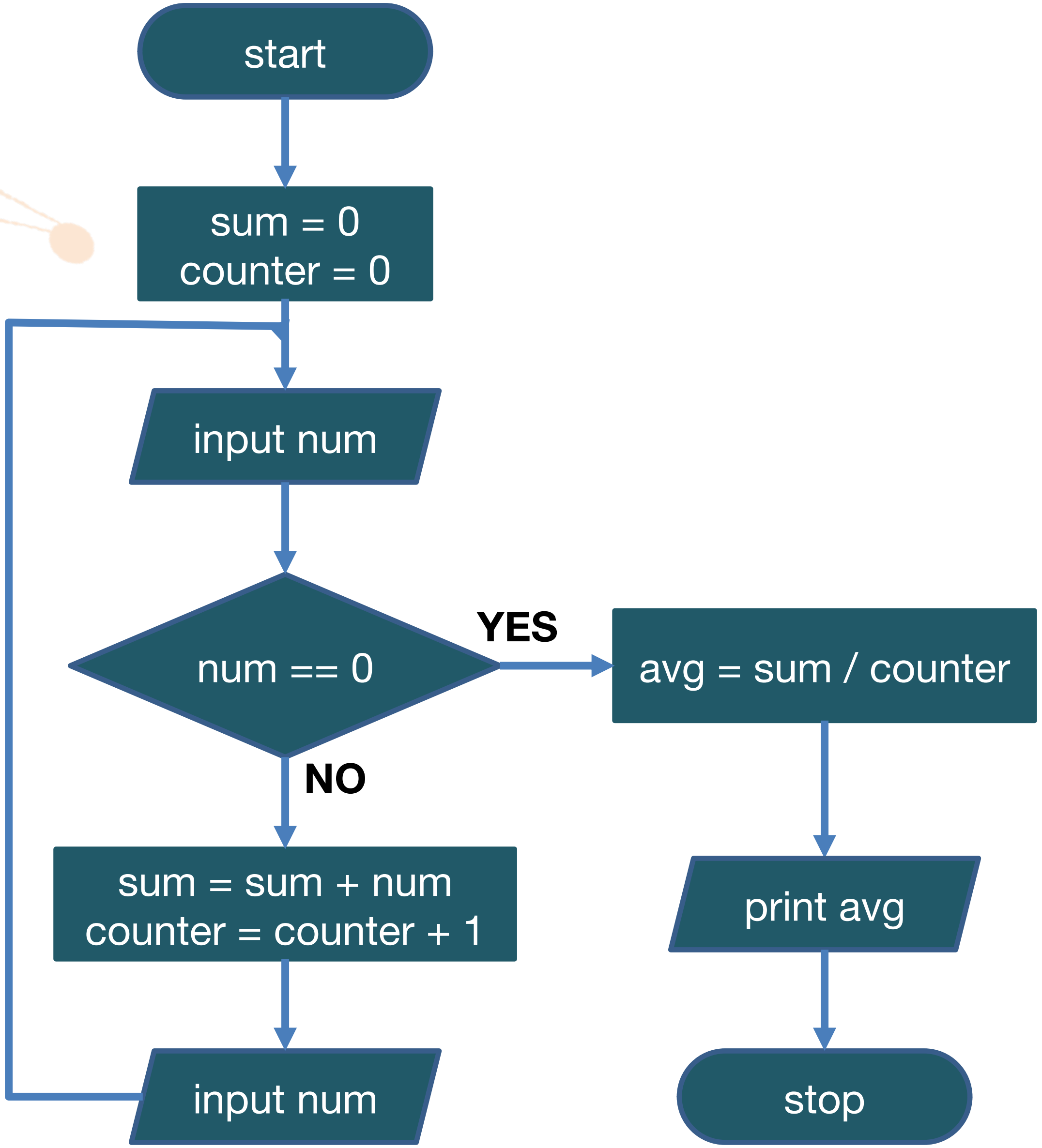
Exercise 6

Design an algorithm as a flowchart that reads user-entered numbers from the input until the user enters the number 0. Then, calculate and return the average value of all previously entered numbers (excluding the 0).

