“Logic is not a body of doctrine, but a **mirror-image of the world**. Logic is transcendental.”

– Ludwig Wittgenstein

## Data overview

To investigate how we can use spreadsheet functions in data analytics, we will use two datasets; one on food crops and the other on their subsidies.

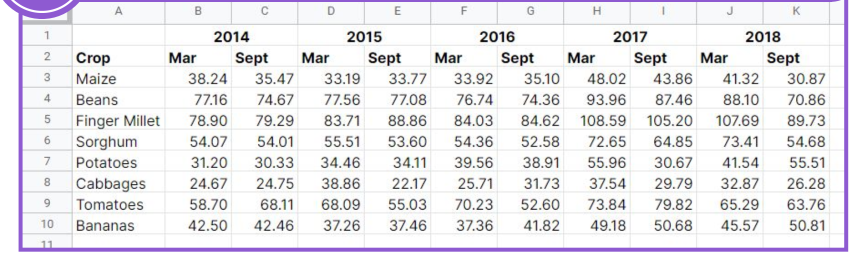
1. **Average retail market prices of selected food crops 2014-2018**

A Kenyan dataset containing prices for selected food crops during the months of March and September for the years 2014-2018.

2. **Food crops subsidies 2014-2018**

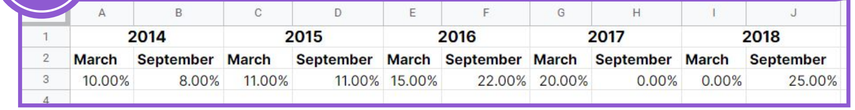
A dataset indicating the subsidies the Kenyan government provided for all food crops during the months of March and September for the years 2014-2018.

### Dataset 1



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 2014 | | 2015 | | 2016 | | 2017 | | 2018 | |
| 2 | Crop | Mar | Sept | Mar | Sept | Mar | Sept | Mar | Sept | Mar | Sept |
| 3 | Maize | 38.24 | 35.47 | 33.19 | 33.77 | 33.92 | 35.10 | 48.02 | 43.86 | 41.32 | 30.87 |
| 4 | Beans | 77.16 | 74.67 | 77.56 | 77.08 | 76.74 | 74.36 | 93.96 | 87.46 | 88.10 | 70.86 |
| 5 | Finger Millet | 78.90 | 79.29 | 83.71 | 88.86 | 84.03 | 84.62 | 108.59 | 105.20 | 107.69 | 89.73 |
| 6 | Sorghum | 54.07 | 54.01 | 55.51 | 53.60 | 54.36 | 52.58 | 72.65 | 64.85 | 73.41 | 54.68 |
| 7 | Potatoes | 31.20 | 30.33 | 34.46 | 34.11 | 39.56 | 38.91 | 55.96 | 30.67 | 41.54 | 55.51 |
| 8 | Cabbages | 24.67 | 24.75 | 38.86 | 22.17 | 25.71 | 31.73 | 37.54 | 29.79 | 32.87 | 26.28 |
| 9 | Tomatoes | 58.70 | 68.11 | 68.09 | 55.03 | 70.23 | 52.60 | 73.84 | 79.82 | 65.29 | 63.76 |
| 10 | Bananas | 42.50 | 42.46 | 37.26 | 37.46 | 37.36 | 41.82 | 49.18 | 50.68 | 45.57 | 50.81 |
| 11 | | | | | | | | | | | |

### Dataset 2



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2014 | | 2015 | | 2016 | | 2017 | | 2018 | |
| 2 | March | September | March | September | March | September | March | September | March | September |
| 3 | 10.00% | 8.00% | 11.00% | 11.00% | 15.00% | 22.00% | 20.00% | 0.00% | 0.00% | 25.00% |
| 4 | | | | | | | | | | |

# The IF and IFS functions

> **IF** is used to return one value if a logical expression evaluates to **TRUE** and another if it evaluates to **FALSE. IFS** evaluates **multiple conditions** and returns a value that corresponds to the **first TRUE** condition.

```
=IF(logical_expression, value_if_true, value_if_false)
=IFS(condition1, value1, [condition2, value2, …])
```

- **`logical_expression`** – An expression or cell reference containing an expression that represents some logical value (**TRUE** or **FALSE**).
- **`value_if_true`** – The value the function returns if **`logical_expression`** is **TRUE**.
- **`value_if_false`** – [OPTIONAL] The value the function returns if **`logical_expression`** is **FALSE**.
- **`condition1`** – The first condition to be evaluated. This can be a boolean, a number, or a reference to any of those.
- **`value1`** – The returned value if **`condition1`** is **TRUE**.
- **`[condition2, value2, …]`** – Additional conditions and values if the previous ones are evaluated **FALSE**.

# The IF and IFS functions

## Example use:

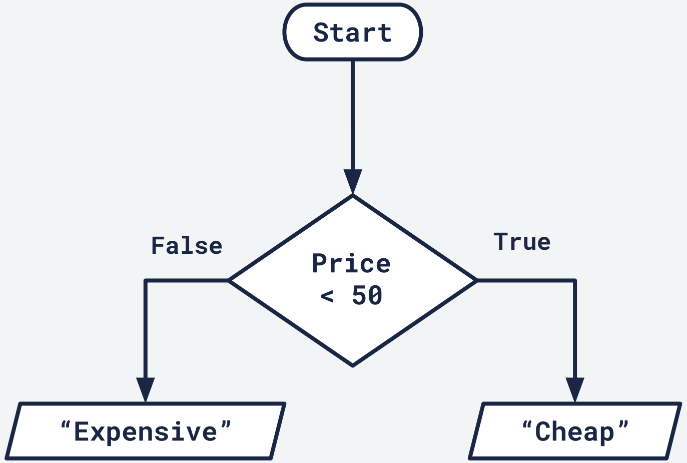
**01.** For March 2014 prices **less than KSh 50,** categorise as "cheap". **Otherwise** categorise as "expensive".

Our task is to separate our list of food crops into two categories based on their prices. Food crops that are less than KSh 50 we will tag as "cheap", while those equal to or more than KSh 50 will be tagged as "expensive".

**01.**

```
If price < 50
- Output = "Cheap"
Else
- Output = "Expensive"
```

**01.**



# The IF and IFS functions

## Example use:

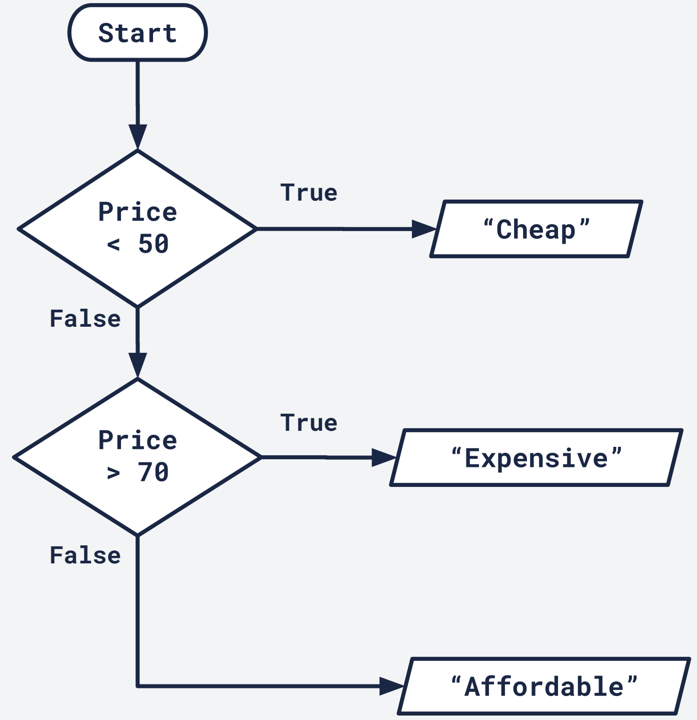
**02.** For March 2014 prices **less than KSh 50**, categorise as "cheap", **more than KSh 70** categorise as "expensive" and **the rest** categorise as "affordable".

