Richtlijnen om HYDROLIB Python scripts conform standaard op te leveren

Versie 0.4

De HYDROLIB verzameling van Python tools voor hydrodynamische workflows staat open voor toevoegen van nieuw ontwikkelde scripts en tools. Dit versterkt de herbruikbaarheid en inzetbaarheid van deze code in o.a. modelstudies. De code van HYDROLIB leeft op GitHub: https://github.com/Deltares/HYDROLIB.

Om code toe te voegen moet gebruik worden gemaakt van het versiebeheersysteem *git*. Nieuwe of aangepaste code ontwikkelt men daarbij in een *branch*; een kopie van de hoofdlijn waarin ontwikkelingen zonder problemen voor andere gebruikers kunnen worden geïmplementeerd. Om de code naar de hoofdlijn te verplaatsen maken we gebruik van zgn. *pull requests*; een aanvraag om code op te nemen in de hoofdlijn. In de hoofdlijn kan na elke geaccepteerde *pull request* een nieuwe versie[1] gepubliceerd worden, die voor iedereen ook met *pip* te installeren is. Voordat een *pull request* geaccepteerd kan worden, moet de nieuw toe te voegen code voldoen aan een aantal eisen in dit document beschreven als richtlijn. Hiermee garanderen we de kwaliteit en bruikbaarheid van HYDROLIB, nu en in de toekomst.

Dit document is een gezamenlijk product van de 13 organisaties die HYDROLIB geïnitieerd hebben. De richtlijn bestaat uit een checklist die doorlopen kan worden door een partij die een tool (set aan scripts) of enkele losse scripts wil opleveren in HYDROLIB. De checklist bestaat uit 3 niveaus: generiek, vereist, aanbeveling:

- Generiek:
    - Akkoord met licentie
    - Intellectueel eigendom
    - Aangemeld als contributor
- Vereist:
    - Generieke scripts
    - Markers voor expliciete keuzes
    - Environment
    - Documentatie:
        - Gebruikersdocumentatie
        - API-docs
    - Review
- Aanbeveling:
    - Documentatie:
        - Tutorial
        - Testmodel
        - Notebook
    - Testen
    - In voorkomende gevallen: projectboard

[1] Release van een nieuwe versie kan wellicht per deel-tool onder HYDROLIB gebeuren en niet noodzakelijk altijd voor het overkoepelende hydrolib pakket. Dit levert flexibiliteit op voor de deel-tools en hun maintainers.

Licentie

Alle HYDROLIB tools en scripts zijn beschikbaar onder LGPL, zie https://www.gnu.org/licenses/lgpl-3.0.html.en. Door een tool of scripts aan te bieden via HYDROLIB gaan zowel ontwikkelaar als opdrachtgever akkoord met die licentie.

Intellectueel eigendom

Het intellectueel eigendom ligt standaard bij de inbrengers van de code. Vaak zijn dit de oorspronkelijke ontwikkelaars (en dan waarschijnlijk: hun werkgever), of de opdrachtgever die de tool heeft laten ontwikkelen. Hierover worden onder het overkoepelende HYDROLIB vooralsnog geen afwijkende afspraken gemaakt.

Contributors en Maintainers

In ieder geval één ontwikkelaar per tool of script dient zich aan te melden als zgn. *contributor* van het overkoepelende HYDROLIB, zodat toekomstige vragen (over bv. aanpassingen) bij de juiste persoon terecht komen. Dit gebeurt automatisch wanneer een ontwikkelaar een *pull* request voor nieuwe codebijdrages doet. Hiermee vervult die persoon de zgn. *maintainer* rol voor die betreffende tool. Indien deze persoon de betreffende organisatie verlaat, draagt die organisatie zorg voor een vervangende *maintainer*. Deze maintainer moet genoemd staan in de README.md van elke tool.
Ook wanneer de contributie resultaat is uit projectwerk, zal dit vaak de maker of andere vertegenwoordiger van de makende partij zijn; alternatief kan dit een vertegenwoordiger van de opdrachtgevende partij zijn, als die zich voldoende toegewijd voelt om eerste aanspreekpunt van de tool te worden.
Het is aan de maintainende organisatie om te kiezen welke mate van support geleverd wordt. Hierover staan in dit handvat bewust geen richtlijnen. Belangrijk is in de eerste plaats dat er een vindbaar en bereikbaar aanspreekpunt is.

