PythonスクリプトからMicrosoft OAuth2認証を行いトークンを取得する

「最初は人力でログインし、その後は refresh_token で自動更新し続ける」流れをモジュール化する

処理フロー概略

1. PythonでローカルにHTTPサーバ（Flask）を起動しておく
2. 認証用URLを生成してブラウザで開く
3. ユーザーがMicrosoftの認証ページでログイン + 同意
4. MicrosoftがリダイレクトURIに code を返す → Flaskで受け取る
5. 認可コード(code)を使って access_token / refresh_token を取得
6. access_token で Graph API（Mail.Send） を呼び出してメール送信
7. 次回以降は refresh_token を使って access_token を自動更新（人手不要）

今後の使い回しを想定してモジュール化(フォルダ化)しておく

「msauth」というフォルダを作成。以下、このフォルダ直下に機能実装をすすめる

▼ auth_config.pyファイル作成(認証設定用)

CLIENT_IDやCLIENT_SECRET情報を取得する処理を記載

.env から読み込めるようにしておく。

```
import os
from dotenv import load_dotenv

load_dotenv()

MSAUTH_CLIENT_ID = os.getenv("MSAUTH_CLIENT_ID")
MSAUTH_CLIENT_SECRET = os.getenv("MSAUTH_CLIENT_SECRET")
MSAUTH_REDIRECT_URI = os.getenv("MSAUTH_REDIRECT_URI")
MSAUTH_AUTHORITY = os.getenv("MSAUTH_AUTHORITY", "https://login
MSAUTH_SCOPES = os.getenv("MSAUTH_SCOPES", "").split()
MSAUTH_TOKEN_FILE = os.getenv("MSAUTH_TOKEN_FILE", "ms_token.js
```

▼ token_store.pyファイル作成

MSから受け取ったトークンの保持 & 読み書き処理。

一度取得したトークンをアプリ内で繰り返し使えるようにしておくため

▼ oauth_client.pyファイル作成

下記、3つの処理を行う関数を用意。

- ユーザーに Microsoft のログイン＋同意ページを表示して、認可コード（authorization code）を取得するためのコールバックエンドポイントを起動する
  - Flaskサーバーを 起動し、コールバックエンドポイントを待機させる
  - webbrowser.open() で認証URLをブラウザで開き、Microsoft 側で認証完了後、指定の callback に code を返す
  - コールバックエンドポイントにきたコールバックを受け取る
- 認可コードを access_token / refresh_token に変換する
  - 認可コード（code）を Microsoft の /token エンドポイントに POST
  - 成功したらトークンを受け取る（JSON形式）
    - access_token、refresh_token、expires_in など
  - トークンを保存

- すでに保存されているトークン情報を使って再認証なしで access_token を取得する（ない場合 or 失敗した場合は初回認証を促す）
    - 保存されているトークンを読み出す。
    - refresh_token があればそれで access_token を再取得
    - 成功したらトークンを保存し直し。失敗 or 保存されてなければ 再ログインからフローをやり直す

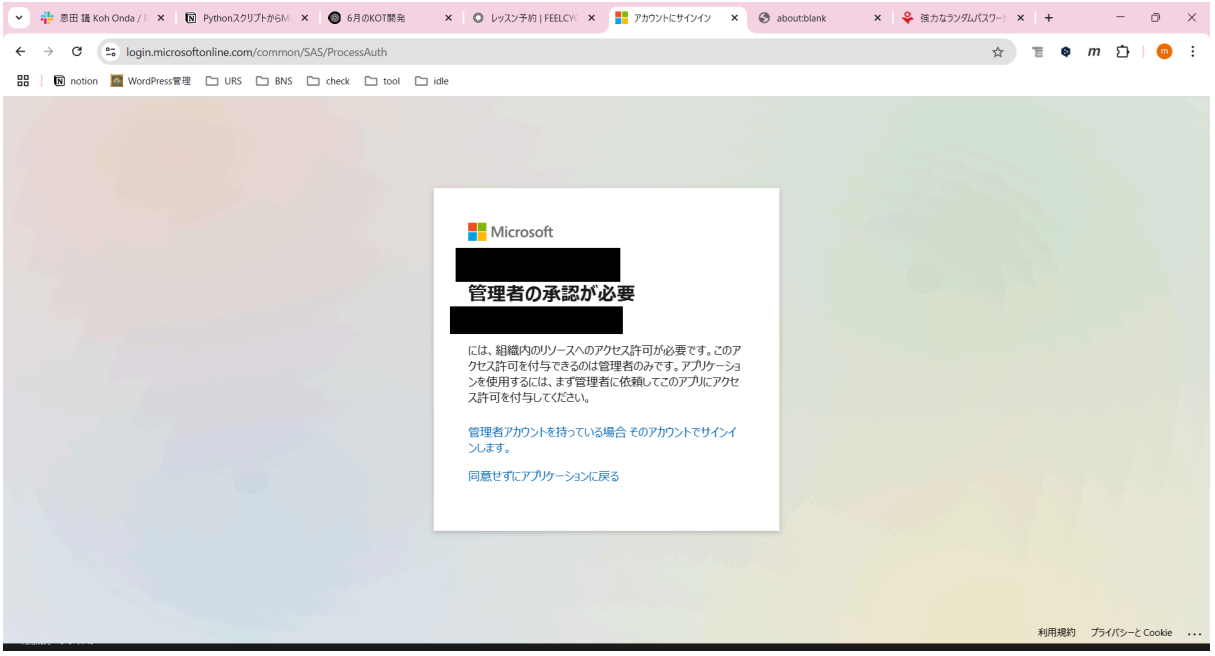
▼ 不足モジュールをインストール

```
pip install requests
pip install flask
```

