EXPLORE AI ACADEMY

Programmatic thinking

Conditional statements

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. **Boolean operators** are used as conjunctions to **combine** (or **exclude**) **statements** in a boolean expression.

Flowcharts

Used to **visually represent** the **flow of control** of logic, algorithms, pseudocode, and conditional statements.

Pseudocode

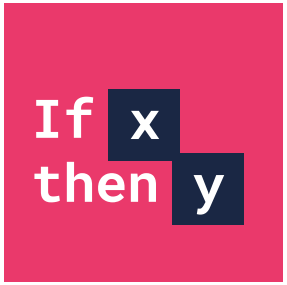
A **sequence of steps and actions** in plain natural language – these are step-by-step descriptions for an algorithm using short but descriptive phrases.

Next up: **Conditional statements**

Conditional statements

> Conditional statements allow us to **represent decision-making** by setting specific conditions.

A statement of "`If x, then y`" is a **conditional statement**, where "**x**" is the **hypothesis** (or **condition**) and "**y**" the **conclusion** (or **consequent**).



An important part of formal logic is understanding **how to use and interpret conditional statements**. Furthermore, conditional statements play a critical role in **solving problems** and **writing good code**.

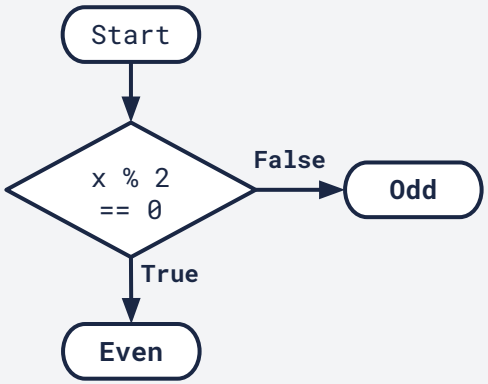
Conditional statement example

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

We've already seen that we can visually represent this as a flowchart and write down the steps in pseudocode. However, we may not have recognised that we were already using a **conditional statement**.

Using programmatic tools **together allows us to represent a problem and its solution more effectively**.

Flowchart



Pseudocode

```
Start

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

Conditional statement example

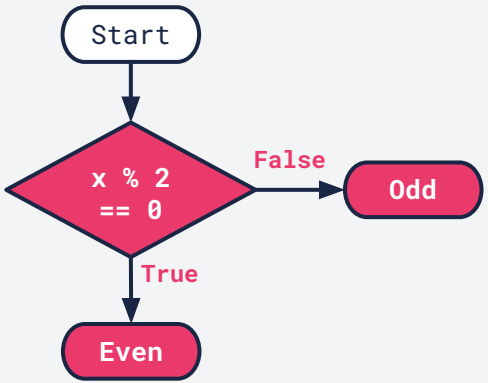
Let's again consider the following example:
Determine whether a number x is an even or odd number.

We've already seen that we can visually represent this as a flowchart and write down the steps in pseudocode. However, we may not have recognised that we were already using a **conditional statement**.

Using programmatic tools **together allows us to represent a problem and its solution more effectively**.

This is an example of the same **if-else statement** represented in two different ways.

Flowchart



Pseudocode

```
Start

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

If statement

> The if statement is the **fundamental decision-making statement**. Specific code is executed when the condition is met; if not, nothing is executed.

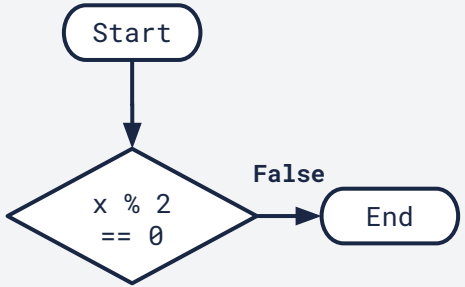
Pseudocode

```
Start

If x % 2 == 0 then
  - what needs to happen when the
    condition is true
End if

End
```

Flowchart



We start the process and set the decision point, which is the condition x % 2 == 0.

If statement

> The if statement is the **fundamental decision-making statement**. Specific code is executed when the condition is met; if not, nothing is executed.

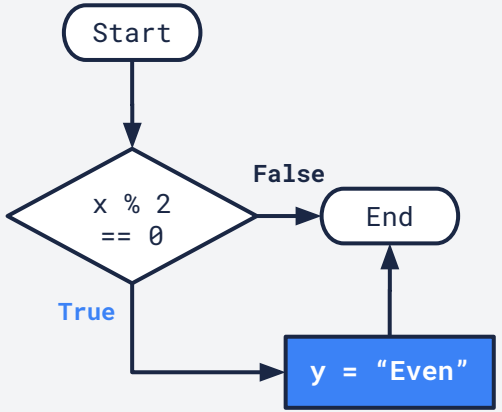
Pseudocode

```
Start

If x % 2 == 0 then
  y = “Even”
End if

End
```

Flowchart



We start the process and set the decision point which is the condition x % 2 == 0.

