EXPLORE AI ACADEMY

Programmatic thinking

Flowcharts and pseudocode

Programmatic thinking tools

Recap:

Algorithms

Specific procedures to solve problems in terms of the **required actions** and the order in which these actions are executed.

Operators are an important tool that algorithms use to **make decisions**.

Operators

Comparison operators are used to **compare numbers** or **strings** to perform the evaluation within a boolean expression and **boolean operators** as conjunctions to **combine** (or **exclude**) **statements** in a boolean expression.

Next up:

Flowcharts

Pseudocode

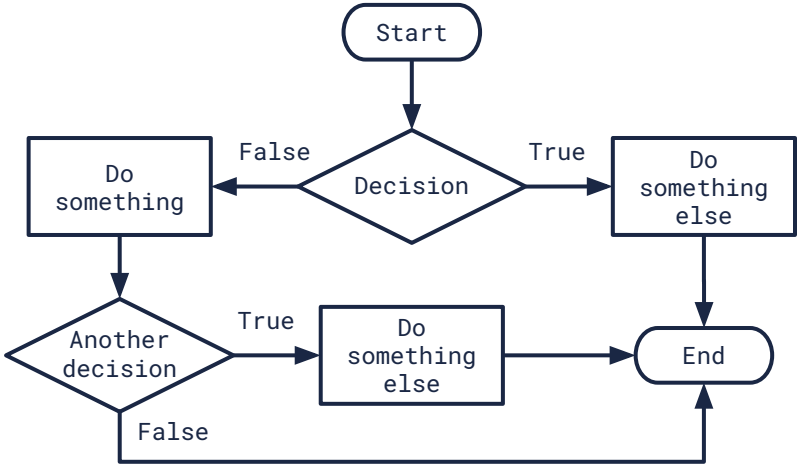
Conditional statements

Flowcharts

> We use flowcharts (aka flow diagrams) to visually **represent the flow of control** of our logic, algorithms, pseudocode, and conditional statements.

We can use a flowchart to **visualise the logic** of an **algorithm** before we even start coding. It may also be useful to map out a flowchart before writing **pseudocode**!

Example of a flowchart:



Flowchart symbols

We use different symbols in a flowchart to **represent different actions** or **steps** in a process.

Terminator

The terminator, also known as the start/end or terminal symbol, represents the **start or end of the process** flow.

Decision

The decision or branch point symbol is where we have our boolean expression. It is the point where a **decision needs to be made** with the outcome of either true or false.

Process

The process or action symbol **represents a single step** or **function**, or an entire **sub-process** within the overall process flow.

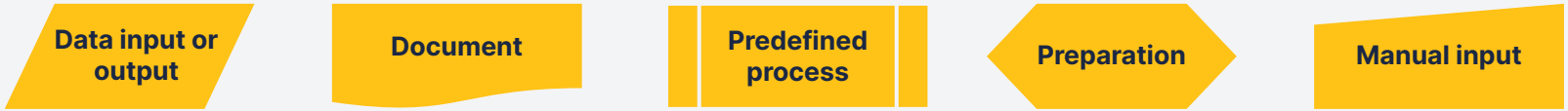
Flow arrow

The flow arrow **connects** the different symbols and **indicates the relationships** between them.

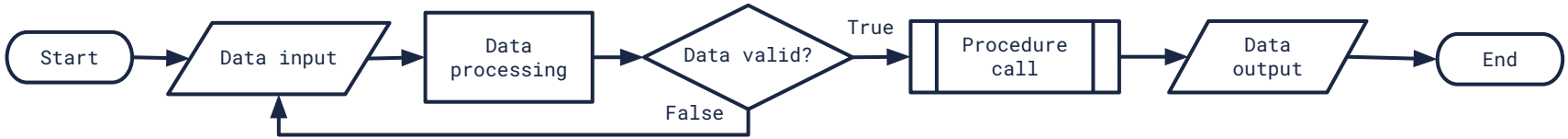
Flowchart symbols

> There are other flowchart symbols that are not often used in problem solving but rather to represent **data processing**.

