Files, Errors and Warnings of `pythontex` 0.18

Ernst Reissner (rei3ner@arcor.de)

Contents

List of Figures

List of Tables

Listings

1 Introduction

This document is created with `lualatex` or that like with output format PDF. The package `tex4ht` is not loaded.

The `pythontex` package together with the auxiliary program with the same name `pythontex`, allows including code, e.g. in Python into a LaTeX document. This document describes the input/output behavior of the auxiliary program `pythontex`, version 0.18 which includes all files read and written and uses pythontex. For example, `1+1=2` has been computed by python.

Interaction of `pythontex` with a LaTeX-to-pdf converter like `lualatex` is comparable to that of other auxiliary programs like `makeindex`: A latex package makes the LaTeX-to-pdf converter extract information for the auxiliary program into a separate file or more. Then the auxiliary program is run which creates further files which the LaTeX-to-pdf converter reads in a second run.

Both, the package `pythontex` and the auxiliary programs `pythontex` and `depythontex`, are described in [Poo21]. Moreover, there is an introduction [Poo] and a gallery [Poo17]. For background on the intentions of package `pythontex`, consult [Poo15].

The integration of `pythontex` into the latex maven plugin in this project is given in [Rei], Section 5.5.

Another source of knowledge on pythontex is the source code hosted at `https://github.com/gpoore/pythontex`. Note that `pythontex` is written in python and we only take into account the code for python3.

At least the following properties are special to package `pythontex`:

- The number of files `pythontex` may create is variable and so by default they are put into a subfolder.
- The output files generated are highly configurable.
- There is more than one auxiliary program tied to the package, besides `pythontex` also `depythontex`.
- The errors and warnings of a `pythontex` run and of a `depythontex` run are not written into a log file.

In [Rei], Section 5.5 a wrapper for `pythontex` is suggested writing the errors and warnings normally coming at standard output or error output into a log file `xxx.plg`. Nevertheless, currently no log file is written.

The package `pythontex` is highly configurable, more than this software allows. Thus, also in this document we assume that neither `\setpythontexoutputdir` setting the output directory nor `\setpythontexworkingdir` setting the working directory are used, because this software assumes the default that the working directory is the directory containing the LaTeX main file `xxx.tex` and the output directory is in the working directory and its name is `pythontex-files-xxx`.

Note that we assume python 3.x is installed only.

2 The converter `pythontex`

As already pointed out in the introduction, we restrict ourselves to the default case in which `pythontex` writes output files only in folder `pythontex-files-xxx`.

Under these assumptions, Figure 1 shows the input and output files of `pythontex`.

The input file is described in Section 2.1 in full detail. Section 2.2 is devoted to the output files of `pythontex`. Note that unlike the wrapper `pythontexW`, the original `pythontex` just prints

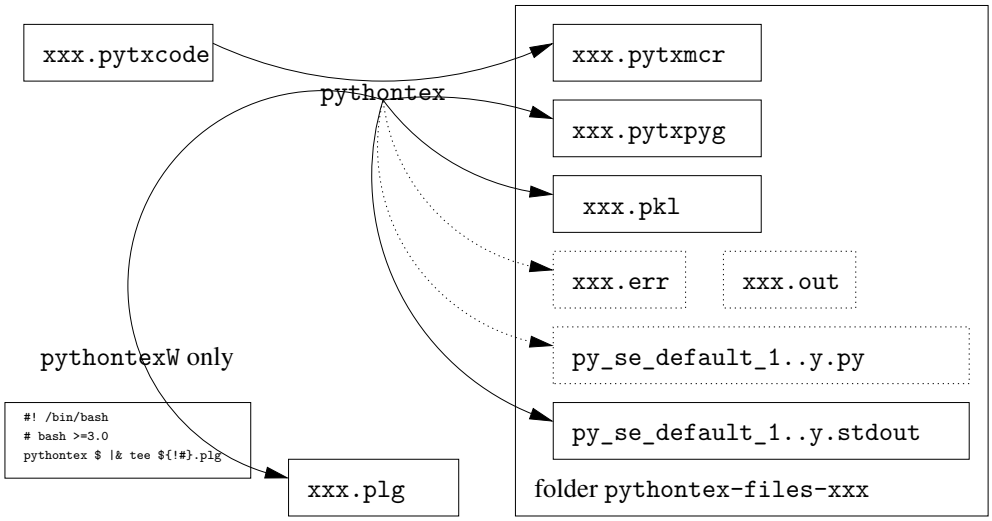

Figure 1: Conversion of a `pytxcode`-file using `pythontex`

errors and warnings. These are all collected in Section 2.3. Finally, Section 2.4 is on the failure codes.

