Create a Custom Python Script in the Paratext Menu

Paratext allows users to create custom Python scripts and to add them to its Custom Tools menu. These scripts are found in the cms folder inside your My Paratext Projects folder. This tutorial will show you how to write these scripts and how to share them with other Paratext users.

What is it for?

These scripts were originally designed to allow Paratext users to create their own custom Scripture checks. The scripts can read the translation text, check for issues, and generate a report. However, your creativity may discover other uses for these tools.

Commands

Every script begins with these two imports:

import re[1]
import sys[2]

import codecs[3]
import os[4]
import _ _builtin_ _[5]
import copy[6]
import csv[7]
import difflib[8]
import xml.etree.ElementTree[9]
import glob[10]
import pickle[11]
import string[12]
import time[13]
import types[14]
import unittest[15]
import zipfile[16]

Standard scripts to help with Paratext functions

SetupChecklist.py

- Sets up the registry to run a checklists test

ScriptureObjects.py

This module contains all the Python objects necessary to access Paratext 6 project data:

- **Reference** - A point in the text, e.g. MAT 3:11
- **Tag** - Information for a single marker from the stylesheet
- **ScriptureText** - The text itself
- **Language** - Information about the language for this text

~~**ScriptureChecksStorage.py**~~

Not funtional. Proper names and biblical term renderings were accessed via this module when kb2 files were used to store Biblical terms. Currently Biblical terms are stored in TermRenderings.xml. Maybe somone can rewrite this to work?

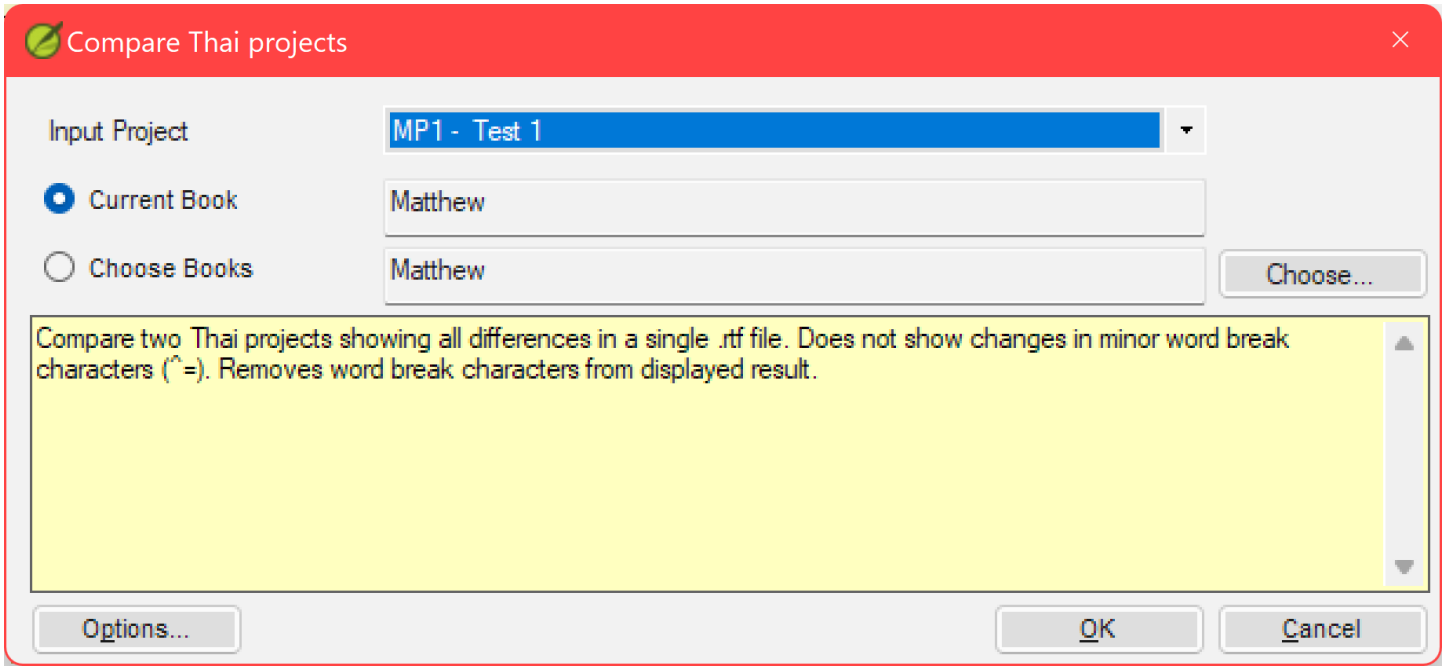
Creating your CMS file

\check

This is followed by a space and the name of the check. This check will be featured in the project menu of Paratext under Tools -> Custom tools

\description

The description will show in the dialog that pops up when selecting the check from the menu:



\subMenu

You can create a new submenu for your scripts or include them in one of the existing submenus just by naming it here. Many scripts are listed in the Unsupported menu, but we are not a fan of unsupported products. If your script is designed to help people, wouldn't you want to follow through?

\rank

This determines where your plugin shows in the list. Rank 1 is near the top. If only Google made it this easy...

\books

Places the Book selector on the dialog:

\check
\description
\helpFile
\noText
\notext
\optionDefault
\optionDescription
\optionHidden
\optionListAllTexts
\optionLocalized
\optionLocalizedName
\optionName
\outputProject
\parameter1

\parameter2
\programToRun
\rank
\script
\subMenu
\submenu
\supportMessage
\toHtml
\toList
\toText
\utf8
\wordlist

Collecting user input

The following options are available for collecting user input:

\optionName
\optionLocalizedName
\optionDefault
\optionDescription
\optionHidden
\optionListAllTexts
\optionLocalized

---

[1] https://docs.python.org/2.7/library/re.html
[2] https://docs.python.org/2.7/library/sys.html
[3] https://docs.python.org/2.7/library/codecs.html
[4] https://docs.python.org/2.7/library/os.html
[5] https://docs.python.org/2.7/library/%3Cstrong%3Ebuiltin%3C/strong%3E.html
[6] https://docs.python.org/2.7/library/copy.html
[7] https://docs.python.org/2.7/library/csv.html
[8] https://docs.python.org/2.7/library/difflib.html
[9] https://docs.python.org/2.7/library/xml.etree.elementtree.html
[10] https://docs.python.org/2.7/library/glob.html
[11] https://docs.python.org/2.7/library/pickle.html
[12] https://docs.python.org/2.7/library/string.html
[13] https://docs.python.org/2.7/library/time.html
[14] https://docs.python.org/2.7/library/types.html
[15] https://docs.python.org/2.7/library/unittest.html
[16] https://docs.python.org/2.7/library/zipfile.html