Generieke scripts

Om de scripts voor anderen bruikbaar te maken dienen ze generiek te zijn. Dat wil zeggen dat het script niet alleen in één omgeving voor één scenario werkt, maar dat het bij iedereen kan werken en bij kan dragen in meerdere scenario's.

- De code bestaat uit één of meerdere functies, de code is modulair, de functies zijn herbruikbaar in andere tools en elke functie heeft één taak.
- Bestandspaden zijn niet hard gecodeerd, men maakt gebruik van Pathlib Python package
- Functies zijn gedocumenteerd met docstrings – input, output, beschrijving van argumenten adhv. zgn Google style[2] en beschrijven ook het waarom van de functie.
- Vanaf de eerste publicatie op GitHub worden verzoeken (bugreport, functionaliteitsuitbreiding, gebruikersvraag) in aparte issues op de HYDROLIB GitHub pagina ingediend, zodat de afstemming en implementatie in de community daar centraal gevoerd kan worden.

Markers voor expliciete keuzes

Indien een Python script expliciete keuzes bevat die de resultaten beïnvloeden, dan dienen deze keuzes in het script gemarkeerd te worden. Daarmee verduidelijken we voor andere gebruikers welke eerder gemaakte keuzes de model resultaten mogelijk beïnvloeden. Hierna volgt een voorbeeld van een marker[3]:

[2] https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html
[3] Het formaat voor keuze-markers is gebaseerd op https://peps.python.org/pep-0350/. Het categorie-veld is optioneel.

```
class _Volume(object):
    # NOTE: Default evaporation and rainfall intensities.
    # Use update({'volume',..}) to customize. <c:choice>
    evaporation = 0
    rainfall = 15
```

Environment

Als het nieuwe script een Python pakket gebruikt dat nog niet opgenomen is in de bestaande environment.yaml, dan moet die dependency worden toegevoegd aan de hydrolib environment.yaml.
Om te voorkomen dat we een zgn. “zwaar” pakket maken is het fijn als het aantal extra pakketten wordt beperkt. Met bestaande pakketten als xarray en geopandas (en al hun dependencies die automatisch meekomen zoals numpy) zou men uit de voeten kunnen.
Aanbeveling: dit kan getest worden door lokaal een schone installatie uit te voeren.

Documentatie

Elk script dient gedocumenteerd te worden met inachtneming van de volgende principes:

- Gebruikersdocumentatie: Uitleg van wat het script doet/ waar je het voor kan gebruiken/ wat de aannames en limitaties zijn. De lengte kan variëren van enkele regels (bijvoorbeeld verwijzend naar welk script wat doet) tot meerdere pagina’s (voor een uitgebreide tool met veel gebruikersopties).
  Deze gebruikersdocumentatie kan onder HYDROLIB worden ondergebracht of op een externe website. Indien een organisatie kiest om de documentatie buiten HYDROLIB te hosten, dan dient wel een korte beschrijving van het script / de tool in HYDROLIB worden aangeboden en met een weblink verwezen naar de documentatie.
- API-documentatie: De onderliggende modules/ functies/ classes zijn met eerdergenoemde docstrings gedocumenteerd, ten bate van automatisch gegenereerde API-documentatie. Dit geldt in elk geval voor de “publieke” functies die gebruikers gaan aanroepen. Aanvullend (maar optioneel) strekt het tot aanbeveling om ook “interne” functies van docstrings te voorzien, om mede-ontwikkelaars te helpen.
- De documentatie dient gemaakt te worden met MkDocs, zie https://www.mkdocs.org/. De bestaande documentatie (zoals die in de *docs* folder bestaan) kan worden uitgebreid.
- Schrijftaal: API-documentatie is altijd in het Engels, gebruikersdocumentatie bij voorkeur ook.
- Aanbevelingen:
  - Elk script wordt opgeleverd met een tutorial die de essentie van de tool demonstreert aan de hand van een eenvoudig voorbeeld.
  - Indien mogelijk een kleine dataset / testmodel met resultaten van het model. Dat een nieuwe gebruiker even kan runnen om te vergelijken of de resultaten overeenkomen.
    (Er wordt nog gewerkt aan een centrale demo-dataset die hergebruikt kan worden: https://github.com/Deltares/HYDROLIB-data.)
  - Indien mogelijk een voorbeeld notebook dat kan draaien in Binder gebruikmakend van de dataset / test model in Binder[4].