Our second task has three categories, which means that we will have to expand the pseudocode for our **If** statement by adding an **Else if**.

**02.**

```
If price < 50
- Output = "Cheap"
Else if price > 70
- Output = "Expensive"
Else
- Output = "Affordable"
```

**02.**

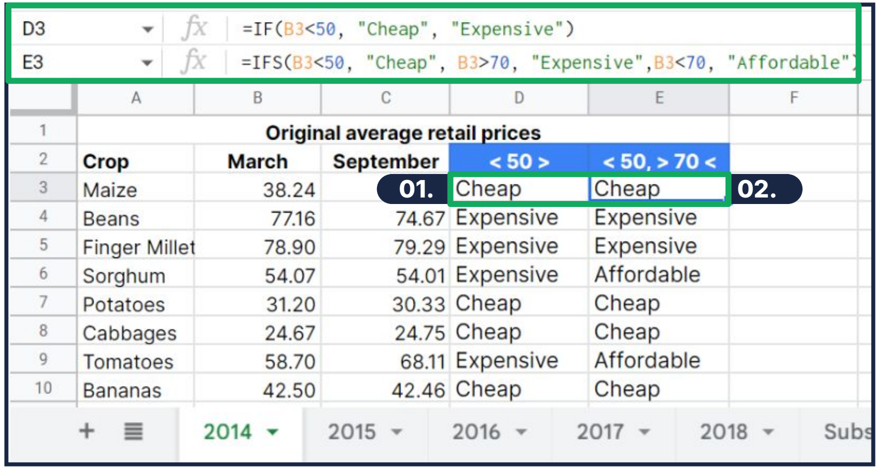


Spreadsheet functions

## The IF and IFS functions

### Example use:

01. Since the first task has only **two conditions** (“cheap” or “expensive”) we will use an **IF function** which evaluates Boolean expressions.
02. The second task has **more than two conditions** (“cheap”, “affordable”, or “expensive”) and will therefore be better evaluated with an **IFS function**.



```
D3  fx  =IF(B3<50, "Cheap", "Expensive")
E3  fx  =IFS(B3<50, "Cheap", B3>70, "Expensive",B3<70, "Affordable")
```

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | Original average retail prices | | | |
| 2 | Crop | March | September | < 50 > | < 50, > 70 < | |
| 3 | Maize | 38.24 | 01. | Cheap | Cheap | 02. |
| 4 | Beans | 77.16 | 74.67 | Expensive | Expensive | |
| 5 | Finger Millet | 78.90 | 79.29 | Expensive | Expensive | |
| 6 | Sorghum | 54.07 | 54.01 | Expensive | Affordable | |
| 7 | Potatoes | 31.20 | 30.33 | Cheap | Cheap | |
| 8 | Cabbages | 24.67 | 24.75 | Cheap | Cheap | |
| 9 | Tomatoes | 58.70 | 68.11 | Expensive | Affordable | |
| 10 | Bananas | 42.50 | 42.46 | Cheap | Cheap | |

01. Enter the formula `=IF(B3<50, "Cheap", "Expensive")` on cell D3.

02. Enter the formula `=IFS(B3<50, "Cheap", B3>70, "Expensive",B3<70, "Affordable")` on cell F3.

# The alternative IF functions

**IFERROR** and **IFNA** functions are used to catch and handle errors in a formula.

```
=IFERROR(value, [value_if_error])
=IFNA(value, value_if_na)
```

- **`value`** – The value to return if **`value`** itself is not an error (**IFERROR**) or the value to check if it is an #N/A error (**IFNA**).
- **`[value_if_error]`** – [OPTIONAL] The value returned if **`value`** is an error.
- **`value_if_na`** – The value to return if **`value`** is an #N/A error.

**IFERROR** function returns the first argument if it is not an error value, otherwise it returns the second argument if present, or a blank if the second argument is absent.

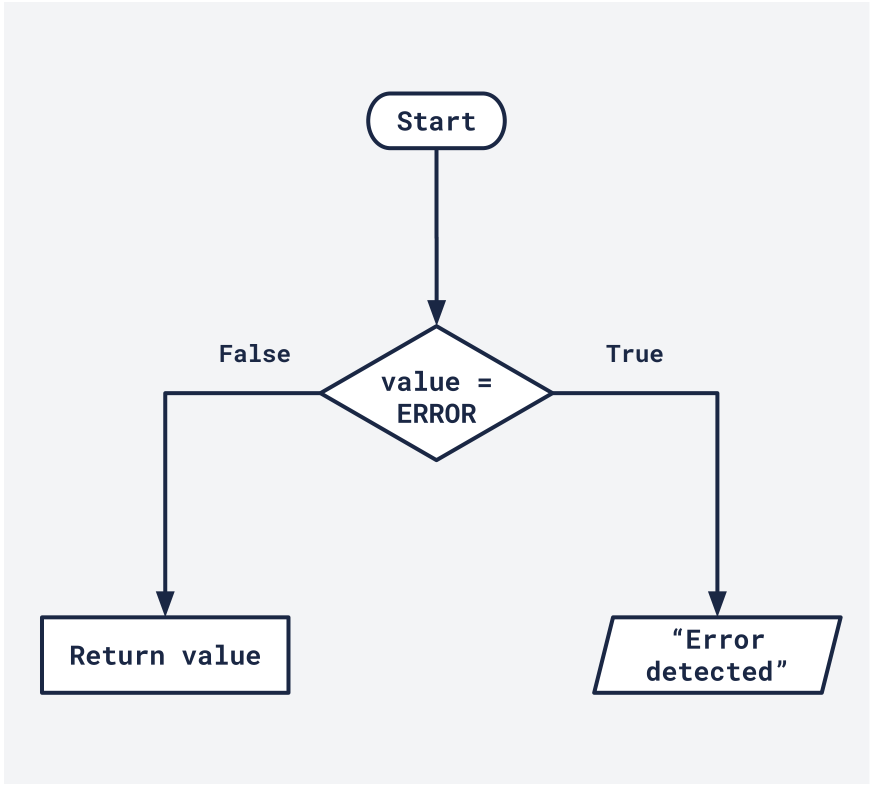
**IFNA** function evaluates a value and returns the specified value if the value is an #N/A error.