When the condition is True, then y is equal to “Even”.

If statement

> We can also set the output to a default value up front so that the output still has a value even when the condition is False.

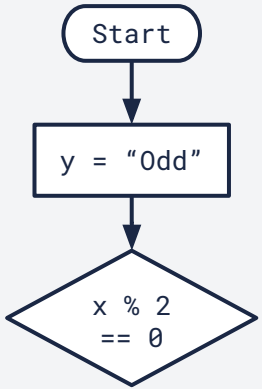
Pseudocode

```
Start


If x % 2 == 0 then
  - what needs to happen when the
    condition is true
End if

End
```

Flowchart



We start the process and set y to being equal to “Odd” by default.

The condition is still x % 2 == 0.

If statement

> We can also set the output to a default value up front so that the output still has a value even when the condition is False.

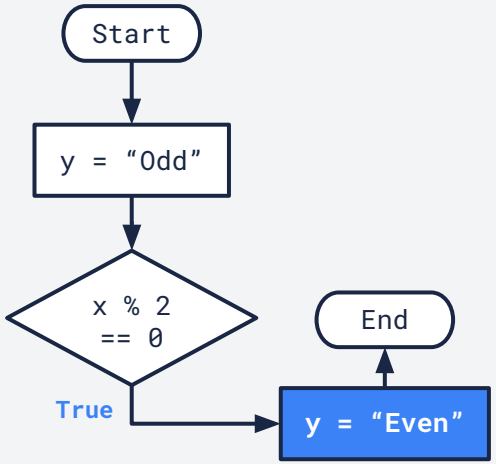
Pseudocode

```
Start


If x % 2 == 0 then
  y = “Even”
End if

End
```

Flowchart



We start the process and set y to being equal to “Odd” by default.

The condition is still x % 2 == 0.

When the condition is True, then y is equal to “Even”.

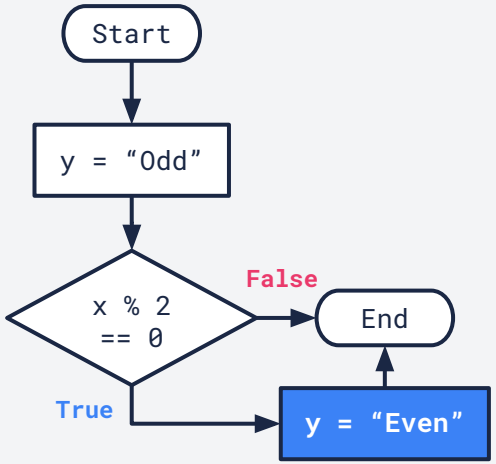
If statement

> We can also set the output to a default value up front so that the output still has a value even when the condition is False.

Pseudocode

```
Start
y = “Odd”
If x % 2 == 0 then
  y = “Even”
End if
End
```

Flowchart



We start the process and set y to being equal to “Odd” by default.

The condition is still x % 2 == 0.

When the condition is True, then y is equal to “Even”.

Now, if the condition is False, then y is equal to the default value of y, which is “Odd”.

If-else statement

> An if-else statement can be used in order to specify a block of code to be executed if the condition in the if statement is false, without setting a default value up front.

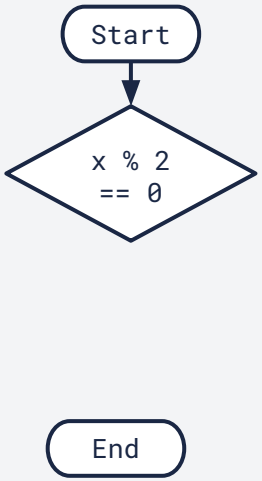
Pseudocode

```
Start

If x % 2 == 0 then
  - what needs to happen when the
    condition is true
Else
  - what needs to happen when the
    condition is false
End if

End
```

Flowchart



The condition is still x % 2 == 0.

If-else statement

An if-else statement can be used in order to specify a block of code to be executed if the condition in the if statement is false, without setting a default value up front.

