You

Can you convert this python code to work in power apps:

```
hoursPerDay = hours
exactTotalHours = totalHours + (totalMinutes // 60)
keyword = "overtime"
hours = exactTotalHours - (hoursPerDay * self.dayNum)
absoluteHours = abs(hours)
justHours = absoluteHours // 1
justMinutes = round((absoluteHours - justHours) * 60, 0)
if hours < 0:
  keyword = "undertime"
```

Copilot

Sure! Converting Python code to Power Apps involves translating the logic into Power Apps formulas. Here's how you can achieve the same functionality in Power Apps:

1. **Define Variables**: Use Set to define variables.
2. **Calculate Total Hours and Minutes**: Use Power Apps functions for calculations.
3. **Conditional Logic**: Use If for conditional statements.

Here's the equivalent code in Power Apps:

```
// Define variables
Set(hoursPerDay, hours);
Set(exactTotalHours, totalHours + Floor(totalMinutes / 60));
Set(keyword, "overtime");

// Calculate hours and minutes
Set(hours, exactTotalHours - (hoursPerDay * self.dayNum));
Set(absoluteHours, Abs(hours));
Set(justHours, Floor(absoluteHours));
Set(justMinutes, Round((absoluteHours - justHours) * 60, 0));

// Conditional logic
If(hours < 0, Set(keyword, "undertime"));
```

Explanation:

- **Set**: Used to define and update variables.
- **Floor**: Equivalent to // in Python for integer division.
- **Abs**: Calculates the absolute value.
- **Round**: Rounds the number to the nearest integer.
- **If**: Implements conditional logic.

This should help you replicate the Python logic in Power Apps. If you have any more questions or need further assistance, feel free to ask!

Learn more: 1. github.com 2. www.bluedialog.com 3. pypi.org

You

How can I add logic to an upload document button that will allow a user to upload a plaintext (.txt) file to my app?

Copilot

To add logic for uploading a plaintext (.txt) file in Power Apps, you can use the **Attachment Control** along with Power Automate to handle the file upload and processing. Here's a step-by-step guide:

Step 1: Add an Attachment Control

1. **Insert an Attachment Control**: Go to the Insert tab, select **Media**, and then choose **Attachment**.
2. **Configure the Attachment Control**: Set the Accept property to "text/plain" to restrict uploads to plaintext files.

Step 2: Create a Power Automate Flow

1. **Create a New Flow**: Go to Power Automate and create a new flow.
2. **Trigger**: Use the **PowerApps** trigger.
3. **Add Actions**:
    - **Create File**: Use the **Create File** action to save the uploaded file to a location like OneDrive or SharePoint.
    - **Get File Content**: If you need to process the file content, use the **Get File Content** action.
    - **Respond to PowerApps**: Use the **Respond to PowerApps** action to send back any necessary data.

Step 3: Connect Power Apps to the Flow

1. **Add the Flow to Power Apps**: In your Power Apps app, go to the Action tab, select **Power Automate**, and add the flow you created.
2. **Configure the Button**: Set the OnSelect property of the upload button to call the flow and pass the file content.

Example Code for Button's OnSelect Property

```
// Assuming the flow is named 'UploadTextFile'
UploadTextFile.Run(
   First(AttachmentControl.Attachments).Name,
   First(AttachmentControl.Attachments).Value
);
```

Step 4: Display the Uploaded Content (Optional)

If you want to display the content of the uploaded file, you can decode the base64 content returned by the flow and show it in a label.

Example Code for Decoding Base64 Content

```
Set(fileContent, JSON(First(AttachmentControl.Attachments).Value, IncludeBinaryData));
Set(decodedContent, TextInput1.Text);
```

This setup will allow users to upload a plaintext file, and you can handle the file content as needed in your app. If you need further customization or have any questions, feel free to ask!

Learn more: 1. www.powerapps911.com 2. powerusers.microsoft.com 3. www.powerapps911.com 4. www.microsoft.com 5. gernaeysoftware.com 6. www.linkedin.com

You

how can I upload the file if there is no Attachment control?

Copilot

If you don't have an Attachment control available in Power Apps, you can still upload a file by leveraging Power Automate. Here's how you can do it:

Step 1: Create a Power Automate Flow

1. **Create a New Flow**: Go to Power Automate and create a new flow.
2. **Trigger**: Use the **PowerApps** trigger.
3. **Add Actions**:
    - **Create File**: Use the **Create File** action to save the uploaded file to a location like OneDrive or SharePoint.
    - **Get File Content**: If you need to process the file content, use the **Get File Content** action.
    - **Respond to PowerApps**: Use the **Respond to PowerApps** action to send back any necessary data.

Step 2: Connect Power Apps to the Flow

1. **Add the Flow to Power Apps**: In your Power Apps app, go to the Action tab, select **Power Automate**, and add the flow you created.
2. **Configure the Button**: Set the OnSelect property of the upload button to call the flow and pass the file content.

Example Code for Button's OnSelect Property

```
// Assuming the flow is named 'UploadTextFile'
UploadTextFile.Run(
   TextInputFileName.Text,
   TextInputFileContent.Text
);
```

Step 3: Use Text Inputs for File Name and Content

1. **Add Text Inputs**: Add two text input controls to your app. One for the file name and one for the file content.
2. **Set Properties**: Ensure the user can input the file name and paste the file content into these text inputs.

Example Code for Decoding Base64 Content (Optional)

If you need to decode base64 content, you can use the following approach:

```
Set(fileContent, JSON(TextInputFileContent.Text, IncludeBinaryData));
Set(decodedContent, TextInput1.Text);
```

Explanation:

- **TextInputFileName**: Text input for the file name.
- **TextInputFileContent**: Text input for the file content.
- **UploadTextFile.Run**: Calls the Power Automate flow with the file name and content.