# The alternative IF functions

## Example use:

01. Detect price values that have an **error**.

```
If value = ERROR
- Output = "Error detected"
Else
- Return value
```

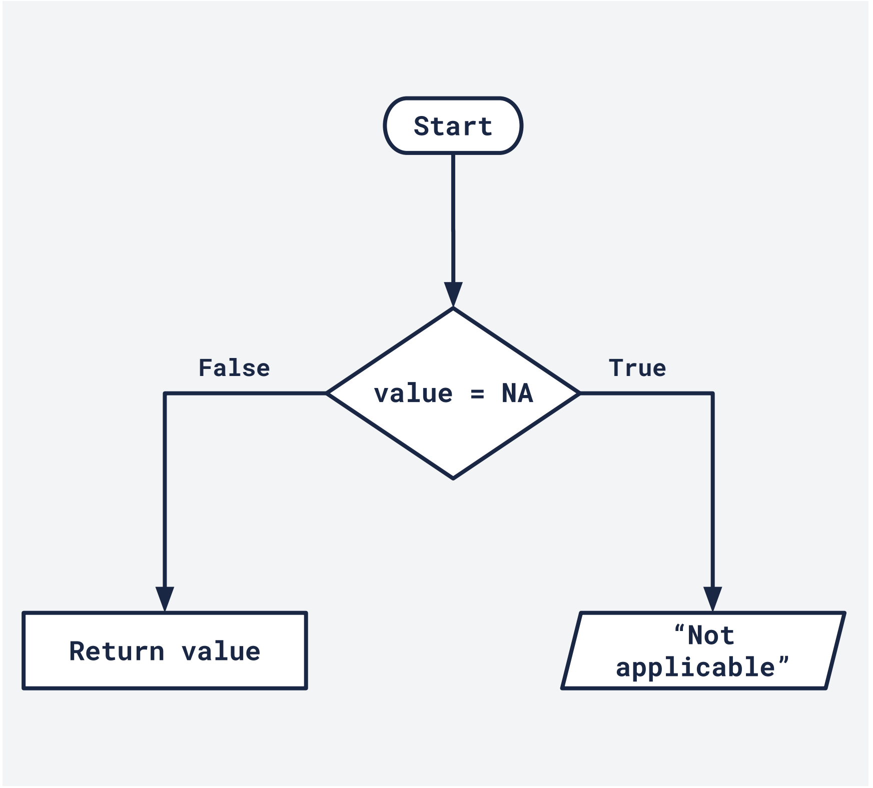


## The alternative IF functions

### Example use:

**02.** Detect price values that have values that are **not applicable**.

```
If value = NA
- Output = “Not applicable”
Else
- Return value
```



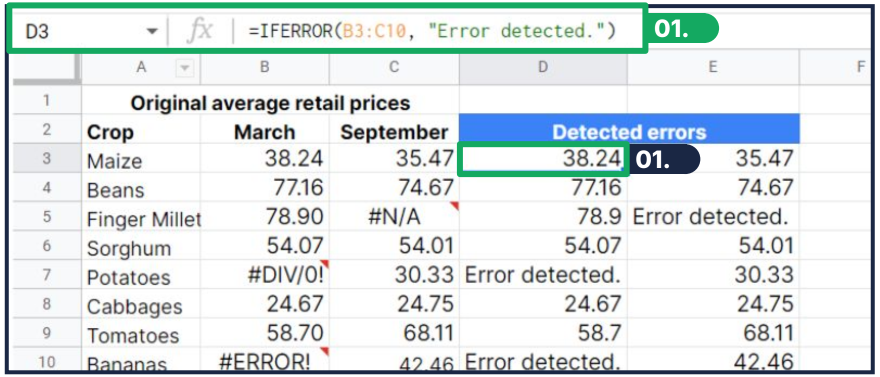
# The alternative IF functions

## Example use:

01. **IFERROR** can detect any type of error and will therefore detect **#N/A** errors among other errors.
02. **IFNA**, however, only detects **#N/A** errors and will not pick up on any other types of errors.

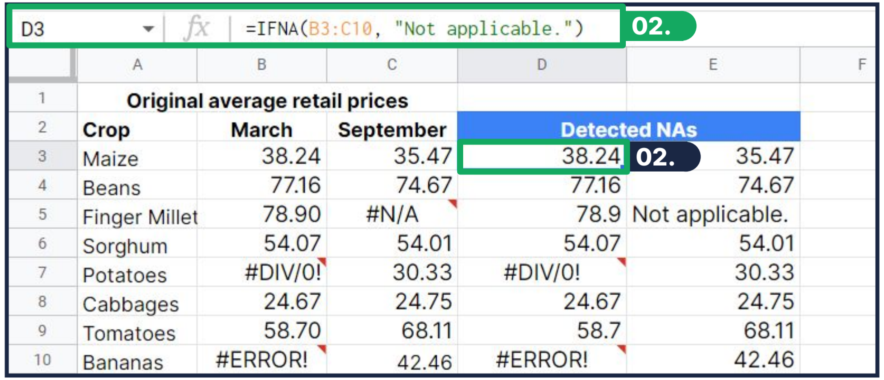
**01.** Enter the formula **`=IFERROR(B3:C10, "Error detected.")`** on cell D3 and press **ENTER**.

**02.** Enter the formula **`=IFNA(B3:C10, "Not applicable.")`** on cell D3 and press **ENTER**.



```
D3   fx   =IFERROR(B3:C10, "Error detected.")
```

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Original average retail prices | | | | | |
| 2 | Crop | March | September | Detected errors | | |
| 3 | Maize | 38.24 | 35.47 | 38.24 | 35.47 | |
| 4 | Beans | 77.16 | 74.67 | 77.16 | 74.67 | |
| 5 | Finger Millet | 78.90 | #N/A | 78.9 | Error detected. | |
| 6 | Sorghum | 54.07 | 54.01 | 54.07 | 54.01 | |
| 7 | Potatoes | #DIV/0! | 30.33 | Error detected. | 30.33 | |
| 8 | Cabbages | 24.67 | 24.75 | 24.67 | 24.75 | |
| 9 | Tomatoes | 58.70 | 68.11 | 58.7 | 68.11 | |
| 10 | Bananas | #ERROR! | 42.46 | Error detected. | 42.46 | |



```
D3   fx   =IFNA(B3:C10, "Not applicable.")
```

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Original average retail prices | | | | | |
| 2 | Crop | March | September | Detected NAs | | |
| 3 | Maize | 38.24 | 35.47 | 38.24 | 35.47 | |
| 4 | Beans | 77.16 | 74.67 | 77.16 | 74.67 | |
| 5 | Finger Millet | 78.90 | #N/A | 78.9 | Not applicable. | |
| 6 | Sorghum | 54.07 | 54.01 | 54.07 | 54.01 | |
| 7 | Potatoes | #DIV/0! | 30.33 | #DIV/0! | 30.33 | |
| 8 | Cabbages | 24.67 | 24.75 | 24.67 | 24.75 | |
| 9 | Tomatoes | 58.70 | 68.11 | 58.7 | 68.11 | |
| 10 | Bananas | #ERROR! | 42.46 | #ERROR! | 42.46 | |

# The alternative IF functions

**SUMIF** is used to **add up** the value of cells within a range that meet **a certain condition** while **SUMIFS adds up** the value of cells within a range that meet **multiple conditions**.

```
=SUMIF(range, criterion, [sum_range])
=SUMIFS(sum_range,criteria_range1,criterion1,[criteria_range2,criterion2,...])
```

- **`range`** – The range which is tested against **`criterion`**.
- **`criterion`** – The pattern or test to apply to **`range`**.
- **`sum_range`** – [OPTIONAL for **`SUMIF`** if different from **`range`**] The range to be summed.
- **`criteria_range1`** – The range to check against **`criterion1`**.
- **`criterion1`** – The pattern or test to apply to **`criteria_range1`**.
- **`criteria_range2`**, **`criterion2`**, ... – [OPTIONAL] Additional ranges and criteria to check.