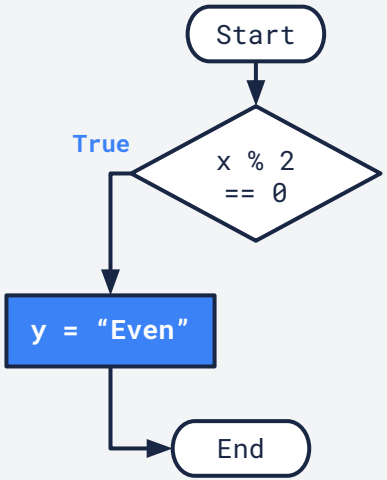
Pseudocode

```
Start

If x % 2 == 0 then
  - y = “Even”
Else
  - what needs to happen when the
    condition is false
End if

End
```

Flowchart



The condition is still x % 2 == 0.

When the condition is True, then y is equal to “Even”.

If-else statement

An if-else statement can be used in order to specify a block of code to be executed if the condition in the if statement is false, without setting a default value up front.

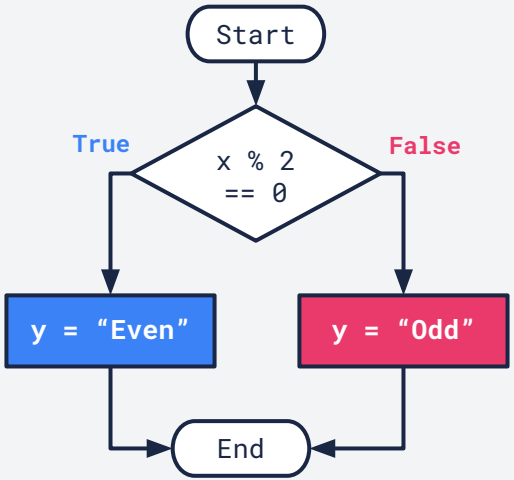
Pseudocode

```
Start

If x % 2 == 0 then
- y = “Even”
Else
- y = “Odd”
End if

End
```

Flowchart



The condition is still x % 2 == 0.

When the condition is True, then y is equal to “Even”.

When the condition is False, then y is equal to “Odd”.

Nested if statement

> We use nested if statements when we must decide on a combination of conditions before deciding the next action or final output.

Example:
Since we can only calculate the modulus of numbers, let's add another condition:

*Determine whether a **variable x** is an even or odd **number**.*

01. Set the condition to see if the input variable **x** is a number.

Pseudocode

```
Start

If x is a number then
- what needs to happen when
the condition is true



End if

End
```

Nested if statement

> We use nested if statements when we must decide on a combination of conditions before deciding the next action or final output.

Example:
Since we can only calculate the modulus of numbers, let's add another condition:

*Determine whether a **variable x** is an even or odd **number**.*

01. Set the condition to see if the input variable **x** is a number.

Pseudocode

```
Start

If x is a number then
  - what needs to happen when
    the condition is true

End if

End
```

02. When the condition is True, then we determine whether it is even or odd.

Nested if statement

> We use nested if statements when we must decide on a combination of conditions before deciding the next action or final output.

Example:
Since we can only calculate the modulus of numbers, let's add another condition:

*Determine whether a **variable x** is an even or odd **number**.*

01. Set the condition to see if the input variable **x** is a number.

Pseudocode

```
Start
If x is a number then
    If x % 2 == 0 then
        y = "Even"
    Else
        y = "Odd"
    End if
End if
End
```

02. When the condition is True, then we determine whether it is even or odd.

03. We use our previous if-else statement when the condition is True.

Note the indent levels.

Nested if statement

> We use nested if statements when we must decide on a combination of conditions before deciding the next action or final output.

Example:
Since we can only calculate the modulus of numbers, let's add another condition:

*Determine whether a **variable x** is an even or odd **number**.*

Pseudocode

```
Start
01. If x is a number then
    - If x % 2 == 0 then
    -   y = "Even"
    - Else
    -   y = "Odd"
    - End if
04. End if

End
```

03.

01. Set the condition to see if the input variable **x** is a number.

02. When the condition is True, then we determine whether it is even or odd.

03. We use the if-else statement from the previous example when condition 1 is True.

04. When condition 1 is False, we exit the if statement without doing anything else.

Nested if statement

01. The flowchart to determine if variable x is a number:

Pseudocode

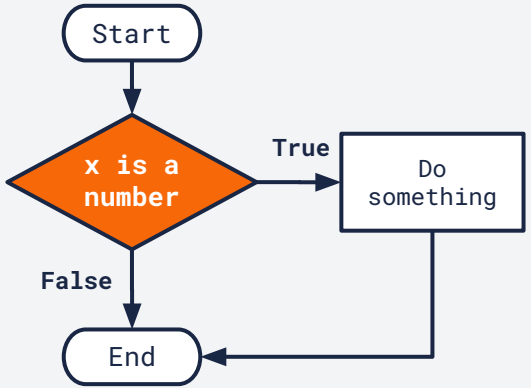
```
Start
If x is a number then

  – what needs to happen when the
    condition is true



End if
End
```