Some data processing flowchart symbols:



Example of a data processing flowchart:



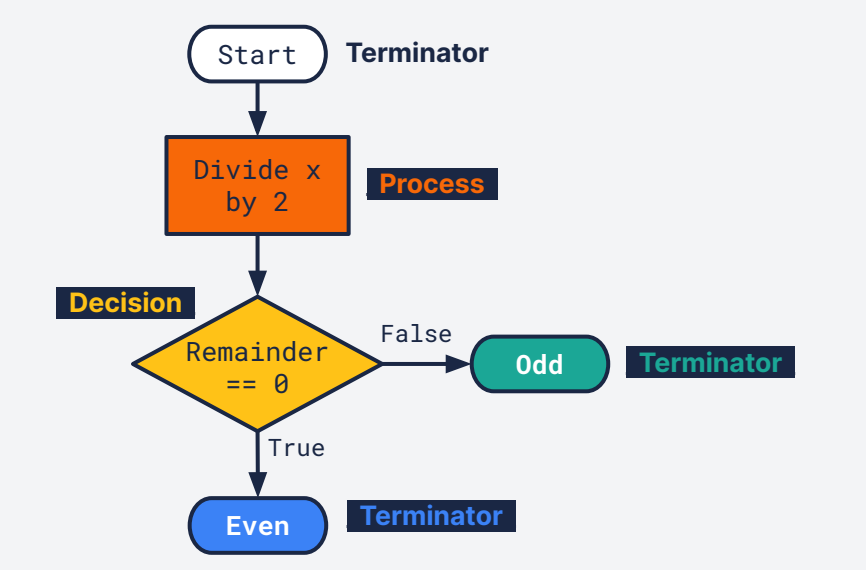
Flowchart example

Let's create a flowchart that represents the following:
Determine whether a number x is an even or odd number.

What we need to do (i.e., the algorithm):
Divide x by 2. If the remainder is equal to 0 then the number is even, and if the remainder is not equal to 0 then the number is odd.

The steps:

- Process: divide x by 2.
- Decision: is the remainder equal to zero or not?
- Output 1: the remainder is equal to zero, so the number is even.
- Output 2: the remainder is not equal to zero, so the number is odd.



We could also have used the **modulus (MOD or %) function**. How would this change the flowchart?

Alternative solution

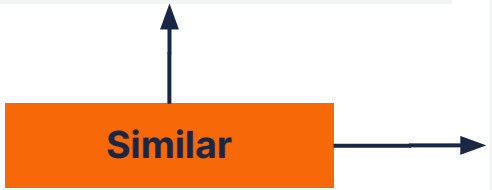
Pseudocode

We can describe a sequence of steps and actions in a plain natural language called **pseudocode**. These are step-by-step descriptions for an algorithm using short but descriptive phrases.

Because pseudocode is written in natural language, **some variation is often observed** in the syntax.

To ensure that **indents** are clear, dashes, lines, or dots are often used in place of the spaces/tabs.

```
IF decision THEN
- Do something
ELSE
- Do something
- IF another decision THEN
- - Do something
```




```
Start

If decision then
- Do something
Else
- Do something
- If another decision then
- - Do something
- End if
End if

End
```

Pseudocode example

Let's again consider the following example:
Determine whether a number x is an even or odd number.

Considering the alternative solution (using the modulus function), the algorithm is:
If x modulus 2 is equal to 0, then the number is even, else the number is odd.

We can represent the output in the following ways:

01. If the number is even the output y is equal to "Even", if the number is odd the output y is equal to "Odd".
02. If the number is even the output y is equal to 1, if the number is odd the output y is equal to 0.

If x % 2 == 0, then y = "Even", else y = "Odd"

```
If x % 2 == 0 then
- y = "Even"
Else
- y = "Odd"
End if
```

If x % 2 == 0, then y = 1, else y = 0

```
IF x % 2 == 0 THEN
- y = 1
ELSE
- y = 0
END IF
```

Pseudocode components

```
if x % 2 == 0 then
  y = 1
else
  y = 0
```

Conditional statement

Represents decision-making by setting specific conditions.

```
if x % 2 == 0 then
  y = 1
else
  y = 0
```

Comparison operator

Compares numbers or strings to perform the evaluation within a boolean expression.

```
if x % 2 == 0 then
  y = 1
else
  y = 0
```

Boolean expression

A statement that results in an answer of either True or False.

```
if x % 2 == 0 then
  y = 1
else
  y = 0
```