2.1 The Input File `xxx.pytxcode`

If a file `xxx.tex` loading package `pythontex` is processed, as is the case for this document, a file `xxx.pytxcode` is created, whether there is python code within `xxx.tex` or not. This file contains a line

```
=>PYTHONTEX:SETTINGS#
```

and below that are specified the package options in the form given by Listing 1.

There is one key which does not refer to a package option: it is `version` which refers to the version of the `pythontex` package which is also the expected version of `pythontex`. If the versions deviate, running `pythontex` emits the fatal error with line number 491 in Table 1.

Interesting: `runall` is a package option, but it is not a valid key in `xxx.pytxcode`: instead, `runall=true/false` is converted into `rerun=always/default`. Note that `pythontex` is not able to process the key `runall` but emits a warning with line number 484 given in Table 4. This document is compiled with option `runall=false`.

For each python code in `xxx.tex`, there is a separate code section in `xxx.pytxcode`. The code sections come in proper order and precede the settings section. This document has a single section with python code, right at the beginning of the introduction. The code is

```
\pys[sname]{1+1=!{1+1}}
```

Listing 2 shows the according section in `xxx.pytxcode`. As always there is a headline starting with `=>PYTHONTEX` then follow, separated by `#` symbols

- the family, i.e. the interpreter, here `py` representing python, coming from the command `\pys`; accordingly for environments,
- the session, here `sname`, which is the optional parameter of the command,

```
=>PYTHONTEX:SETTINGS#
version=0.18
outputdir=pythontex-files-pythontexInOut
workingdir=.
workingdirset=false
gobble=none
rerun=default
hashdependencies=default
makestderr=false
stderrfilename=full
keeptemps=all
pyfuture=default
pyconfuture=none
pygments=false
pygglobal=:GLOBAL||
fvextfile=55
pyconbanner=none
pyconfilename=stdin
depythontex=true
```

Listing 1: The settings section of `pythontexInOut.pytxcode`

```
=>PYTHONTEX#py#sname#default#0#s#####142#
1+1=!{1+1}
```

Listing 2: The sole code section of `pythontexInOut.pytxcode`

- next suspected the restart identifier, seemingly always `default`
- the command, here `s`, also determined by the command `\pys`,
- the context which is empty,
- arguments which are empty here,
- the number of the instance, which runs from 0 to the number of commands minus one
- and the line number which is the line in the LaTeX file, where the command or the according environment starts.

If running `pythontex` on the job `xxx`, we obtain for this manual with a trailing empty line.

```
This is PythonTeX 0.18

------------------------------------------------
PythonTeX:  manualLMP - 0 error(s), 0 warning(s)
```

The folder `pythontex-files-manualLMP` is created but may be empty because there is no code.

2.2 The Output Files

Figure 1 shows that the output files of `pythontex` are all in folder `pythontex-files-xxx`. Temporary files in dotted boxes, so these can be seen only if the `pythontex` run is interrupted, e.g. by failure. The other files are called *final*. The Figure also indicates, that the wrapper `pythontexW` writes a log file in addition.

Among the final files, there is `xxx.pytxmcr` which starts something like

```
%Last time of file creation:  1656851667.5282867
```

and contains processed `pygments` code according to [Poo21], page 107.

Although indicates the time of the last `pythontex` wrote the file, seemingly, `pythontex` does not update if it is unchanged. So it does not indicate the last run.

2.3 Errors and Warnings at standard/error output

Line No	*	Message	RC
219	—	`Invalid --interpreter argument`	2
246	error	`You have launched PythonTeX using pythontex2/3.py directly.`	2
271	error	`You have launched PythonTeX using pythontex2/3.py directly.`	2
292	error	`Code file xxx.pytxcode does not exist. Run LaTeX …`	1
327	error	`Directory naming collision between the following files:…`	1
362	error	`Code file xxx.pytxcode does not exist. Run LaTeX …`	1
370	—	`The .pytxcode file appears to have an outdated format …` `Run LaTeX to make sure the file is current`	1
406	error	`Unable to parse package option fvextfile.`	1
491	error	`The version of the PythonTeX scripts does not match` `the last code saved by the document …`	1
2864	—	—	1

Table 1: Fatal errors with number code of `pythontex`

Fatal errors are understood to be those which exit via `sys.exit` *immediately* with an error code other than 0 or with a message. Thus, at most one fatal error can occur. This document doesn't treat fatal errors with error message. Table 1 gives an overview of fatal errors with number code. It gives the line number of the according `sys.exit` command, whether the message is preceded by a line `* PythonTeX error`, followed by an abbreviation of the proper message and finally the error code. Of course, just the error code indicates that it is an error, even if the message does not start with `* PythonTeX error`, and for the cases with lines 219 and 370, the missing indication in the message seems to be just a bug or a weakness.