Flowchart



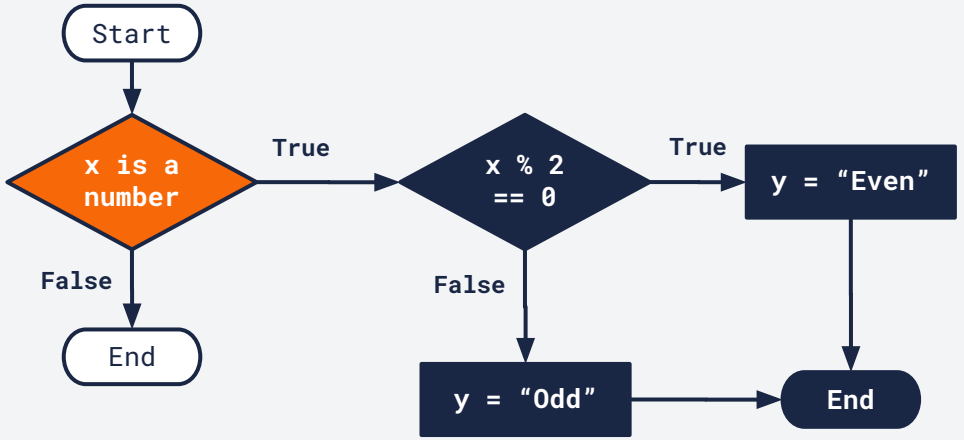
Nested if statement

02. **The flowchart of the nested if statement that checks both if x is a number and whether it's even or odd:**

Pseudocode

```
Start
If x is a number then
- If x % 2 == 0 then
- - y = “Even”
- Else
- - y = “Odd”
- End if
End if
End
```

Flowchart



Nested if statement

Example:
*Determine whether a **variable x** is a number greater than 10 and even or odd.*

We can solve this by amending the first condition with a **boolean operator**.

However, now we can only say that the variable **x** is either not a number or smaller than 10, i.e. we don't know the specific reason for not giving **y** as "Even" or "Odd".

Pseudocode

```
Start
If x is a number AND x > 10 then
- If x % 2 == 0 then
- - y = "Even and greater than 10"
- Else
- - y = "Odd and greater than 10"
- End if
Else
- y = "Not a number or smaller than 10"
End if
End
```