Assignment operators

Assigns the value on the right to the variable on the left of the operator.

```
if x % 2 == 0 then
    y = 1
else
    y = 0
```

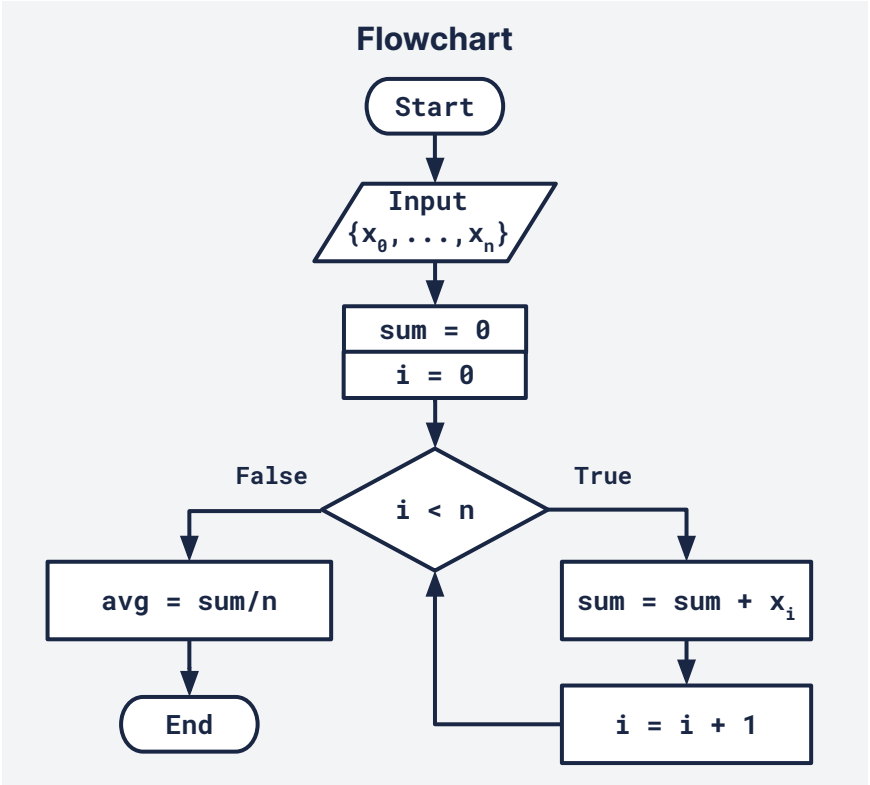
Indents

Groups statements that are intended to be executed together.

Flowchart and pseudocode example

Example:
Calculate the average of a list of **n** numbers.

1. Our input is the list of **n** numbers, where each value is $\mathbf{x_i}$; we make use of set notation: $\{x_0, x_1, x_2, \dots, x_{n-1}\}$.
2. We need to keep track of the sum of the values (**sum**) and the $\mathbf{i}^{th}$ number in the list.
3. We check if the value of **i** is smaller than the length of the list.
4. When the condition is True, we calculate **sum** by adding the number $\mathbf{x_i}$ in the list to the previous sum.
5. We also add one to our position value.
6. We repeat steps 3 to 5 until the condition is False.
7. When the condition is False, we divide the sum by the list length, and the output is the average (**avg**).