Programming Elements

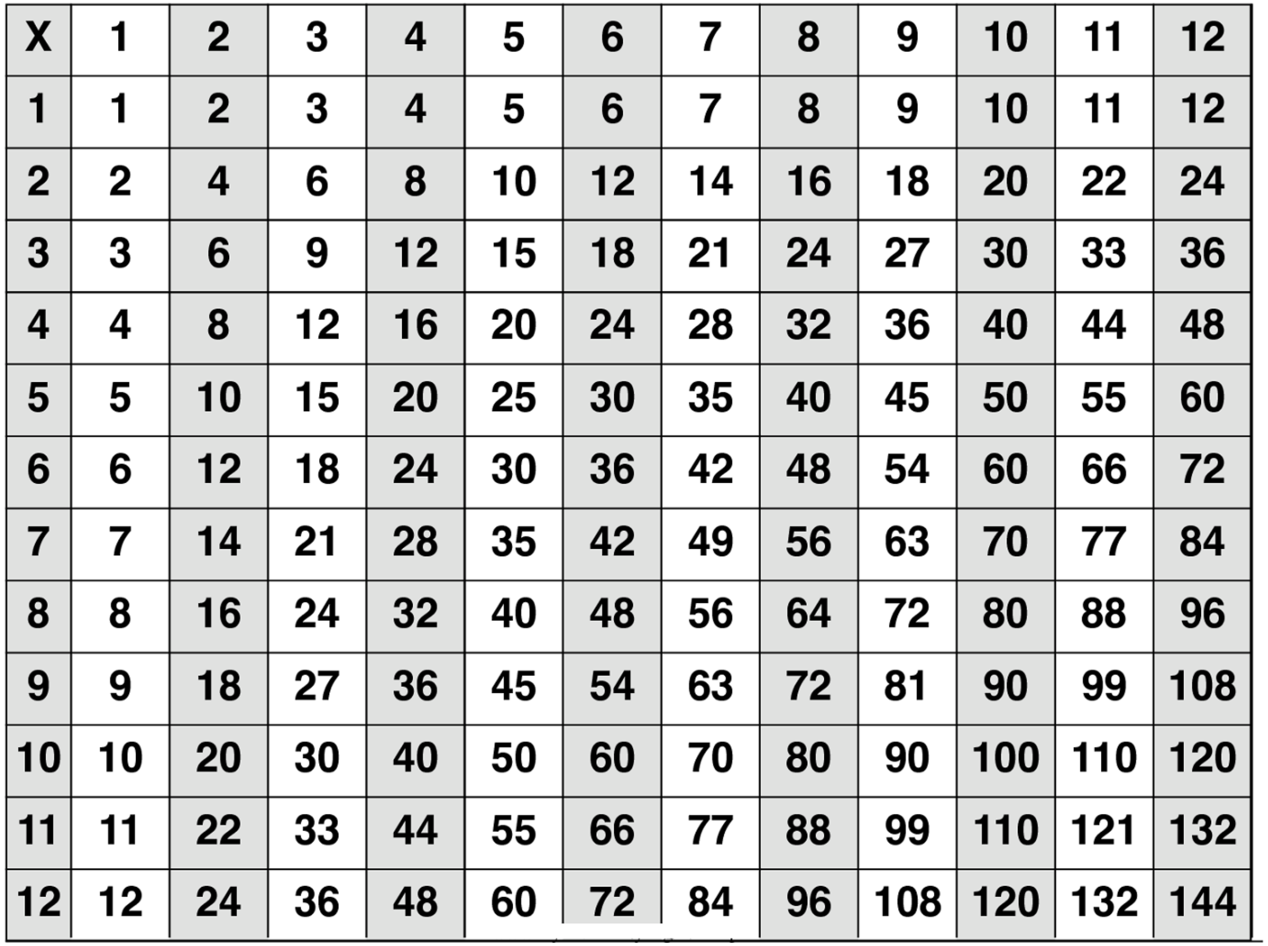
Loops

The Center of Applied Data Science



Exercise 7