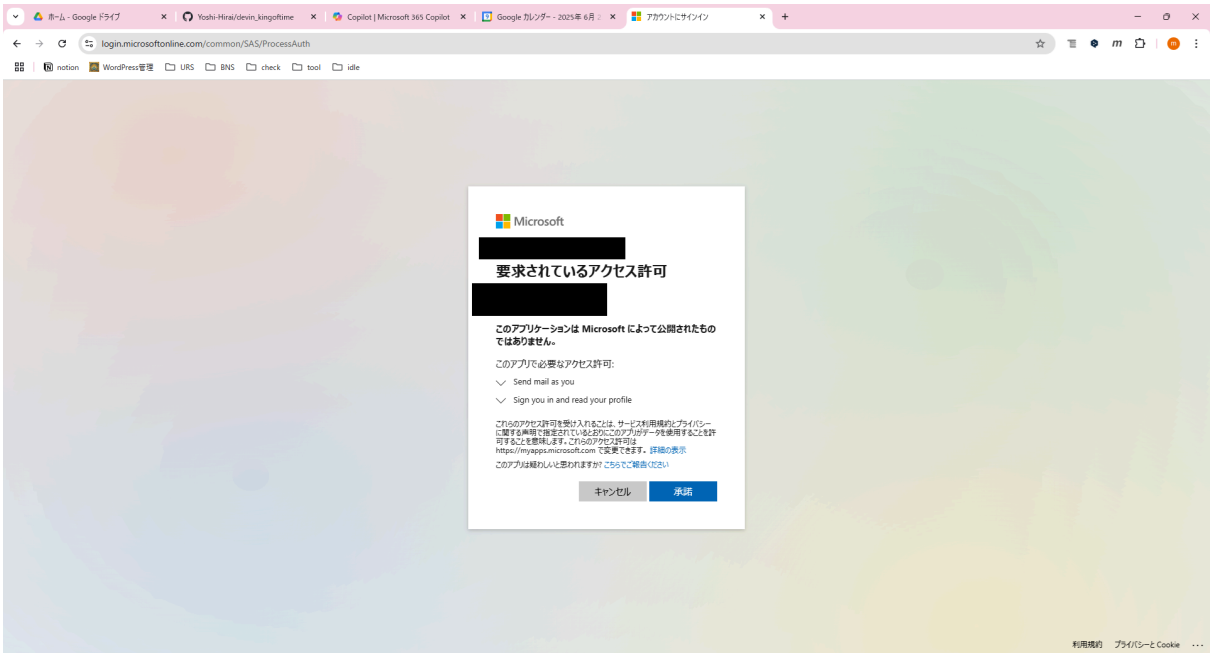
管理者の方に、コールバックエンドポイントを登録してもらう

テストコードを実行&結果

ログインはできたようだが、コールバック(リダイレクトURL)に認証コードを返す部分で処理が通らない様子。



管理者の方にご連絡し、再度設定いただく。



コールバックエンドポイントにcodeが送信されてきたのを確認。

コールバックに送られてきた認可コードを用いてトークンをリクエストする処理が失敗する

HTTPError: 401 Client Error: Unauthorized for url:　となる。

改めて管理者さんから送られてきたOAuth2.0の情報を見直す。

> grant_type=client_credentials ← この指定は「クライアント資格情報フロー」専用
> code がない → ユーザー認証ステップを通過しない
> scope=https://graph.microsoft.com/.default ← これは アプリケーション権限（A

となっており、管理者さんが設定したのは「Client Credentials Flow（ユーザー認証なし）」であるが、私の実装スクリプトは「Authorization Code Flow（ユーザー認証あり）」であるので、エラーが発生する

「Client Credentials Flow（ユーザー認証なし）」「Authorization Code Flow（ユーザー認証あり）」の違い

	Authorization Code Flow	Client Credentials Flow
トークン取得に必要なもの	code, client_id, client_secret	client_id, client_secret, .default
ブラウザログイン	要	不要

使えるAPI	Delegated（ユーザーの代理）	Application（バックグラウンド処理）

「Client Credentials Flow（ユーザー認証なし）」に対応して実行

単純にトークンを取得する関数を作成してテスト実行

- 結果
  - HTTPError: 401 Client Error: Unauthorized
  - Invalid client secret provided. Ensure the secret being sent in the request is the client secret value, not the client secret ID, for a secret added to app
  - (訳)
  - 無効なクライアントシークレットが指定されました。リクエストで送信されるシークレットが、アプリに追加されたシークレットのクライアントシークレットIDではなく、値であることを確認してください。
  - 管理者さんにお問い合わせし、クライアントシークレットのValueを連絡いただいた。設定したところ、きちんとトークンの取得が完了した。