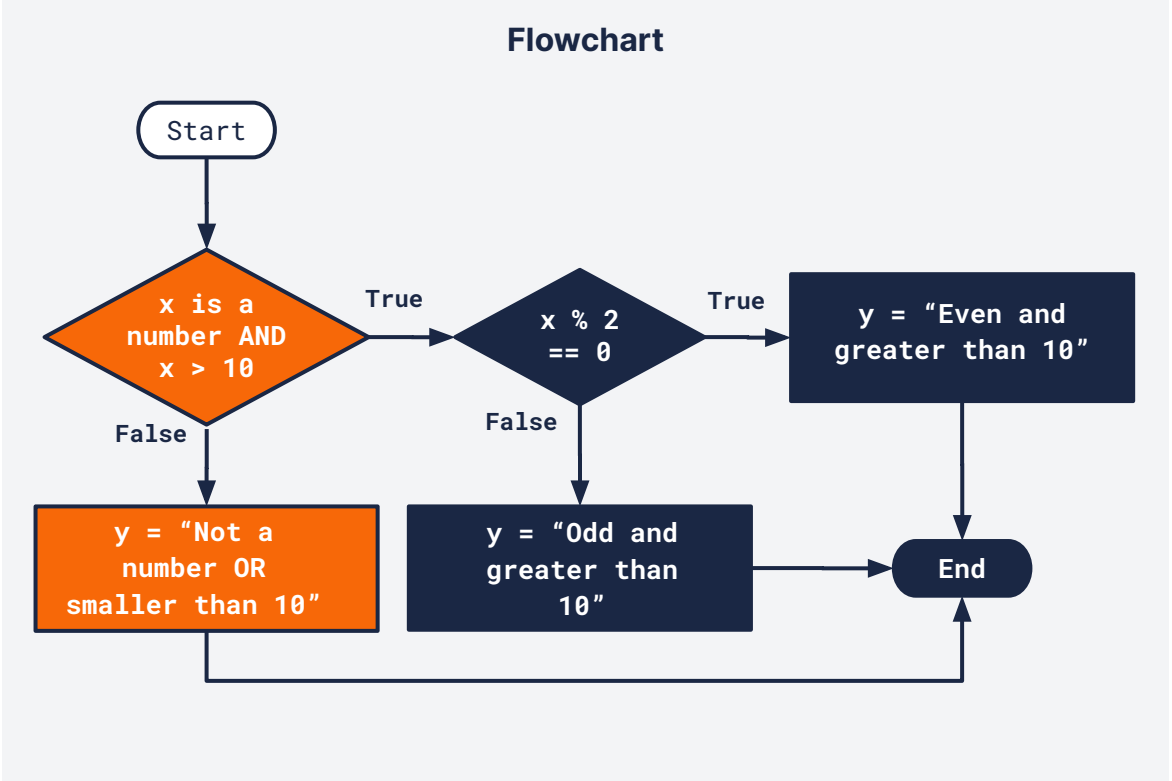
Nested if statement

Example:

*Determine whether a **variable x** is a number greater than 10 and even or odd.*

We can amend the first condition with a **boolean operator** to consider the second condition (x > 10) in conjunction with the first.

However, now we can only say that the variable **x** is either not a number or smaller than 10, i.e. we don't know the specific reason for not giving **y** as "Even" or "Odd".



Nested if statement

Example:
*Determine whether a **variable x** is a number greater than 10 and even or odd.*

We can also nest an additional if statement to test the two conditions separately, i.e. (**x** is a number) and (**x** > 10).

01. *Is variable **x** a number?*

Note the indent levels.

Pseudocode

```
Start
If x is a number then
- what needs to happen when the condition is true




Else
- y = “Not a number”
End if
End
```

Nested if statement

Example:
*Determine whether a **variable x** is a number greater than 10 and even or odd.*

We can also nest an additional if statement to test the two conditions separately, i.e. (**x** is a number) and (**x** > 10).

01. *Is variable **x** a number?*

02. *Is variable **x** greater than 10?*

Note the indent levels.

Pseudocode

```
Start
If x is a number then
- If x > 10 then
- - what needs to happen when the condition is true


- Else
- - y = “Smaller than 10”
- End if
Else
- y = “Not a number”
End if
End
```

Nested if statement

Example:

*Determine whether a **variable x** is a number greater than 10 and even or odd.*

We can also nest an additional if statement to test the two conditions separately, i.e. (**x** is a number) and (**x** > 10).

01. *Is variable **x** a number?*

02. *Is variable **x** greater than 10?*

03. *Is variable **x** even or odd?*

Note the indent levels.

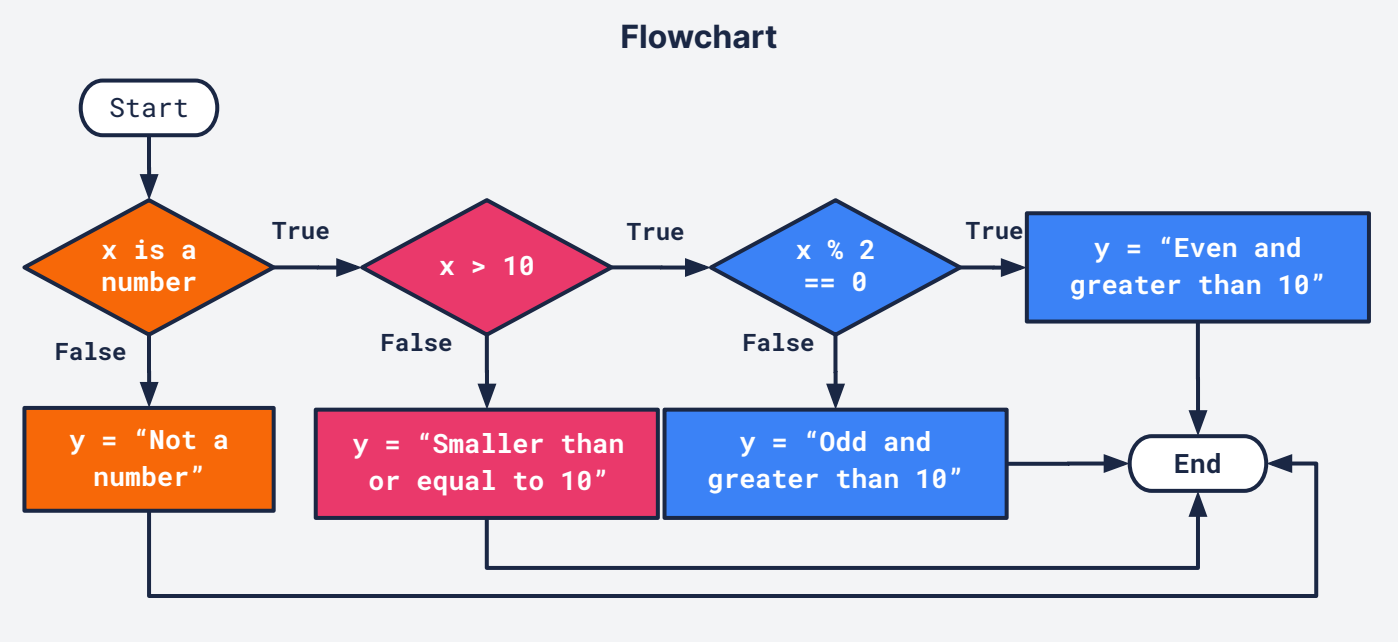
Pseudocode

```
Start
If x is a number then
- If x > 10 then
- - If x % 2 == 0 then
- - - y = “Even and greater than 10”
- - Else
- - - y = “Odd and greater than 10”
- - End if
- Else
- - y = “Smaller than or equal to 10”
- End if
Else
- y = “Not a number”
End if
End
```