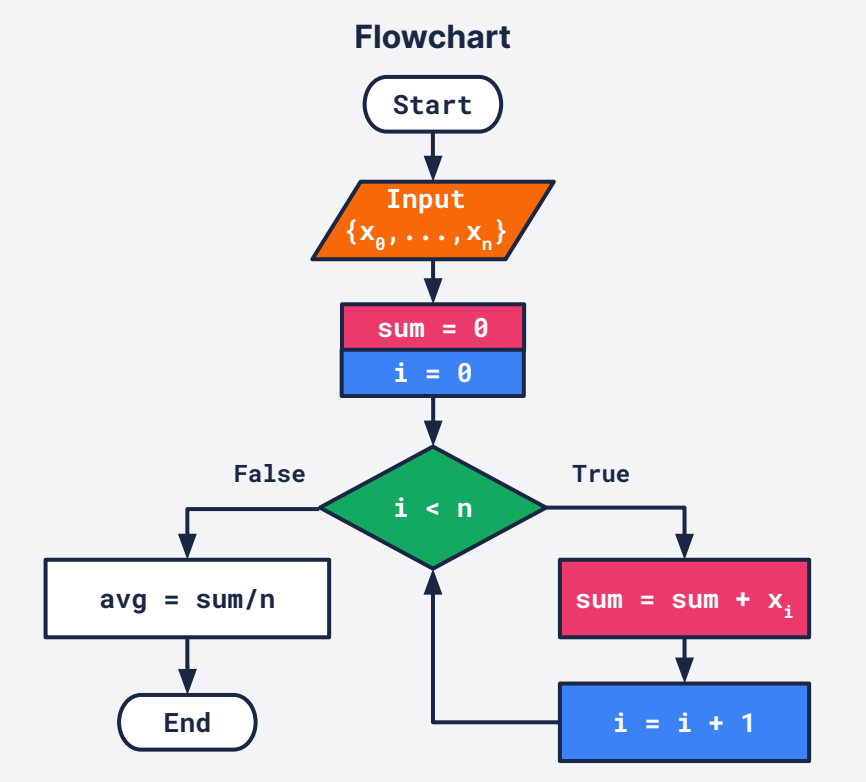
Flowchart and pseudocode example

Example:
Calculate the average of a list of **n** numbers.

Let's consider the following list: 2 | 5 | 6 | 8

n = 4	sum = 0	i = 0

x_0 = 2	0 < 4 ? True	
	sum = 0 + 2 = 2	i = 0 + 1 = 1



Flowchart and pseudocode example

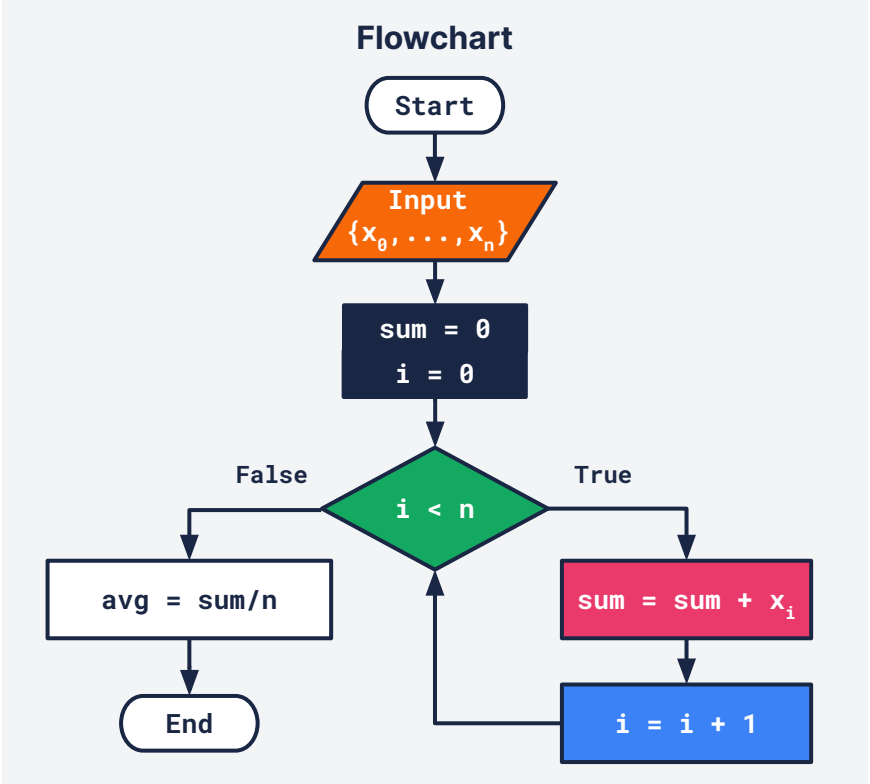
Example:
Calculate the average of a list of **n** numbers.

Let's consider the following list: 2 | 5 | 6 | 8

n = 4	sum = 0	i = 0

x_0 = 2	0 < 4 ? True	
	sum = 0 + 2 = 2	i = 0 + 1 = 1

x_1 = 5	1 < 4 ? True	
	sum = 2 + 5 = 7	i = 1 + 1 = 2



Flowchart and pseudocode example

Example:
Calculate the average of a list of **n** numbers.