# The alternative IF functions

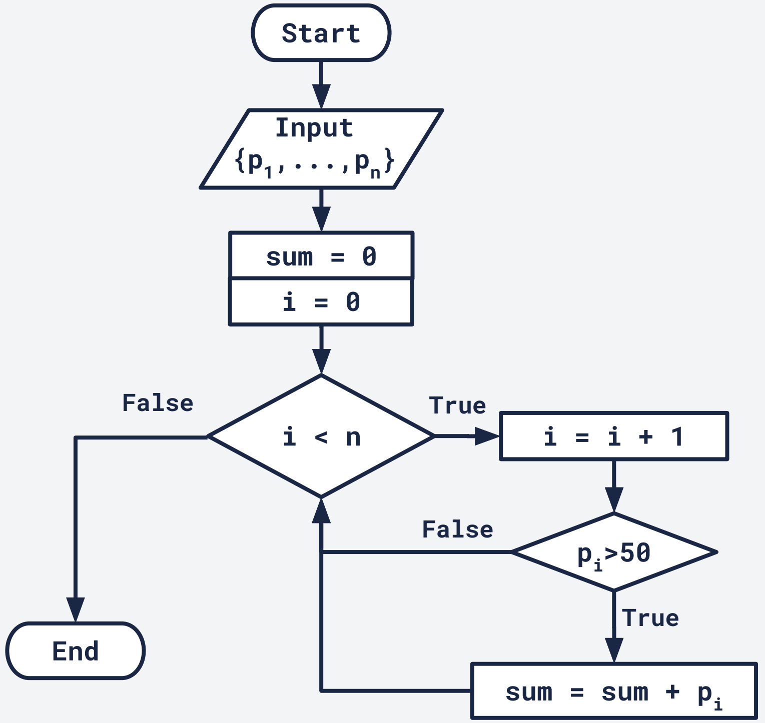
## Example use:

01. **Sum up** the prices of all food crops that were **more than KSh 50** in March 2014.

```
Input {$p_1$,...,$p_n$}
sum = 0
i = 0

While i < n do
- i = i + 1
- If $p_i$ > 50
- - sum = sum + $p_i$
```

Let the set of prices in March 2014 be represented by $\{p_1,...,p_n\}$ where **n** is the number of prices in the list.



# The alternative IF functions

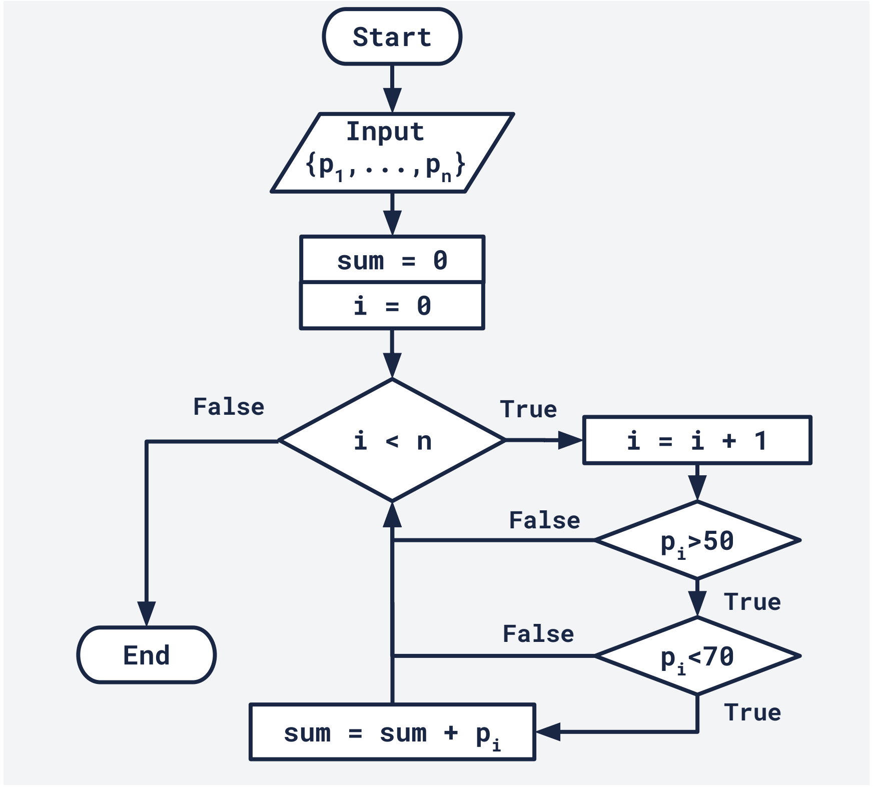
## Example use:

**02.** **Sum up** the prices of all food crops that were **more than KSh 50** but **less than KSh 70** in March 2014.

```
Input {p_1,...,p_n}
sum = 0
i = 0

While i < n do
- i = i + 1
- If p_i > 50
- - If p_i < 70
- - - sum = sum + p_i
```

Let the set of prices in March 2014 be represented by $\{p_1,...,p_n\}$ where **n** is the number of prices in the list.



# The alternative IF functions

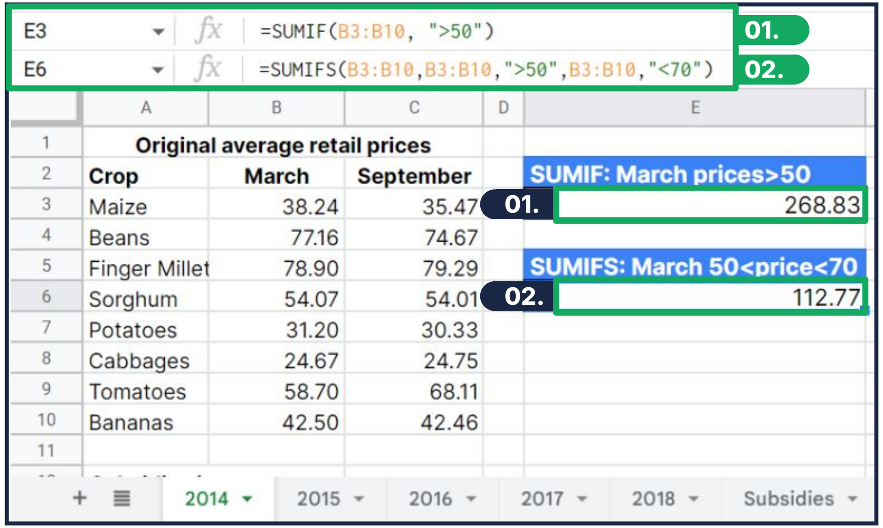
## Example use:

Our **SUMIF's `range`** will be the same as the **`sum_range`** and, therefore, **`sum_range`** will not be included in our function.

**SUMIFS** will use the same range of cells for both **`criteria_range1`** and **`criteria_range2`** as well as **`sum_range`**.