[4] Meer informatie over Binder-gebruik op: https://github.com/Deltares/HYDROLIB-core/issues/250#issuecomment-1168433877

Testen

- Elk toegevoegd script zou tenminste één test moeten hebben, en liefst meer, zodat er altijd gecontroleerd kan worden of de functie nog werkt naar behoren, ook bij eventuele toekomstige wijzigingen in code.
- Een geldige test kan ook een notebook zijn dat een script succesvol uitvoert. Deze notebooks kunnen ook ingesteld worden om automatisch in de testsuites mee te draaien op GitHub.
- Zoals hiervoor ook al gemeld onder documentatie: er wordt gewerkt aan een centrale demo-dataset die hergebruikt kan worden, ook voor testen: https://github.com/Deltares/HYDROLIB-data.
- We maken gebruik van pytest om te testen, in de *tests* folder[5] zijn hier voorbeelden van te vinden. Elke tool mag in zijn eigen folder een *tests* subfolder aanmaken, waarna deze tests automatisch gevonden worden.
- Een *pull request* wordt automatisch getest[6] door GitHub, dus ook in elke aparte branch en niet alleen in *main*. Mits de nieuwe code in een test wordt aangeroepen, komen hierdoor snel eventuele problemen in beeld.

Review

Als bovenstaande zaken in orde zijn moet als laatste iemand anders de *pull request* doornemen en deze afwijzen, vragen om wijzigingen of deze goedkeuren. In het laatste geval kan de code gemerged worden, waarna automatisch een nieuwe release van HYDROLIB wordt gemaakt.
Voornaamste aandachtspunten tijdens zo'n review zijn de vereiste onderdelen zoals samengevat in dit handvat en de code in het algemeen.
De reviewer kan een externe contributor zijn, een opdrachtgever, of een collega. Achterliggende gedachte is niet om extra werk te creëren, maar dat een tweede paar ogen vaak gemiste fouten ontdekt, en ook dat tijdens deze review al eerste kennisdeling plaatsvindt.

Projectboard

Als een bepaalde tool meer is dan enkele scripts, en het de bedoeling is dat hier actief aan doorontwikkeld wordt door meerdere personen, kan optioneel een projectboard aangemaakt worden op de publieke GitHub pagina's. Voor losse en gereed opgeleverde scripts is dit niet nodig. Dit is pas nuttig als voor een bepaalde tool veel (10+) issues open staan en er meerdere issues tegelijkertijd door meerdere ontwikkelaars uitgevoerd, gereviewd en getest worden. Dit projectboard dient dan als een issuetracker waarmee zowel de ontwikkelaars als externen op de hoogte zijn van de voortgang van huidige en voorgestelde toekomstige ontwikkelingen.

[5] Veel voorbeeld testfuncties zijn ook te zien in de HYDROLIB-core repository op: https://github.com/Deltares/HYDROLIB-core/tree/main/tests
[6] Niet alleen elke pull request, maar zelfs elke code commit naar de repository wordt automatisch getest.