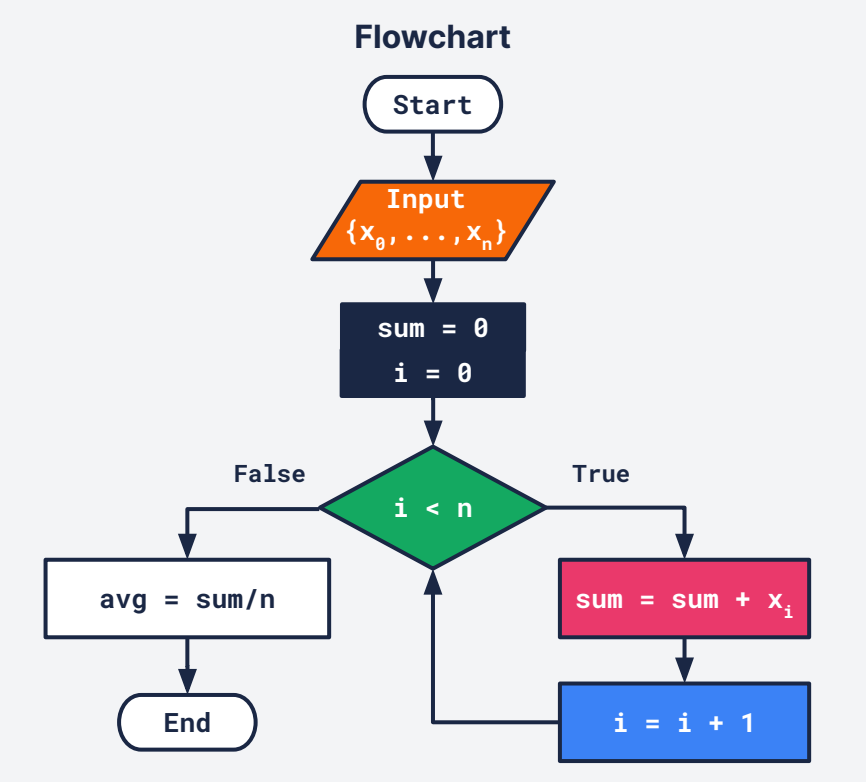
Let's consider the following list: 2 | 5 | 6 | 8

n = 4	sum = 0	i = 0

x_0 = 2	0 < 4 ? True	
	sum = 0 + 2 = 2	i = 0 + 1 = 1

x_1 = 5	1 < 4 ? True	
	sum = 2 + 5 = 7	i = 1 + 1 = 2

x_2 = 6	2 < 4 ? True	
	sum = 7 + 6 = 13	i = 2 + 1 = 3



Flowchart and pseudocode example

Example:
Calculate the average of a list of **n** numbers.