**01.** Enter the formula **`=SUMIF(B3:B10, ">50")`** on cell E3 and press **ENTER**.

**02.** Enter the formula **`=SUMIFS(B3:B10,B3:B10,">50",B3:B10,"<70")`** on cell E6 and press **ENTER**.



```
E3  fx  =SUMIF(B3:B10, ">50")          01.
E6  fx  =SUMIFS(B3:B10,B3:B10,">50",B3:B10,"<70")   02.
```

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Original average retail prices | | | | |
| 2 | Crop | March | September | | SUMIF: March prices>50 |
| 3 | Maize | 38.24 | 35.47 | 01. | 268.83 |
| 4 | Beans | 77.16 | 74.67 | | |
| 5 | Finger Millet | 78.90 | 79.29 | | SUMIFS: March 50<price<70 |
| 6 | Sorghum | 54.07 | 54.01 | 02. | 112.77 |
| 7 | Potatoes | 31.20 | 30.33 | | |
| 8 | Cabbages | 24.67 | 24.75 | | |
| 9 | Tomatoes | 58.70 | 68.11 | | |
| 10 | Bananas | 42.50 | 42.46 | | |
| 11 | | | | | |

# The alternative IF functions

**COUNTIF** function **counts cells** in a range that meet **a single condition** while **COUNTIFS** function **counts cells** in a range that meet **multiple conditions**.

```
=COUNTIF(range, criterion)
=COUNTIFS(criteria_range1,criterion1,[criteria_range2,…],[criterion2,…])
```

- `range` – The range that is tested against `criterion`.
- `criterion` – The pattern or test to apply to `range`.

- `criteria_range1` – The range to check against `criterion1`.
- `criterion1` – The pattern or test to apply to `criteria_range1`.
- `criteria_range2, criterion2...` – [OPTIONAL] Additional ranges and criteria to check.

# The alternative IF functions

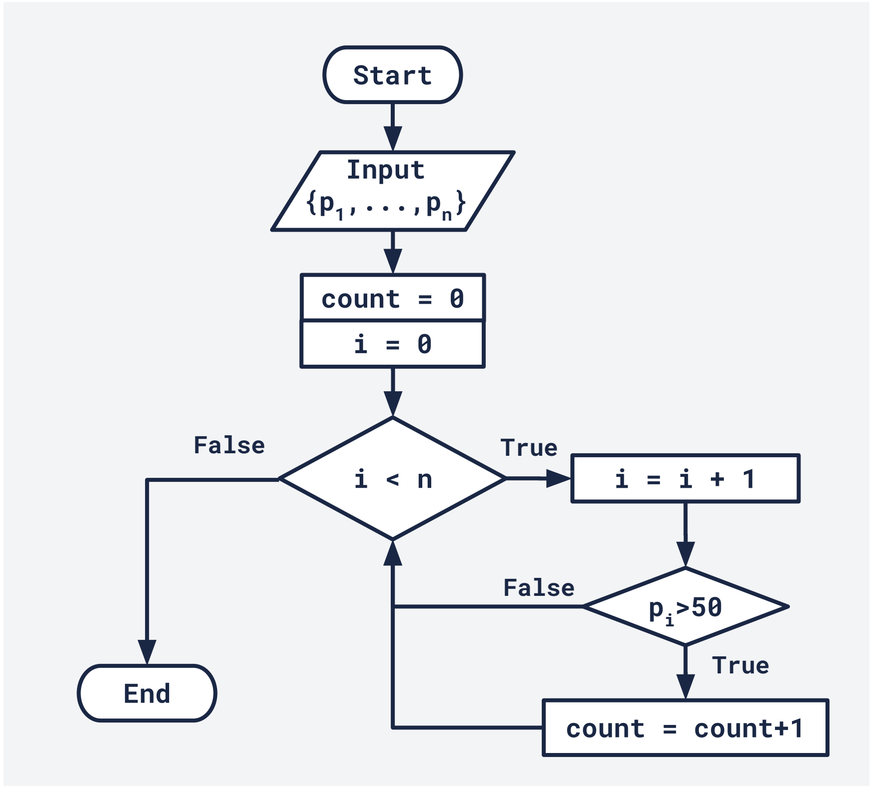
## Example use:

**01.** **Count** the number of cells containing food crop prices that were **more than KSh 50** in March 2014.

```
Input {$p_1$,...,$p_n$}
count = 0
i = 0

While i < n do
- i = i + 1
- If $p_i$ > 50
- - count = count + 1
```

We see that our *logic* remains the same as per our sum example.



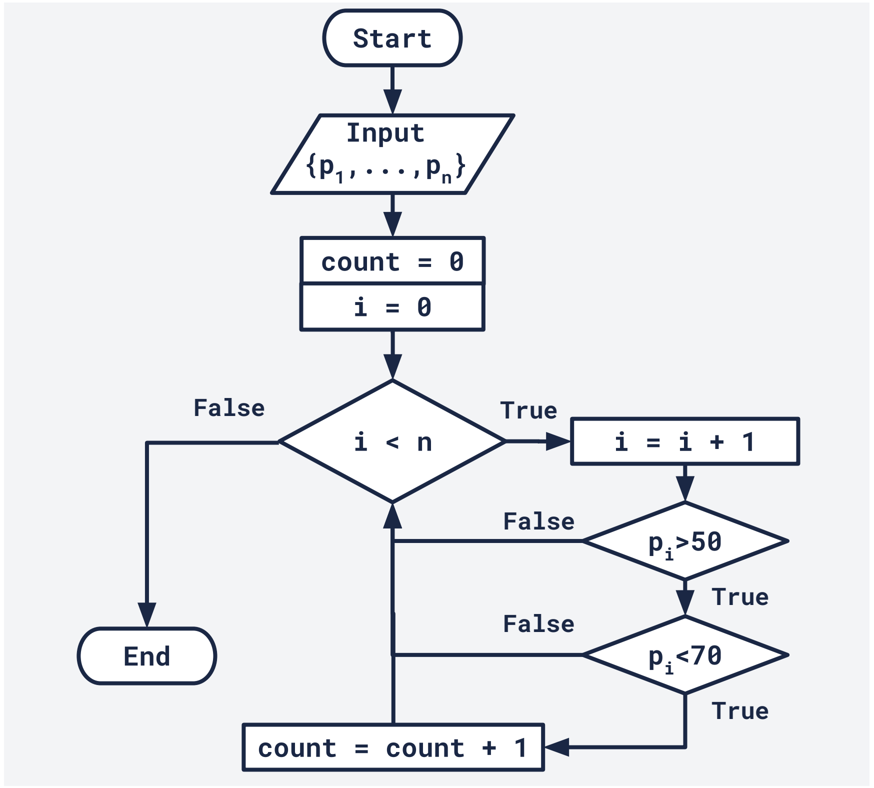
## The alternative IF functions

### Example use:

02. **Count** the number of cells containing food crop prices that were **more than KSh 50** but **less than KSh 70** in March 2014.

```
Input {$p_1$,...,$p_n$}
count = 0
i = 0

While i < n do
- i = i + 1
- If $p_i$ > 50
- - If $p_i$ < 70
- - - count = count + 1
```