Nested if statement

Example:

*Determine whether a **variable x** is a number greater than 10 and even or odd.*



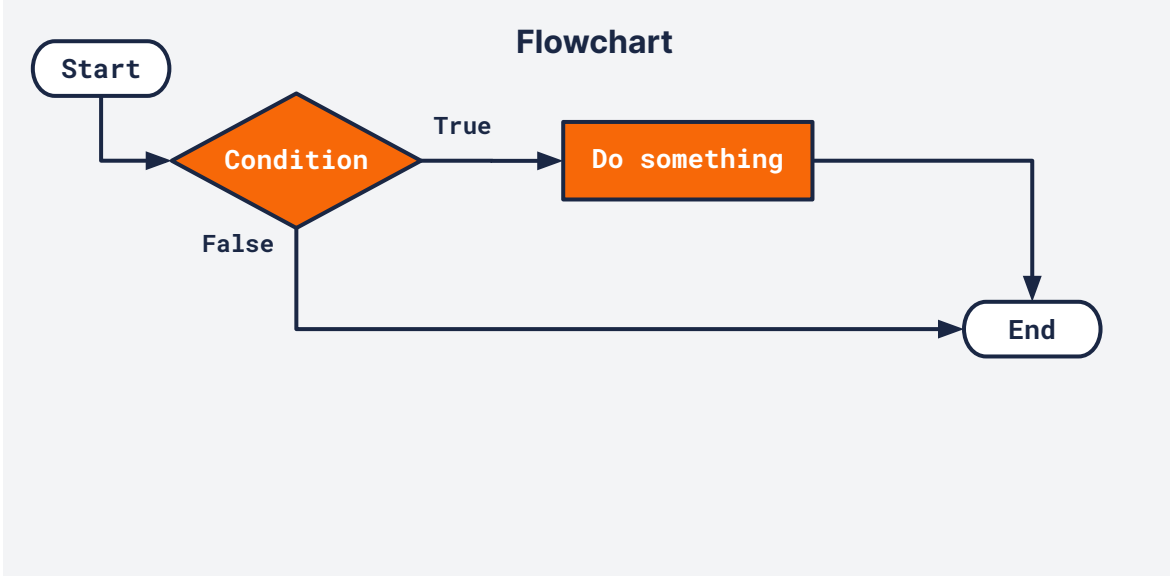
Note:
The succeeding conditions are only considered when the previous conditions are **True**.

If-else-if ladder

> We use if-else-if ladders when we need to test multiple conditions and execute different code based on which of the conditions are met. It's a way to chain multiple if statements together.

```
If (condition) then
- Do something
...
```

We expect the if statement to terminate when the condition is false.



