```
"""OB.DAAC direct S3 attempt using /s3credentials (this is the version that fails
for you with 403 HeadObject / AccessDenied ListBucket).

Expected failure (based on your environment):
- s3fs.open(...) -> 403 Forbidden on HeadObject
- list_objects_v2(...) -> AccessDenied with explicit deny on ListBucket

Usage:
  python obdaac_direct_s3_not_working.py --username YOUR_EDL --password
YOUR_EDL_PASSWORD

Notes:
- Region is set to us-west-2.
- This script includes both list attempt and direct open attempt.
"""

import argparse
import base64
import json
import requests
import boto3
import s3fs
from netCDF4 import Dataset

S3CRED_ENDPOINT = "https://obdaac-tea.earthdatacloud.nasa.gov/s3credentials"
BUCKET = "ob-cumulus-prod-public"
KEY = "AQUA_MODIS.20260303T053500.L2.OC.NRT.nc"
REGION = "us-west-2"


def retrieve_credentials(edl_username: str, edl_password: str, s3_endpoint: str =
S3CRED_ENDPOINT) -> dict:
    login_resp = requests.get(s3_endpoint, allow_redirects=False, timeout=60)
    login_resp.raise_for_status()

    if "location" not in login_resp.headers:
        raise RuntimeError("Expected redirect Location header from s3credentials endpoint")

    auth = f"{edl_username}:{edl_password}"
    encoded_auth = base64.b64encode(auth.encode("ascii")).decode("ascii")

    auth_redirect = requests.post(
        login_resp.headers["location"],
        data={"credentials": encoded_auth},
        headers={"Origin": s3_endpoint},
        allow_redirects=False,
        timeout=60,
    )
```

```
    auth_redirect.raise_for_status()

    if "location" not in auth_redirect.headers:
        raise RuntimeError("Expected redirect Location header after posting credentials")

    final = requests.get(auth_redirect.headers["location"], allow_redirects=False, timeout=60)
    final.raise_for_status()

    if "accessToken" not in final.cookies:
        raise RuntimeError("Did not receive accessToken cookie from EDL flow")

    results = requests.get(
        s3_endpoint,
        cookies={"accessToken": final.cookies["accessToken"]},
        timeout=60,
    )
    results.raise_for_status()
    return results.json()


def main():
    p = argparse.ArgumentParser()
    p.add_argument("--username", required=True)
    p.add_argument("--password", required=True)
    p.add_argument("--bucket", default=BUCKET)
    p.add_argument("--key", default=KEY)
    p.add_argument("--region", default=REGION)
    args = p.parse_args()

    creds = retrieve_credentials(args.username, args.password)
    print("Got STS creds; expiration:", creds.get("expiration"))

    # 1) Try list (often denied)
    print("\nTrying list_objects_v2 (may be denied)...")
    s3 = boto3.client(
        "s3",
        region_name=args.region,
        aws_access_key_id=creds["accessKeyId"],
        aws_secret_access_key=creds["secretAccessKey"],
        aws_session_token=creds["sessionToken"],
    )

    try:
        resp = s3.list_objects_v2(Bucket=args.bucket, MaxKeys=5)
        print("List succeeded. Keys:", [o["Key"] for o in resp.get("Contents", [])])
    except Exception as ex:
        print("List failed:", repr(ex))
```

```
    # 2) Try open via s3fs (often fails on HeadObject)
    print("\nTrying s3fs.open (may fail on HeadObject)...")
    fs = s3fs.S3FileSystem(
        key=creds["accessKeyId"],
        secret=creds["secretAccessKey"],
        token=creds["sessionToken"],
        client_kwargs={"region_name": args.region},
    )

    s3_path = f"s3://{args.bucket}/{args.key}"
    print("Opening:", s3_path)

    try:
        with fs.open(s3_path, "rb") as f:
            data = f.read()
        nc = Dataset("inmem.nc", mode="r", memory=data)
        print("Opened netCDF groups:", list(nc.groups.keys()))
        nc.close()
    except Exception as ex:
        print("Open failed:", repr(ex))


if __name__ == "__main__":
    main()
```