Spreadsheet functions

# The alternative IF functions

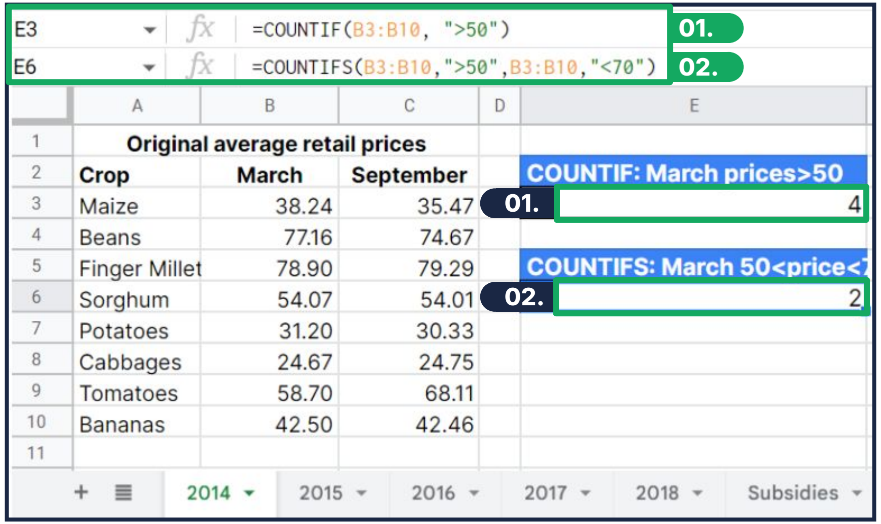
## Example use:

The **range** on **COUNTIF** as well as **`criteria_range1`** and **`criteria_range2`** on **COUNTIFS** will be the column that contains prices for the month of **March**.

Our statements will be **exclusive** of the given **price boundaries** since we want to count values that are **more than** KSh 50 for the first problem and **more than** KSh 50 and **less than** KSh 70 for the second problem.

**01.** Enter the formula **`=COUNTIF(B3:B10, ">50")`** on cell E3 and press **ENTER**.

**02.** Enter the formula **`=COUNTIFS(B3:B10,">50",B3:B10,"<70")`** on cell E6 and press **ENTER**.



```
E3  fx  =COUNTIF(B3:B10, ">50")              01.
E6  fx  =COUNTIFS(B3:B10,">50",B3:B10,"<70") 02.
```

|  | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Original average retail prices |  |  |  |  |
| 2 | Crop | March | September |  | COUNTIF: March prices>50 |
| 3 | Maize | 38.24 | 35.47 | 01. | 4 |
| 4 | Beans | 77.16 | 74.67 |  |  |
| 5 | Finger Millet | 78.90 | 79.29 |  | COUNTIFS: March 50<price<7 |
| 6 | Sorghum | 54.07 | 54.01 | 02. | 2 |
| 7 | Potatoes | 31.20 | 30.33 |  |  |
| 8 | Cabbages | 24.67 | 24.75 |  |  |
| 9 | Tomatoes | 58.70 | 68.11 |  |  |
| 10 | Bananas | 42.50 | 42.46 |  |  |
| 11 |  |  |  |  |  |

## The SWITCH function

> **SWITCH** evaluates a **logical expression** and **matches** it with **one of the cases** available. It then returns the **value** defined in that **case block**. If there is no matching case, it returns the **default** value.

```
=SWITCH(expression, case1, value1, [case2_or_default, …], [value2, …])
```

- **`expression`** – The value(s) to be evaluated.
- **`case1`** – The first case to be checked against **`expression`**.
- **`value1`** – The corresponding value to be returned when **`case1`** matches **`expression`**.
- **`case2_or_default…`** – [OPTIONAL] Additional cases to try if the previous ones don't match **`expression`**, or an optional default value to be returned if none of the cases match **`expression`**.
- **`value2…`** – [OPTIONAL] Additional values to be returned if their corresponding cases match **`expression`**.

# The SWITCH function

**Example use:**

Let's say we want to **categorise** the food crops. We will list cases with the least number of items first and let the default have the most number of items.

Based on our list of food crops, our categories are **Legume**, **Vegetable**, **Fruit**, and **Grain**.

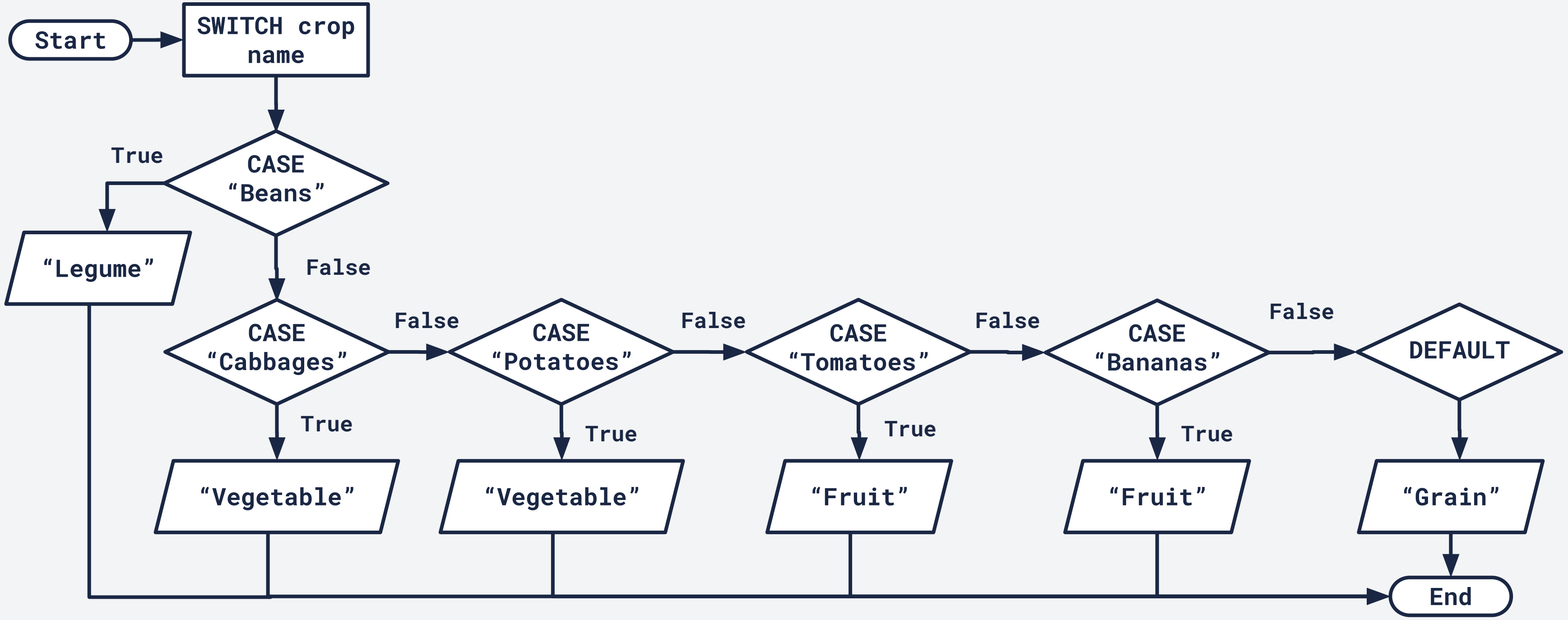
1. **Legume** has *one* crop; **beans.**
2. **Vegetable** has *two* crop; **cabbages** and **potatoes**.
3. **Fruit** has *two* crops; **tomatoes** and **bananas**.
4. **Grain** has *three* crops; **maize**, **sorghum**, and **finger millet**.