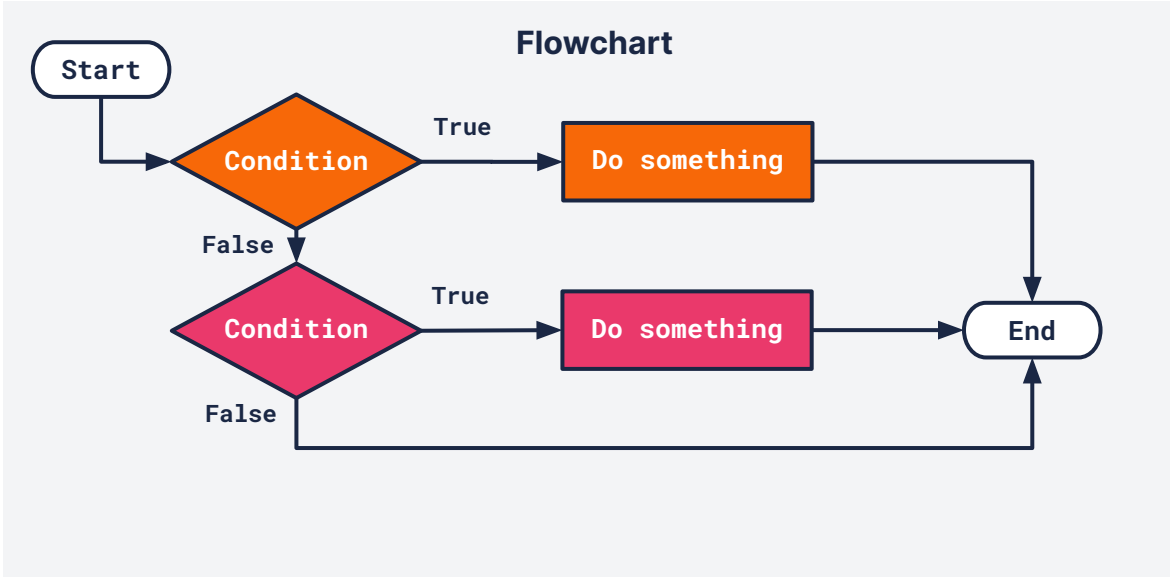
If-else-if ladder

> We use if-else-if ladders when we need to test multiple conditions and execute different code based on which of the conditions are met. It's a way to chain multiple if statements together.

```
If (condition) then
- Do something
Else if (another condition) then
- Do something different
...
```

Now, when the first if statement is false, the second condition is considered.

When the second condition is also false, then the if-else-if ladder terminates.



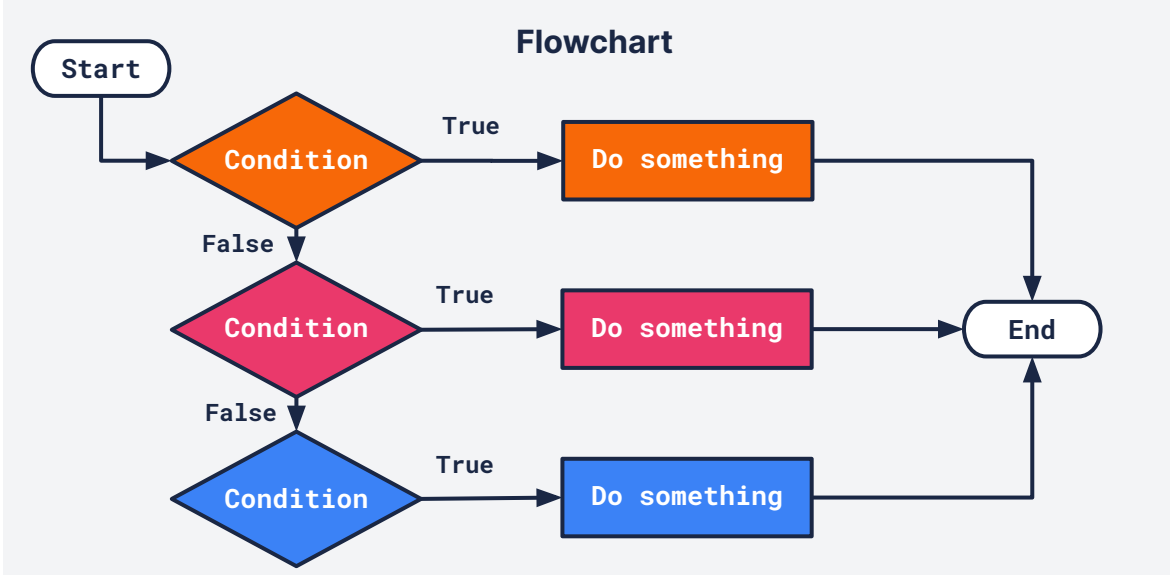
If-else-if ladder

> We use if-else-if ladders when we need to test multiple conditions and execute different code based on which of the conditions are met. It's a way to chain multiple if statements together.

```
If (condition) then
- Do something
Else if (another condition) then
- Do something different
Else if (another condition) then
- Do something different
```

If all three conditions are false, then the if-else-if ladder terminates*.