A special case is in line 2864: `pythontex` has a counter for non-fatal errors and another counter for warnings. Error code 1 is returned also, if at least one non-fatal error was counted and if command line option `--error-exit-code` is set to `true`, which is the default according to [Poo21], Section 3.2. Fatal errors cannot be suppressed via that command line option. Strictly speaking, the case of line 2864 is a fatal error itself, but it shall not be treated as such, and it is appropriate that it has no message, just an error code if configured so.

In contrast, non-fatal errors, are errors which do not immediately cause `pythontex` to exit. Have a look at the messages collected in Table 2: It is structured similar to Table 1 except that the exit code (which non-fatal errors don't have) is replaced by the line number of the increment of the error counter, except in one case, where there is no increment at all. In this case, the line number, which is 2422, refers to the error message. The author considers this a bug in `pythontex`. Observe that all these errors have messages starting with `* PythonTeX error`. All entries in Table 2 refer to non-fatal errors: They are errors because of either the message or the increment of the error count. Also, they are non-fatal because they don't lead to an immediate exit with failure code.

Line No	*	Message	inc err/warn
655	error	Cannot find dependency …	e
1359	error	Currently, non-Python consoles are not supported	e
1605	error	Missing output file for …	e
1611	error	Running code for Julia console failed	e
1696	error	Cannot find dependency. It belongs to …	e
1765	error	Missing stderr file for …	e
1960	error	Line number xxx could not be synced with the document …	e
2343	error	An error occurred but no error messages were identified. …	e
2422	error	Could not find external file xxx The file was not pygmentized	—

Table 2: Non-fatal errors of `pythontex`

Table 3 collects messages starting with `* PythonTeX stderr` indicating that they are handed over from included code. They are treated either as errors or warnings, increasing exactly one of the according counters, which is indicated by the last column of the table. Note that the message gives no indication on whether it is counted as an error or as a warning: One and the same message form can be both. Distinction is just by the counter incremented. Since there are at least two lines of code where the increments are performed, at least one for an error and one for a warning, the line number given in the table refers to the code where the message `* PythonTeX stderr` is printed.

Without the irregularity given in Table 2 line with number 2422, a non-fatal error is just tied with messages for which the error count is incremented. The irregularity can be included by specifying that a non-fatal error is if the error counter is incremented or the message starts with `* PythonTeX error`.

Line No	*	Message	inc err/warn
1899	stderr	…on line …in "…"	e/w
1899	stderr	…on line …	e/w
2061	stderr	…near line …in "…"	e/w
2063	stderr	…near line …	e/w
2164	stderr	…near line …in "…"	e/w
2166	stderr	…near line …	e/w
2654	stderr	…in console startup code	e/w
2677	stderr	…near line …in custom code for console	e/w
2679	stderr	…near line …in console code	e/w

Table 3: StdErr (non-fatal) errors of `pythontex`

Table 4 contains proper warnings always incrementing the warnings counter. So definition of warnings is simple: A warning is what increases the warning counter.

Line No	*	Message	inc warn
340	warning	Potential directory naming collision …	yes
413	warning	Invalid value for package option fvextfile	yes
484	warning	Unknown option …	yes
685	warning	Session xxx has rerun=never	yes
		But its code or dependencies have been modified	
1446	warning	The following have dependencies that have been modified	yes
1737	warning	Custom code for xxx attempted to print or write to stdout	yes

Table 4: Warnings of `pythontex`

Finally, Table 5 states notices, seemingly mere info not directly tied to an error or a warning. The according message is identified by its setart `* PythonTeX stderr`.

Line Number	*	Message
2276	notice	Line number ...could not be synced with the document
2336	notice	x message(s) could not be classified Interpreted as y, based on the return code(s)

Table 5: Notices of `pythontex`

It is difficult to analyze the code around line 2276, but it seems as if synchronization of line numbers occurs only in conjunction with non-fatal errors and warnings, because synchronization is needed only to locate those in code text.

