Py_to_PDF

May 8, 2025

[ ]:
```python
#!/usr/bin/env python3
"""
ORC API client module for water-line detection pipeline.

Provides:
 - urljoin: helper to build API URLs portably
 - ORC: client class to authenticate and post timeseries and images

Usage:
    orc = ORC(base_url, username, password)
    orc.post_timeseries(site_id, data_dict)
    orc.post_video(data=data_dict, img=image_path)
"""
import os
import logging
from datetime import datetime
import requests

# Module-level logger
logger = logging.getLogger(__name__)


def urljoin(*args) -> str:
    """
    Join URL segments using OS path join and normalize to forward slashes.

    Args:
        *args: URL parts to concatenate.

    Returns:
        Fully joined URL string with '/' separators.
    """
    joined = os.path.join(*args)
    return joined.replace("\\", "/")


class ORC:
```

```
    """
    Client for interacting with the Open River Cam REST API.

    Supports token-based or username/password authentication.
    Provides methods to post time series data and video/images.
    """
    def __init__(self, base_url: str, token: str=None, username: str=None,␣
↪password: str=None):
        """
        Initialize ORC client.

        Args:
            base_url: Root URL of the ORC API, e.g. "https://openrivercam.com/
↪api".
            token: Optional existing Bearer token.
            username: Email/username for login (if token not provided).
            password: Password for login (if token not provided).

        Raises:
            ValueError: if neither token nor (username and password) are␣
↪provided.
        """
        self.base_url = base_url
        self.username = username
        self.password = password

        # Determine authentication token
        if token:
            self.token = token
        elif username and password:
            self.token = self.get_token()
        else:
            raise ValueError("Must supply either token or username+password")

    @property
    def headers(self) -> dict:
        """
        HTTP headers including Authorization if token is set.

        Returns:
            Dictionary of headers for API requests.
        """
        # Include Bearer token header
        return {"Authorization": f"Bearer {self.token}"} if hasattr(self,␣
↪"token") else {}

    def get_token(self) -> str:
```

```
        """
        Obtain a new JWT token via the ORC token endpoint using email/password.

        Returns:
            Access token string.

        Raises:
            ValueError: if the request fails or password is missing.
        """
        if not self.password:
            raise ValueError("Password required to get token.")
        # Construct token URL and payload
        url = urljoin(self.base_url, "token/")
        data = {"email": self.username, "password": self.password}
        r = requests.post(url, headers={}, json=data)

        # Check for successful token response
        if r.status_code in (200, 201):
            return r.json().get("access")
        else:
            logger.error(f"Token request failed: {r.status_code}, {r.text}")
            raise ValueError(f"Token request failed with status code {r.
↪status_code}")

    def post_request(self, url: str, data: dict=None, files: dict=None) ->␣
↪requests.Response:
        """
        Generic POST helper for JSON or multipart requests.

        Args:
            url: Full endpoint URL.
            data: JSON body or form data.
            files: Optional dict of file handles for multipart upload.

        Returns:
            requests.Response object on success.

        Raises:
            ValueError: if status code not 200 or 201.
        """
        if files:
            # Send multipart/form-data when uploading files
            r = requests.post(url, headers=self.headers, data=data, files=files)
        else:
            # Send JSON body
            r = requests.post(url, headers=self.headers, json=data)
```

```
        if r.status_code in (200, 201):
            return r
        else:
            logger.error(f"POST request failed: {r.status_code}, {r.text}")
            raise ValueError(f"POST request failed: {r.status_code}, {r.text}")

    def post_timeseries(self, site_id: int, data: dict) -> requests.Response:
        """
        Upload a time series datapoint to the ORC API.

        Args:
            site_id: Identifier of the camera site.
            data: Dict with keys 'timestamp' (datetime or ISO string) and 'h' (water height).

        Returns:
            Response object from the POST.
        """
        # Convert datetime to ISO8601 string if needed
        if isinstance(data.get("timestamp"), datetime):
            data["timestamp"] = data["timestamp"].strftime("%Y-%m-%dT%H:%M:%SZ")
        # Build endpoint URL
        url = urljoin(self.base_url, "site", str(site_id), "timeseries/")
        return self.post_request(url, data=data)

    def post_video(self, data: dict, file: str=None, img: str=None) -> requests.Response:
        """
        Upload video or image snapshot to the ORC video endpoint.

        Args:
            data: Dict with 'timestamp' (datetime or ISO), 'camera_config', plus optional metadata.
            file: Local filepath to a video file to upload.
            img: Local filepath to an image file to upload.

        Returns:
            Response object from the POST.
        """
        # Build URL
        url = urljoin(self.base_url, "video/")
        # Ensure timestamp is ISO string
        if isinstance(data.get("timestamp"), datetime):
            data["timestamp"] = data["timestamp"].strftime("%Y-%m-%dT%H:%M:%SZ")
        # Enforce required status and camera_config
        data.setdefault("status", 4)
        if "camera_config" not in data:
```

```
            raise ValueError("You must supply 'camera_config' in the data dict.
↪")

        files = {}
        # Attach video file if provided
        if file:
            if not os.path.exists(file):
                raise IOError(f"File {file} does not exist.")
            files["file"] = (os.path.basename(file), open(file, "rb"))
        # Attach image snapshot if provided
        if img:
            if not os.path.exists(img):
                raise IOError(f"Image file {img} does not exist.")
            files["image"] = (os.path.basename(img), open(img, "rb"))

        return self.post_request(url, data=data, files=files)
```