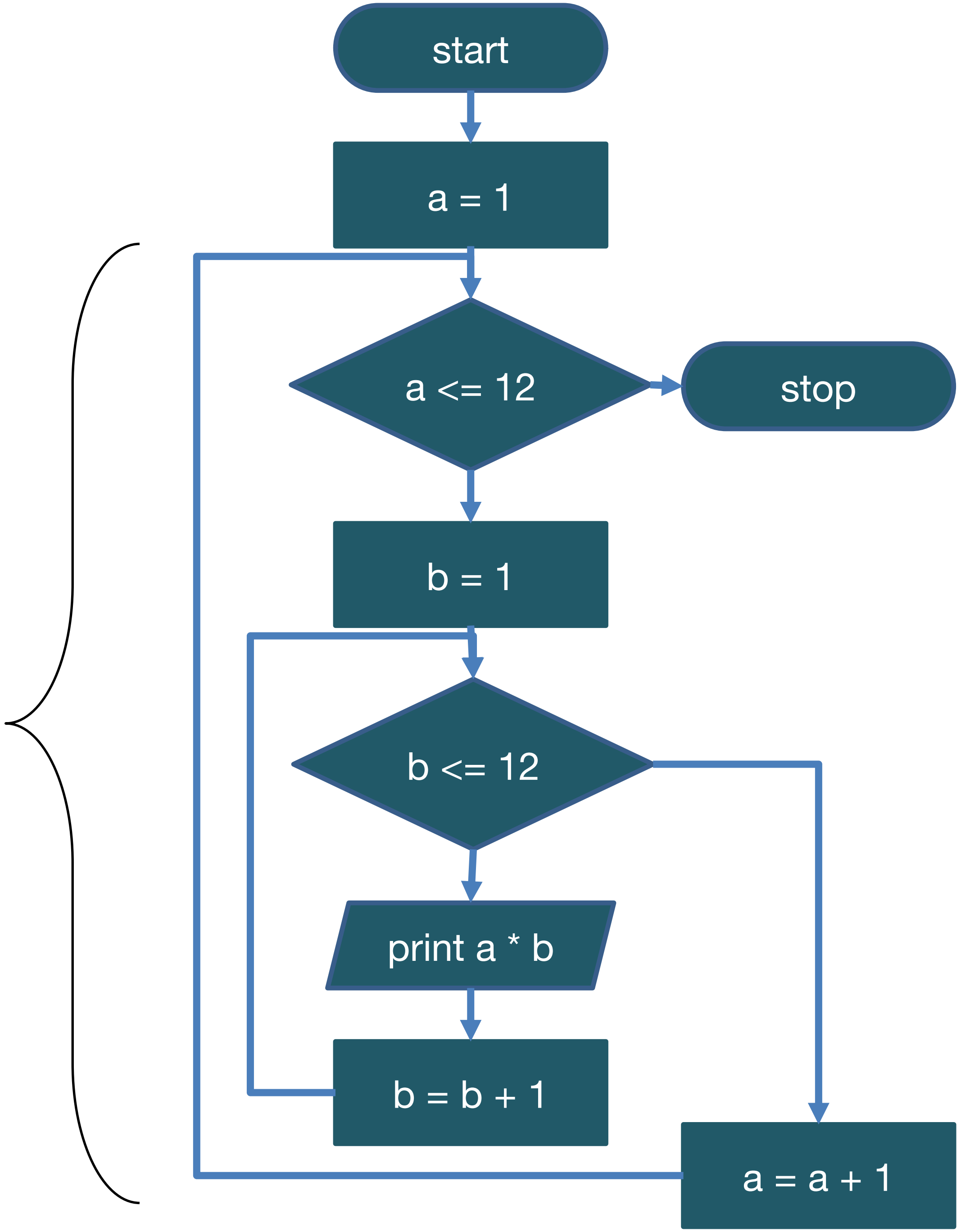
Design an algorithm as a flowchart that prints the multiplication table for numbers from 1 to 12.