Here we only have three different conditions and consequences, but we could include many more.



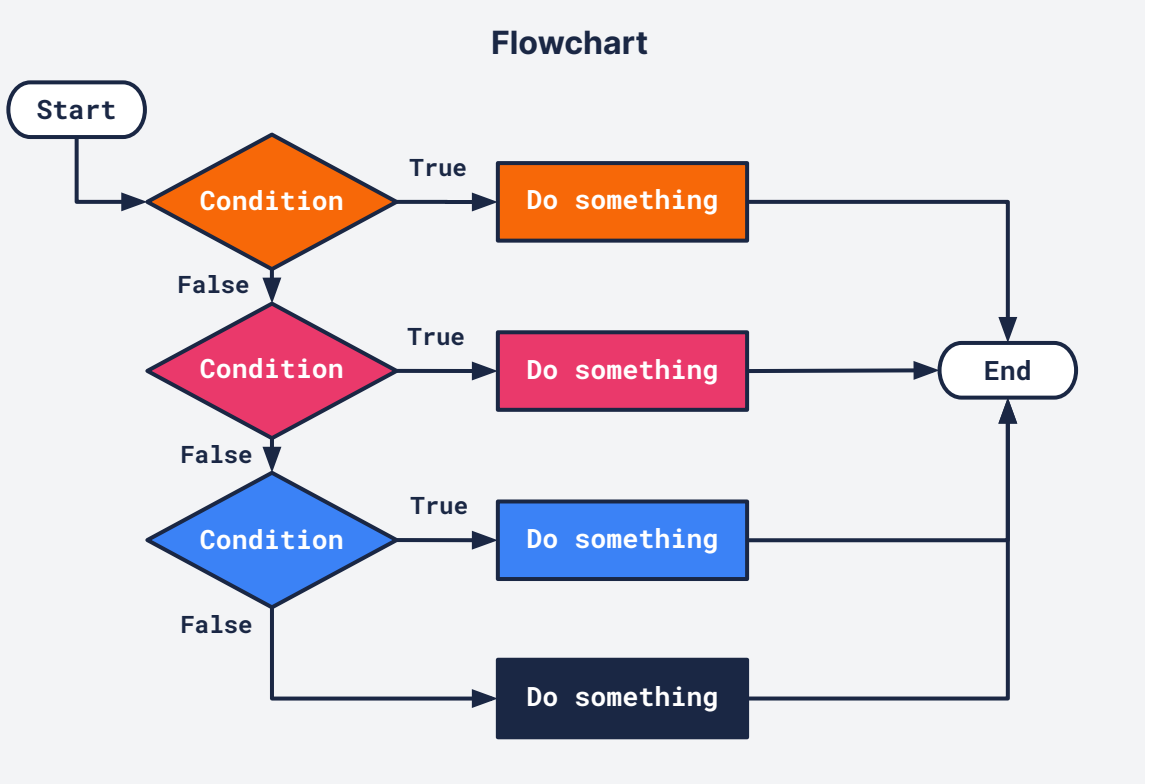
*We don't always have to show terminations. If neither a True nor False flow arrow is shown, it is assumed that it leads to the end terminator.

If-else-if ladder

```
If (condition) then
- Do something
Else if (another condition) then
- Do something different
Else if (another condition) then
- Do something different
Else
- Do something different
```

If all three conditions are false, we can also use an **else statement** to execute another process before the ladder terminates.

Note:
The succeeding conditions are only considered when the previous conditions are **False**.



If-else-if ladder

Let's see if we can rewrite the example used in the nested if statements by using an if-else-if ladder.

Example:

*Determine whether a **variable x** is a number greater than 10 and even or odd.*

01. *Is variable **x** a number or not?*

02. *Is variable **x** greater than 10 or smaller than or equal to 10?*

03. *Is variable **x** even or odd?*

Pseudocode

```
Start
If x is not a number then
- y = "Not a number"
Else if x <= 10 then
- y = "Smaller than or equal to 10"
Else if x % 2 == 0 then
- y = "Even and greater than 10"
Else
- y = "Odd and greater than 10"
End if
End
```

In order to use the if-else-if ladder, we need specific consequences for the conditions when they are **False**, i.e. the inverse of what we used in the nested if statement.

If-else-if ladder

Let's see if we can rewrite the example used in the nested if statements by using an if-else-if ladder.

Example:

*Determine whether a **variable x** is a number greater than 10 and even or odd.*

01. *Is variable **x** a number or not?*

02. *Is variable **x** greater than 10 or smaller than or equal to 10?*

03. *Is variable **x** even or odd?*