Since **Grain** has the most number of crops, we will make it the **default** case.

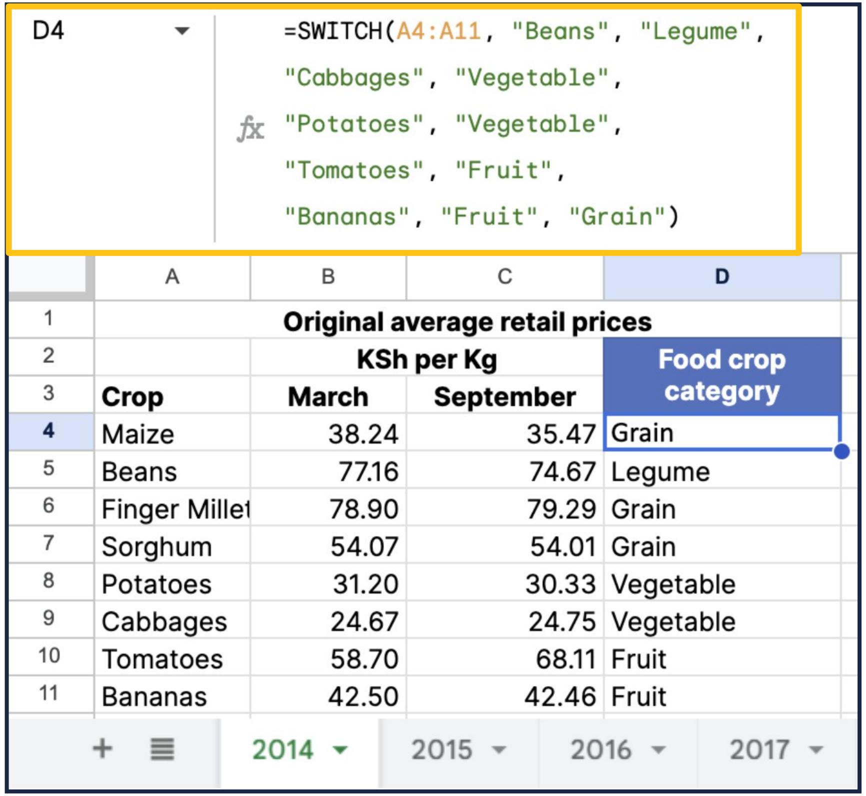
```
SWITCH crop name
- CASE "Beans"
- - Output = "Legume"
- CASE "Cabbages"
- - Output = "Vegetable"
- CASE "Potatoes"
- - Output = "Vegetable"
- CASE "Tomatoes"
- - Output = "Fruit"
- CASE "Bananas"
- - Output = "Fruit"
- DEFAULT
- - Output = "Grain"
```

# The SWITCH function

**Example use:**



# The SWITCH function



D4

```
=SWITCH(A4:A11, "Beans", "Legume",
"Cabbages", "Vegetable",
"Potatoes", "Vegetable",
"Tomatoes", "Fruit",
"Bananas", "Fruit", "Grain")
```

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Original average retail prices | | | |
| 2 | | KSh per Kg | | Food crop category |
| 3 | Crop | March | September | |
| 4 | Maize | 38.24 | 35.47 | Grain |
| 5 | Beans | 77.16 | 74.67 | Legume |
| 6 | Finger Millet | 78.90 | 79.29 | Grain |
| 7 | Sorghum | 54.07 | 54.01 | Grain |
| 8 | Potatoes | 31.20 | 30.33 | Vegetable |
| 9 | Cabbages | 24.67 | 24.75 | Vegetable |
| 10 | Tomatoes | 58.70 | 68.11 | Fruit |
| 11 | Bananas | 42.50 | 42.46 | Fruit |

## Example use:

The **order** in which we list our cases will determine **how long** our **SWITCH** statement will be.

**01.** Enter the formula **`=SWITCH(A4:A11, "Beans", "Legume", "Cabbages", "Vegetable", "Potatoes", "Vegetable","Tomatoes", "Fruit", "Bananas", "Fruit", "Grain")`** in cell D4.

**02.** Press **Enter**.

Try rewriting the statement below with **"Tubers"** as your default to see the length difference.

Spreadsheet functions

# SWITCH versus IFS

Both **SWITCH** and **IFS** are logical functions that allow you to perform different actions based on multiple conditions. However, there are some scenarios where you may prefer to use **SWITCH** over **IFS**, or vice versa.

**SWITCH** is preferred when:

1. You have a **single value or cell** that you want to match with **multiple possible outcomes**.
2. You want to **avoid evaluating unnecessary conditions**. With **IFS**, all conditions are evaluated, even if one of the earlier conditions is true. With **SWITCH**, only the condition that matches the specified value is evaluated, which saves processing time.

**IFS** is often more flexible than **SWITCH** because it can handle a broader range of conditions and outcomes.

**IFS** is more flexible than **SWITCH** because:

1. It allows you to evaluate **multiple conditions**, whereas **SWITCH** only allows you to **match a single value with multiple outcomes**. This means that if you have multiple conditions that you need to evaluate, you can use **IFS** to handle all of them.
2. It allows you to **specify different outcomes** based on the conditions that are met. This means that you can use formulas to **calculate outcomes** based on the data in your sheet.

## Other logic functions

The **AND** function returns **true** if **all** of the provided arguments **are logically true**, and **false** if any of the provided arguments **are logically false** while the **NOT** function returns the **opposite** of a logical value.

```
=AND(logical_expression1,(logical_expression2, ...])
=NOT(logical_expression)
```

- **`logical_expression/logical_expression1`** – An expression that represents some logical value, i.e. TRUE or FALSE, or an expression that can be coerced to a logical value.
- **`logical_expression2…`** – [OPTIONAL] More expressions that represent logical values.

**Examples:**

- `=AND(TRUE,FALSE,TRUE)` returns **FALSE**
- `=AND(FALSE,FALSE,FALSE)` returns **FALSE**
- `=AND(TRUE,TRUE,TRUE)`returns **TRUE**
- `=NOT(TRUE)` returns **FALSE**
- `=NOT(FALSE)` returns **TRUE**

**Note:** The number 0 is logically False; all other numbers (including negative numbers) are logically True.

# Other logic functions

> The **OR** function returns **true** if **any** argument **is logically true**, and **false** if **all** arguments **are logically false**, while **XOR** returns **true** if an **odd number of arguments are logically true**, and **false otherwise**.

```
=OR(logical_expression1, [logical_expression2, …])
=XOR(logical_expression1, [logical_expression2, …])
```

- `logical_expression1` – An expression that represents some logical value, i.e. TRUE or FALSE, or an expression that can be coerced to a logical value.
- `logical_expression2…` – [OPTIONAL] More expressions that represent logical values.

**Examples:**

- `=OR(TRUE,FALSE,TRUE)` returns **TRUE**
- `=OR(FALSE,FALSE,FALSE)` returns **FALSE**
- `=OR(TRUE,TRUE,TRUE)`returns **TRUE**
- `=XOR(TRUE, FALSE, TRUE)` returns **FALSE**
- `=XOR(FALSE, FALSE, TRUE)` returns **TRUE**

# Other logic functions

## Example use:

For 2014, tag all food crops as either **rising price trend** if the price in **September is higher than that in March**, or **dropping price trend** if the price in **September is lower than the price in March**.