X	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

Programming Elements

Loops



Nested loops

Outline

1 Introduction

2 What is an Algorithm?

3 Describing Algorithms with Flowcharts

4 Introduction to Programming

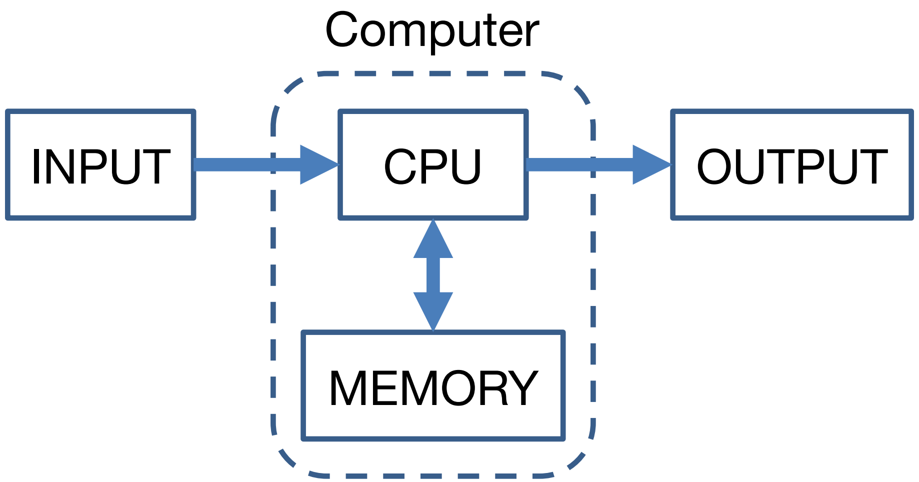
Implementing Algorithms as Computer Programs

Computers can be instructed to execute algorithms

Computer programs are implementations of algorithms

Programming languages are the formal languages of computers

- e.g. C, Java, Python, R



Introduction to Programming

Writing Pseudocode

Programming elements, i.e. variables, conditional statements, and loops, can be represented by nearly all programming languages, albeit with slight differences

Python

```
counter = 0
while counter < 5:
  counter = counter + 1
  if counter != 3:
    print(counter)
```

Java

```
int counter = 0;
while(counter < 5) {
  counter = counter + 1;
  if(counter != 3) {
    System.out.println(counter);
}}
```

R

```
counter <- 0
while(counter < 5) {
  counter <- counter + 1;
  if(counter != 3) {
    print(counter)
}}
```

➔ Identical output!

