

## You

How can I rewrite this python function as a named formula in power apps: `def checkDays(): daysCount = 0 if day2Visible = True: daysCount += 1 if day3Visible = True: daysCount += 1 if day4Visible = True: daysCount += 1 if day5Visible = True: daysCount += 1 if day6Visible = True: daysCount += 1 if day7Visible = True: daysCount += 1 if day8Visible = True: daysCount += 1 if day9Visible = True: daysCount += 1 if day10Visible = True: daysCount += 1 return daysCount`

## Copilot

Certainly! In Power Apps, you can achieve similar functionality using **named formulas**. These are reusable functions that can be defined once and referenced throughout your app. Let's rewrite your Python function as a named formula in Power Apps:

1. First, enable the **Named formulas** experimental feature in Power Apps (authoring version 3.22091 or later).
2. Reload your app after enabling the feature.
3. Add your formulas to the `App.Formulas` property.

Here's how you can rewrite your function:

```
// Define a named formula for counting visible days
fxCountVisibleDays =
    CountIf([day2Visible, day3Visible, day4Visible, day5Visible, day6Visible, day7Visible, day8Visible, day9Visible, day10Visible], true)

// Usage example:
// You can reference fxCountVisibleDays anywhere in your app
// For instance, in the Items property of a gallery:
Items = Filter(MyDataSource, fxCountVisibleDays > 0)
```

In this example:

- `fxCountVisibleDays` calculates the count of visible days.
- We use the `CountIf` function to count the number of true values among the visibility properties (e.g., `day2Visible`, `day3Visible`, etc.).

Remember that named formulas are read-only and always have up-to-date return values. They can significantly improve app loading times compared to using global variables defined in the `App.OnStart` event. Give it a try, and feel free to adapt this approach to your specific use case!



## You

What is a quick way to set multiple variables to false in power apps?

## Copilot

Certainly! In Power Apps, you can set multiple variables to `false` using the semicolon (;) to separate statements. Here's an example:

```
If(txt_Value.Text = "",
    Set(isEmpty, true);;
    Set(isError, true);;
    Set(another, 123);
    Set(isEmpty, false);;
    SubmitForm(EditForm1)
)
```

In this example:

- If `txt_Value` is empty, it sets `isEmpty` and `isError` to `true`, assigns a value to `another`, and finally sets `isEmpty` back to `false`. Adjust the variable names and conditions as needed for your specific scenario! 😊🚀

## You

No, I mean is there a way of shortneing this statement in power apps: `Set(day2Visible, false); Set(day3Visible, false); Set(day4Visible, false); Set(day5Visible, false); Set(day6Visible, false); Set(day7Visible, false); Set(day8Visible, false); Set(day9Visible, false); Set(day10Visible, false);`

## Copilot

Certainly! To shorten the repetitive assignment of `false` to multiple variables in Power Apps, you can use a **collection** or an **array** to store the variables and then update them in a loop. Here's an example using a collection:

1. Create a collection (e.g., `colVisibleDays`) with the names of your variables (e.g., `day2Visible`, `day3Visible`, etc.).
2. Use a `ForAll` function to iterate through the collection and set each variable to `false`.

Here's how you can do it:

```
// Create a collection with variable names
ClearCollect(colVisibleDays,
    { Name: "day2Visible" },
    { Name: "day3Visible" },
    { Name: "day4Visible" },
    // ... add other variables here
    { Name: "day10Visible" }
);
```

```
// Set all variables to false
ForAll(colVisibleDays,
```

```

        Set(VarName, Name)); // Get the variable name
        Set(ThisItem[VarName], false) // Set the variable to false
    )

```

In this example:

- colVisibleDays contains the names of your variables.
- The ForAll loop iterates through each item in the collection.
- VarName holds the current variable name.
- ThisItem[VarName] refers to the actual variable, and we set it to false.

Feel free to adjust the collection and variable names according to your specific scenario! 😊🚀

## You

How do I do this if the variables can change one at a time dynamically while the power app is running?

## Copilot

If the variables can change dynamically during runtime in your Power App, you can handle this by using a **collection** to track the variable names and their values. Here's how you can achieve this:

1. Create a collection (e.g., colVisibleDays) to store the variable names and their corresponding values.
2. Whenever a variable changes, update the collection accordingly.
3. Use the collection to manage the visibility of your days.

Here's an example:

```

// Initialize the collection (you can do this in App.OnStart)
ClearCollect(colVisibleDays,
    { Name: "day2Visible", Value: false },
    { Name: "day3Visible", Value: false },
    // ... add other variables here
    { Name: "day10Visible", Value: false }
);

// When a variable changes (e.g., day2Visible becomes true)
UpdateContext({ varChangedDay: "day2Visible" }); // Set the changed variable name

// Update the collection value based on the changed variable
ForAll(colVisibleDays,
    If(Name = varChangedDay, UpdateIf(colVisibleDays, Name = varChangedDay, { Name: Name, Value: true })))
);

// Usage example:
// You can reference colVisibleDays anywhere in your app
// For instance, in the Visible property of a control:
Visible = LookUp(colVisibleDays, Name = "day2Visible").Value

```

In this example:

- colVisibleDays contains the variable names and their values.
- When a variable changes (e.g., day2Visible becomes true), set varChangedDay to the changed variable name.
- The ForAll loop updates the collection value for the changed variable.
- You can use LookUp(colVisibleDays, Name = "day2Visible").Value to get the current value of day2Visible.

Remember to adapt this approach to your specific use case, and feel free to adjust the collection structure as needed! 😊🚀