Etape 3 : Lier les librairies

ajoute l'interpréter dans le nom du .so résultant

python_add_library (conception_orientee_objet MODULE WITH_SOABI
conception_orientee_objet.cpp)

target_link_libraries (conception_orientee_objet PUBLIC
exemple_adder
Boost::python)

Etape 4 : Ecriture des fonctions dans le fichier .cpp

#include <boost/python.hpp>

#include "conception_orientee_objet/exemple_adder.hpp"

BOOST_PYTHON_MODULE (conception_orientee_objet) { boost::python::def
("add", coo::add); }

url GitHub github.com/nim65s/conception-orientee-objet

07/09

Outils

Style défini en Python par https://peps.python.org/pep-0008/

Sur UNIX flake8 hello.py donne les erreurs de style dans le terminal

Pour le reformater black hello.py

Pour le C++ clang-format -i nom.cpp conforme à clang.llvm.org/docs

Statistique : ① Projet Mypy : vérifie l'utilisation des types appelé avec mypy nom.py pour afficher les erreurs

② Pyupgrade : passage d'un code source à une nouvelle version de Python (clang-tidy pour le C++)

Gestion des outils : Solution 1 : utiliser un éditeur de texte ou un IDE embarquant ces outils.

Solution 2 : pre-comit

Créer un fichier dans le répertoire projet contenant

repos:
- repo: https://github

cf diapo 18

Exécution avec : pre-commit run -a
↳ sur tous les fichiers

pre-commit install l'exécute automatiquement avant le commit

<u>Solution 3</u> : pre-commit CI est implémenté directement sur la forge si outils non installés en local

Introduction à Git 2

A chaque création d'un dossier test, src, etc...

On ajoute un répertoire avec add-subdirectory (test) au CMakeLists.txt à la racine :

Puis dans le CMakeList.txt du dossier, on réalise les target link

⚠ au nom du fichier dans le commit ⇒ tout le chemin absolu est noté

Dans un interpréteur Python, on importe un module avec import conception-orientée

On utilise une fonction de ce module avec le . par exemple conception-orientee.add

import accepte adder.py / adder.[ ].so ou adder/__fichier__.py

<u>Python wapper function</u> pour assurer qu'il n'y aura pas de souci de types, plus convivial pour l'utilisateur.

def add(first : int | str
↳ accepte un type ou l'autre

<u>Informations sur une fonction :</u> conception_orientee_object.binary_add ? affiche la doc

ou conception-orientee_object.binary ?? affiche le chemin.

<u>Module de test unitaire unitest :</u>

```
import unitest
from conception_orientee object import add
class TestAdder (unitest.Testcase):
    def test_adder_integer(self):
        self.assert.Equal(4, 3), 7)

if __name__ = "__main__"
```

test d'égalité → assert.Equal

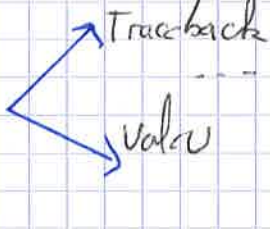
Exécute le main si appel de test

Astuce : Ajouter ce fichier à la liste de test contenu dans la variable PYTHONPATH

↳ Dans le fichier CMakeList if (BUILD_PYTHON_INTERFACE)
add_test(
NAME test-python
COMMAND ${Python_EXECUTABLE}
-m unittest
WORKING DIRECTORY ${CMAKE_CURRENT_SOURCE_
DIR}
set_tests_properties (test-python PROPERTIES
PYTHONPATH = ${CMAKE_BINARY_

Doctests """ pour un commentaire qui s'étend sur plusieurs lignes
On s'en sert par créer de la documentation (placée en début de class ou de fonction).
Les doctests s'assure que les exemples pris dans la doc sont bien le résultat de la fonction décrite doit être introduit par 3 chevrons >>> add(2, 6)
8

Pour prévoir un message d'erreur >>> add(-1, "ga")



seulement la 1ère et la dernière ligne

Mise en place des outils : intégration continue (vérification intégré à l'envoi)

Fichier de pre-commit-config.yaml

Après l'intégration de pre-commit, l'exécution des outils s'est faite une fois.
le symbole ✗ signifie que le pre-commit ne s'est pas bien passé.

Configuration de actions paramètre l'intégration continue

name : build and test nom de l'action

on : [push, pull_request] quand l'action est-elle exécuté

jobs:
build:
run-on : ubuntu-lastest
steps :
uses : actions/check@v3

Penser à demander l'affichage des erreurs

Dans l'onglet workflow, on voit les tests et vérifications effectués

Attention à la différence entre la version de Python utilisée sur la machine locale et les vérifications d'intégration continue effectuées par GitHub.

=> il faut parfois spécifier la version de Python à utiliser

Possibilité d'utiliser le projet conception-orientee-objet comme template (cf. paramètre lors de la création d'un repository