This setup allows users to upload a plaintext file by entering the file name and content directly into text inputs. The Power Automate flow will handle the file creation and any additional processing required^1^^2^^3^.

If you need further customization or have any questions, feel free to ask!

Learn more: 1. www.powerapps911.com 2. www.spguides.com 3. www.matthewdevaney.com
4. techcommunity.microsoft.com

You

How can I press a button in power apps and then it lets me browse file explorer for a file?

Copilot

To allow users to browse their file explorer and select a file in Power Apps, you can use the **Power Apps component framework (PCF)**. Specifically, the PickFile function can be used to open a dialog box for file selection. Here's how you can set it up:

Step-by-Step Guide

1. **Create a Custom Component**:
    - Use the Power Apps Component Framework (PCF) to create a custom component that can handle file selection.
    - You can use the context.device.pickFile method to open the file explorer.
2. **Add the Custom Component to Your App**:
    - Once the custom component is created, add it to your Power Apps canvas app.

Example Code for Custom Component

Here's a basic example of how you might set up the custom component:

```
import { IInputs, IOutputs } from "./generated/ManifestTypes";

export class FilePicker implements ComponentFramework.StandardControl<IInputs, IOutputs> {
  private _container: HTMLDivElement;
  private _notifyOutputChanged: () => void;
  private _fileContent: string;

  constructor() {}

  public init(
    context: ComponentFramework.Context<IInputs>,
    notifyOutputChanged: () => void,
    state: ComponentFramework.Dictionary,
    container: HTMLDivElement
  ): void {
    this._container = container;
    this._notifyOutputChanged = notifyOutputChanged;

    const button = document.createElement("button");
    button.innerText = "Browse File";
    button.onclick = this.openFilePicker.bind(this);
    this._container.appendChild(button);
  }

  private openFilePicker(): void {
    const options: ComponentFramework.DeviceApi.PickFileOptions = {
      accept: ["text/plain"],
      allowMultipleFiles: false
    };

    context.device.pickFile(options).then((files) => {
      if (files && files.length > 0) {
        const file = files[0];
        const reader = new FileReader();
```

```
                reader.onload = (e) => {
                    this._fileContent = e.target.result as string;
                    this._notifyOutputChanged();
                };
                reader.readAsText(file);
            }
        });
    }

    public updateView(context: ComponentFramework.Context<IInputs>): void {}

    public getOutputs(): IOutputs {
        return {
            fileContent: this._fileContent
        };
    }

    public destroy(): void {}
}
```

Explanation:

- **Button Creation**: A button is created to trigger the file picker.
- **File Picker**: The context.device.pickFile method opens the file explorer.
- **File Reading**: The selected file is read as text and stored in a variable.

Adding the Component to Power Apps

1. **Import the Component**: Import the custom component into your Power Apps environment.
2. **Use the Component**: Add the component to your canvas app and bind it to the necessary controls.

Using the File Content

Once the file is selected and read, you can use the file content in your app as needed.

This approach allows you to integrate file browsing functionality directly into your Power Apps application^1^^2^^3^^4^.

If you need further assistance or have any questions, feel free to ask!

Learn more: 1. www.matthewdevaney.com 2. www.youtube.com 3. learn.microsoft.com
4. learn.microsoft.com 5. learn.microsoft.com 6. www.powerapps911.com

You

I'm on power apps, where do I got to create a custom component like the one you mentioned

Copilot

To create a custom component like the one you mentioned in Power Apps, you need to use the Power Apps Component Framework (PCF). Here's a step-by-step guide to get you started:

Step 1: Set Up Your Development Environment

☐ **Install Node.js**: Ensure you have Node.js installed on your machine.

☐ **Install Power Platform CLI**: Use the following command to install the Power Platform CLI:

```
npm install -g pac
```

Step 2: Create a New PCF Project

- **Create a New Project**: Open your command prompt or terminal and run the following commands:

```
pac pcf init --namespace YourNamespace --name FilePicker --template field
cd FilePicker
npm install
```

Step 3: Implement Your Component

1. **Edit the Code**: Replace the content of index.ts with your custom component code. Ensure you have the necessary imports and logic as shown in your example.

2. **Update ControlManifest.Input.xml**: Define the properties and metadata for your component.

Example ControlManifest.Input.xml

```
<control namespace="YourNamespace" constructor="FilePicker" version="1.0.0">
  <resources>
    <code type="typescript" path="index.ts" />
  </resources>
  <property name="fileContent" display-name-key="File Content" type="SingleLine.Text" />
</control>
```

Step 4: Build and Test Your Component

- **Build the Component**: Run the following command to build your component:

```
npm run build
```

Test Locally: Use the following command to test your component locally:

```
npm start
```

Step 5: Deploy Your Component

- **Create a Solution**: Package your component into a solution using the Power Platform CLI:

```
pac solution init --publisher-name YourPublisher --publisher-prefix YourPrefix
pac solution add-reference --path FilePicker
pac solution build
```

2. **Import the Solution**: Import the solution into your Power Apps environment.

Step 6: Use the Component in Power Apps

1. **Add the Component**: Go to your Power Apps canvas app, and add the custom component from the Insert pane.

2. **Configure the Component**: Bind the component to the necessary controls and use it as needed.

This process will help you create and use a custom file picker component in Power Apps. If you have any questions or need further assistance, feel free to ask!

Conversation by Fifa Utuber with Copilot on Monday, Sep 16 2024