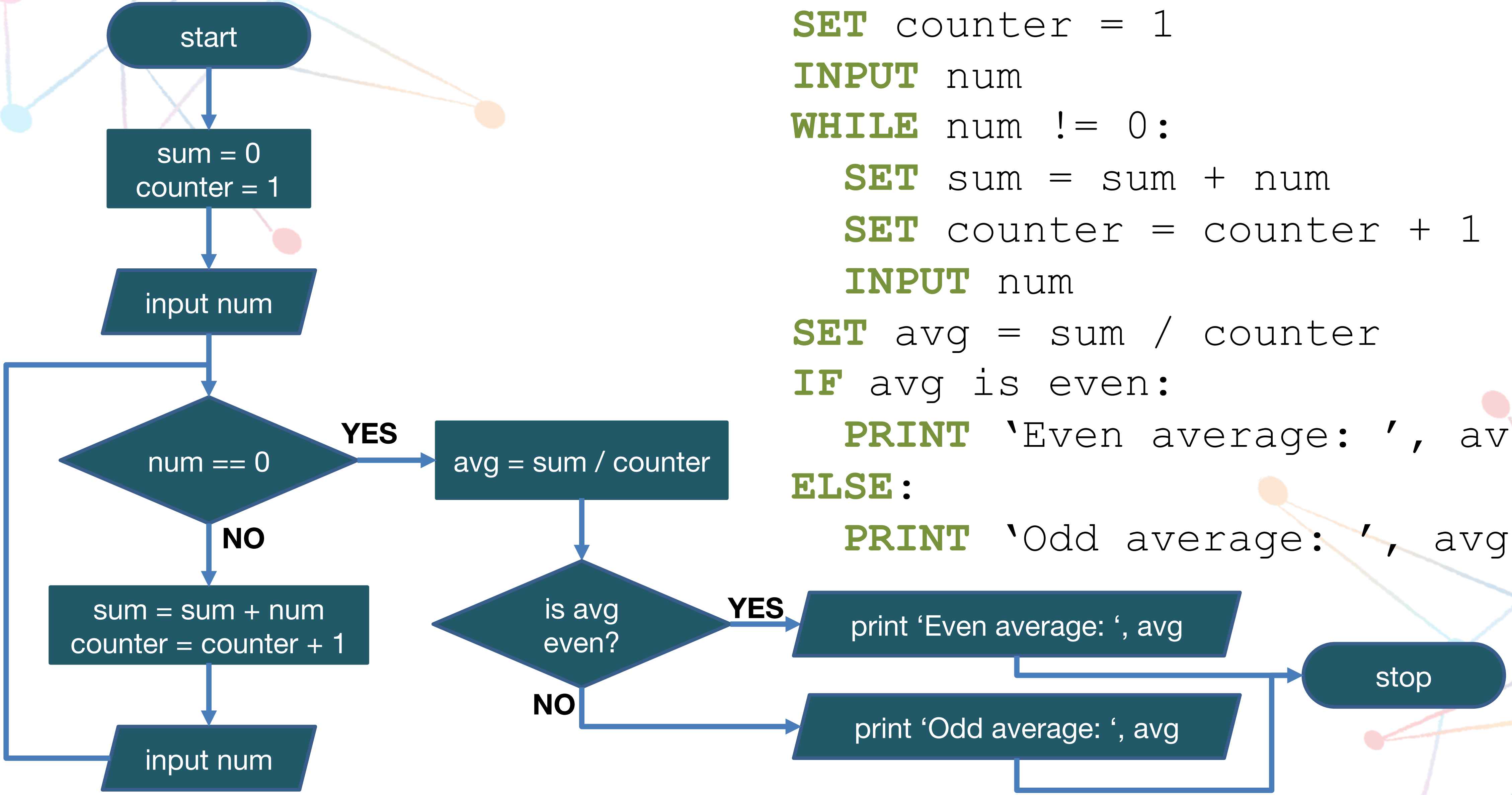
Introduction to Programming

Writing Pseudocode

- **Pseudocode** is an informal, human-readable, descriptive language meant to resemble common programming languages in its structure
- There is no formal syntax – many variations exist that resemble different programming languages

Introduction to Programming

Pseudocode Example



```
ALGORITHM "Averages":
  SET sum = 0
  SET counter = 1
  INPUT num
  WHILE num != 0:
    SET sum = sum + num
    SET counter = counter + 1
    INPUT num
  SET avg = sum / counter
  IF avg is even:
    PRINT 'Even average: ', avg
  ELSE:
    PRINT 'Odd average: ', avg
```

Pseudocode Example

Exercise 8

Design an algorithm to play the ‘Fizz Buzz’ game. The algorithm should count from 1 to 100 and print out the numbers. However, if a number is divisible by 3, the algorithm should print ‘Fizz’ instead of the number. If a number is divisible by 5, it should print ’Buzz’ instead of the number. If a number is divisible by both 3 and 5, the algorithm should print ‘Fizz Buzz’ instead of the number. The output should therefore look as follows:

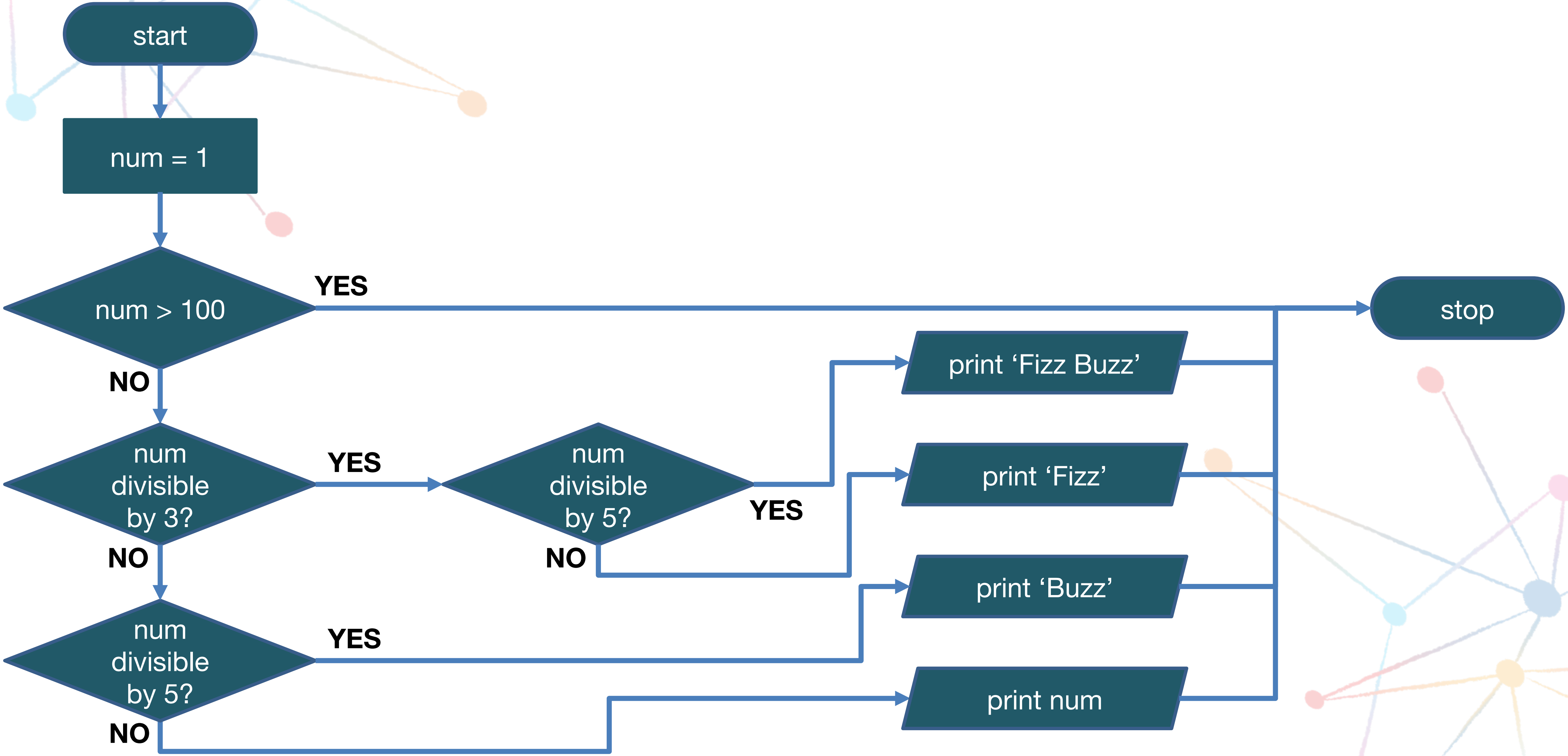
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz Buzz, 16

Begin by designing a flowchart for this algorithm and then try to translate it into pseudo code.

Introduction to Programming

Pseudocode Example

The Center of Applied Data Science



Introduction to Programming

Pseudocode Example

```
ALGORITHM “Fizz Buzz”:
  SET num = 1
  INPUT num
  WHILE num < 100:
    IF num divisible by 3:
      IF num divisible by 5:
        PRINT ‘Fizz Buzz’
      ELSE:
        PRINT ‘Fizz’
    ELSE:
      IF num divisible by 5:
        PRINT ‘Buzz’
      ELSE:
        PRINT num
```

Summary

- **Algorithms** are detailed instructions, written in a formal language, that describe the solution to a problem
- Common programming elements of algorithms are:
  - **Variables** to easily store, reference, and modify values
  - **Conditional statements** to make decisions in the algorithm flow
  - **Loops** to repeatedly execute certain steps
- Algorithms can be concisely and visually represented as **flowcharts**
- Algorithms can be represented with **pseudocode** to resemble an implementation as a computer program → more intuitive to design a flowchart first and then translate it into pseudocode.

The Center of Applied Data Science

E: info@thecads.org

W : www.thecads.org