Let's consider the following list: 2 | 5 | 6 | 8

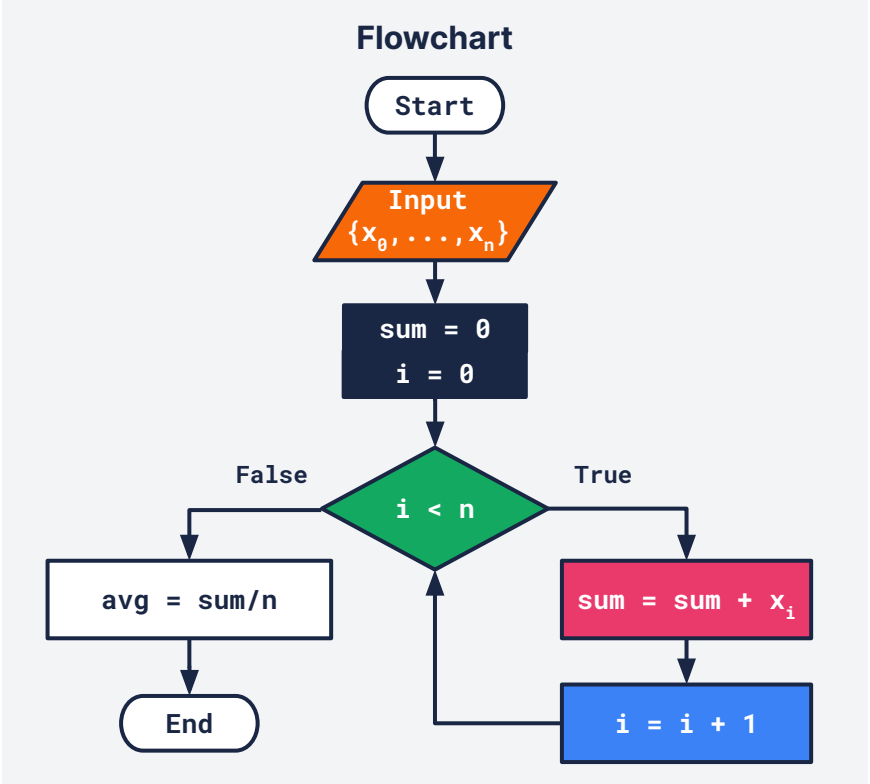
n = 4	sum = 0	i = 0

...

x_1 = 5	1 < 4 ? True	
	sum = 2 + 5 = 7	i = 1 + 1 = 2

x_2 = 6	2 < 4 ? True	
	sum = 7 + 6 = 13	i = 2 + 1 = 3

x_3 = 8	3 < 4 ? True	
	sum = 13 + 8 = 21	i = 3 + 1 = 4

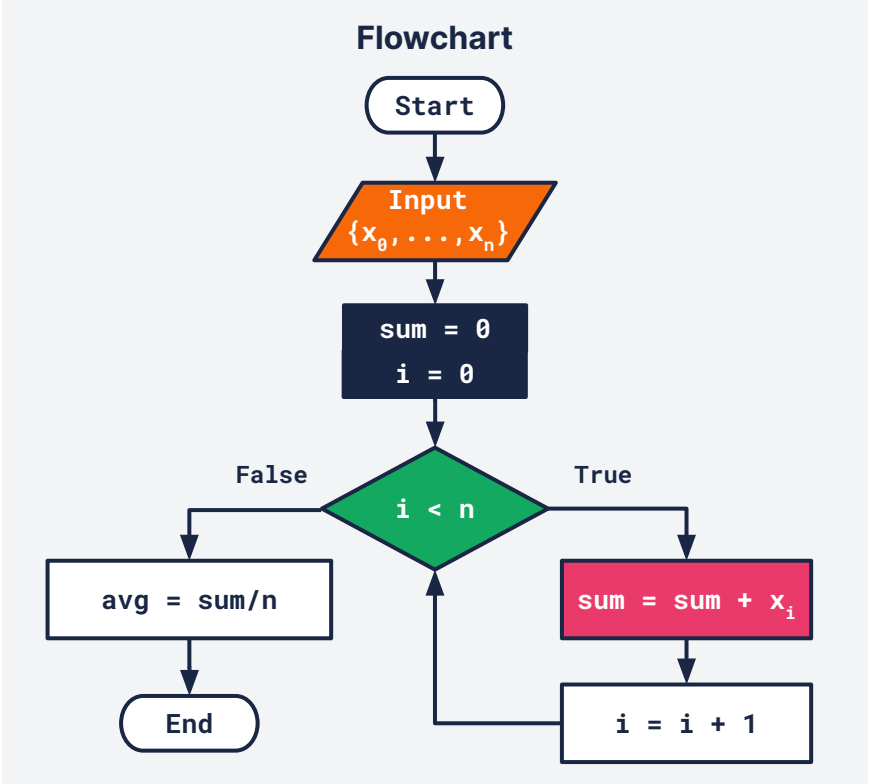


Flowchart and pseudocode example

Example:
Calculate the average of a list of **n** numbers.

Let's consider the following list: 2 | 5 | 6 | 8 | [ ]

```
n = 4        sum = 0          i = 0
...
x2 = 6       2 < 4 ? True
             sum = 7 + 6 = 13     i = 2 + 1 = 3
x3 = 8       3 < 4 ? True
             sum = 13 + 8 = 21    i = 3 + 1 = 4
             4 < 4 ? False
             avg = 21 / 4 = 5.25
```



Flowchart and pseudocode example

Example:
Calculate the average of a list of **n** numbers.

From the flowchart, we know that we need to repeat the condition x_i **<** **n** until it is False.

This is a **conditional loop** called a while loop. A conditional loop repeats a block of code as long as the condition is met.

```
Start

Input {x0,x1, x2, ... , xn}
sum = 0
i = 0

While i < n do
- sum = sum + xi
- i = i + 1
End while

avg = sum/n

End
```