Analyzing the code preceding line 2336 shows that the according notice comes up only if a stderr message could not be identified as a non-fatal error or a warning, so both counters are increased by the number of events which could not be classified. So again, an error or a warning is indicated by a nonzero counter, but one of the counters may be too high.

So also notices are recognized via error count and warning count.

At the end of the log, which is currently written to stdio, `pythontex` summarizes the (non-fatal) errors and warnings which occurred. So besides the proper messages, there is a summary. It takes one of the following forms: Either

```
------------------------------------------------
PythonTeX:␣␣pythontexInOut
␣␣␣␣-␣Old:␣␣␣␣␣␣␣0␣error(s),␣0␣warning(s)
␣␣␣␣-␣Current:␣␣0␣error(s),␣0␣warning(s)
```

or

```
------------------------------------------------
PythonTeX:␣␣pythontexInOut␣-␣0␣error(s),␣0␣warning(s)
```

where of course `pythontexInOut` is to be replaced by the jobname and the number of errors and warnings may be different from 0.

The pattern to match at least one non-fatal error in java style is

```
(PythonTeX:␣␣.+␣-|␣␣␣␣-␣Current:␣)␣[1-9][0-9]*␣error\\(s\\),␣[0-9]+␣warning\\(s\\)
```

The last thing to do is to take into account the irregularity of the non-fatal error in Table 2, line number 2422, by adding the alternative `* PythonTeX error` to the regular expression, which detects nothing additional but this case. So we arrive at the following pattern, where the dots must be replaced by the above pattern

```
\\*␣PythonTeX␣error|...
```

For warnings, we can use the same but do not need the bugfix. The result is

```
(PythonTeX:␣␣.+␣-|␣␣␣␣-␣Current:␣)␣[0-9]+␣error\\(s\\),␣[1-9][0-9]*␣warning\\(s\\)
```

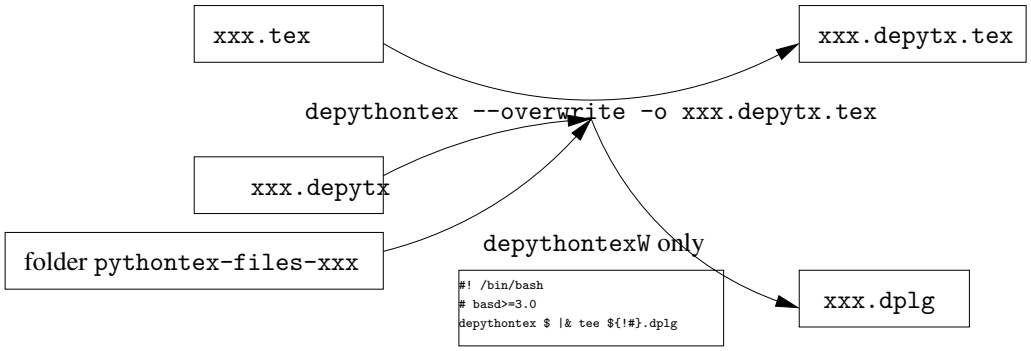

Figure 2: Conversion of a `depytx`-file using `depythontex`

2.4 Failure codes

3 The converter `depythontex`

3.1 The Input File `xxx.dpytx`

3.2 The Output Files

3.3 Errors and Warnings at standard/error output

3.4 Failure codes

4 References

[Poo] Geoffrey M. Poore. PythonTEX Quick-start. `https://github.com/gpoore/pythontex/blob/master/pythontex_quickstart/pythontex_quickstart.pdf`.

[Poo15] Geoffrey M. Poore. PythonTeX: reproducible documents with LaTeX, Python, and more. *Computational Science & Discovery*, 8(1), 7 2015. doi:10.1088/1749-4699/8/1/014010.

[Poo17] Geoffrey M. Poore. PythonTEX Gallery. `https://github.com/gpoore/pythontex/blob/master/pythontex_gallery/pythontex_gallery.pdf`, 7 2017.

[Poo21] Geoffrey M. Poore. *The pythontex package*. gpoore at gmail.com, `github.com/gpoore/pythontex`, v1.8 edition, 6 2021.

[Rei] E. Reißner. *Manual for the latex-maven-plugin and for an according ant-task, Version X.Y*. The current version is vailable at `http://www.simuline.eu/LatexMavenPlugin/manualLMP.pdf`.