For this problem, we need to identify a trend for each food crop from March to September based on whether a crop's price went up or down.

We also have to ensure that the trend is true for both the **original retail prices** and **subsidised retail prices**.

Since we have not been told what to do with crops that don't fall in the category of dropping or rising price trend, we will create a third label called **unidentified trend** to cater for this.

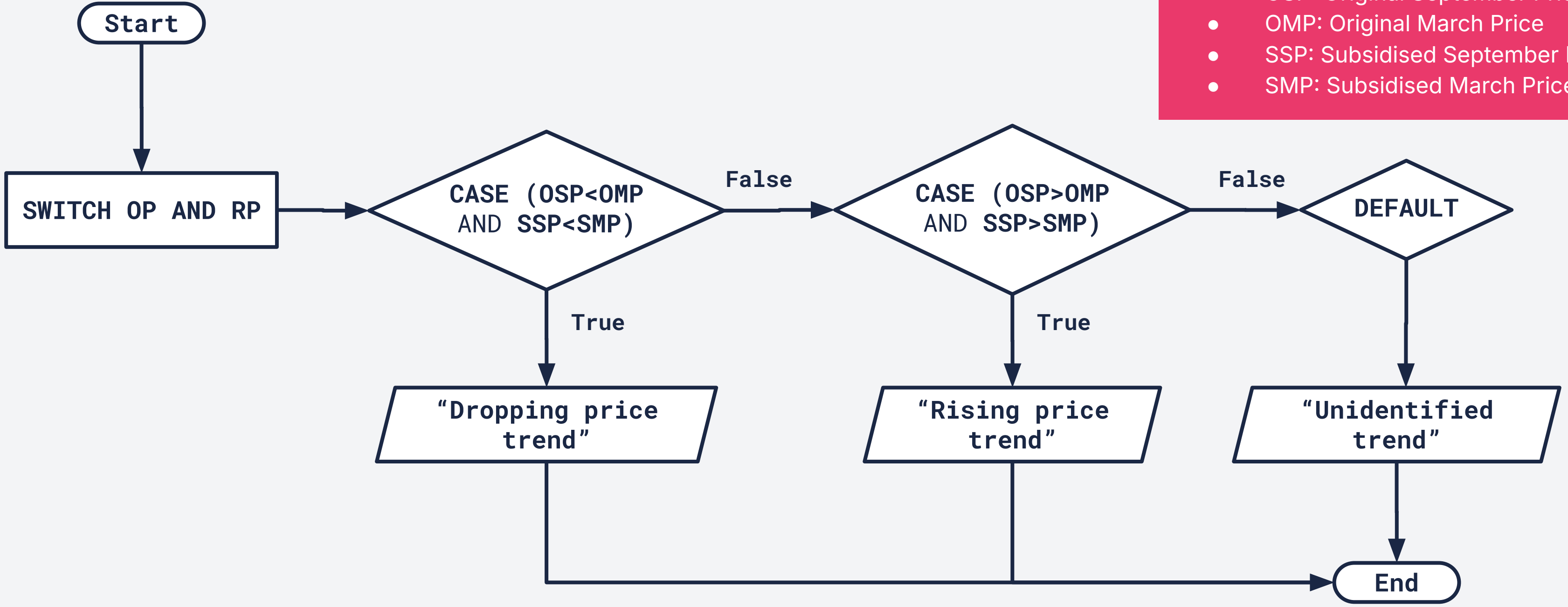
Let **original retail price** in **Sept** be **OSP** and in **March OMP**. Let **subsidised retail price** in **Sept** be **SSP** and in **March SMP**. Also, let all original prices be **OP** and all retail prices be **RP**.

```
SWITCH OP AND RP
- CASE(OSP < OMP AND SSP < SMP)
- - Output =“Dropping price trend”
- CASE(OSP > OMP AND SSP > SMP)
- - Output = “Rising price trend”
- DEFAULT
- - Output = “Unidentified trend”
```

# Other logic functions

## Example use:

- OP: Original Prices
- SP: Subsidised Prices
- OSP: Original September Price
- OMP: Original March Price
- SSP: Subsidised September Price
- SMP: Subsidised March Price



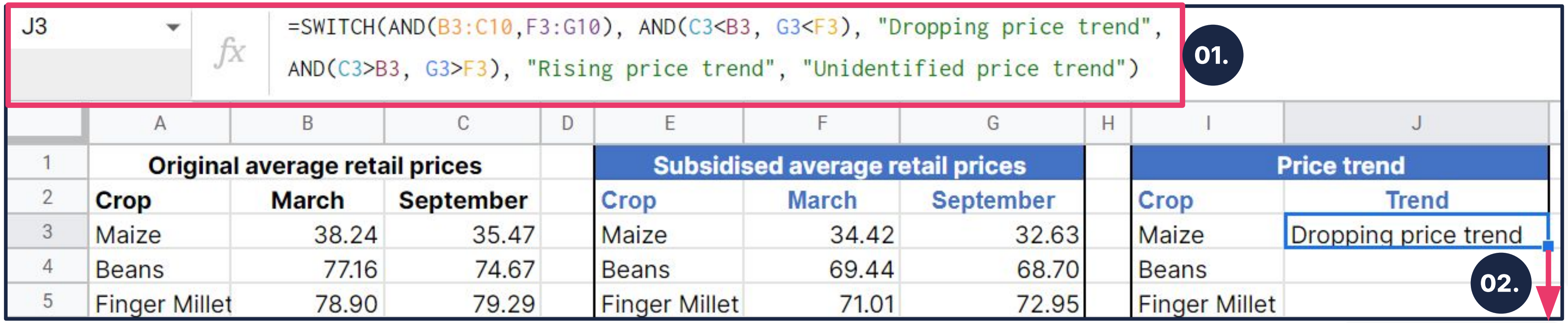
# Other logic functions

## Example use:

Our **SWITCH** function will evaluate prices in both the original and subsidised retail prices, so the **`expression`** will be an **AND** function. Our first two cases will check for conditions using the original and subsidised tables and will also be **AND** functions. Our default case will be crops that **didn't fit** into either of the first two cases.

**01.** Enter the formula **`=SWITCH(AND(B3:C10,F3:G10), AND(C3<B3, G3<F3), "Dropping price trend", AND(C3>B3, G3>F3), "Rising price trend", "Unidentified price trend")`** on cell J3.

**02.** Replicate the formula by dragging the fill handle down.



J3

```
=SWITCH(AND(B3:C10,F3:G10), AND(C3<B3, G3<F3), "Dropping price trend",
AND(C3>B3, G3>F3), "Rising price trend", "Unidentified price trend")
```

|  | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Original average retail prices | | | | Subsidised average retail prices | | | | Price trend | |
| 2 | Crop | March | September | | Crop | March | September | | Crop | Trend |
| 3 | Maize | 38.24 | 35.47 | | Maize | 34.42 | 32.63 | | Maize | Dropping price trend |
| 4 | Beans | 77.16 | 74.67 | | Beans | 69.44 | 68.70 | | Beans | |
| 5 | Finger Millet | 78.90 | 79.29 | | Finger Millet | 71.01 | 72.95 | | Finger Millet | |