35.10. `pipes` — Interface to shell pipelines

Source code: Lib/pipes.py

The `pipes` module defines a class to abstract the concept of a *pipeline* — a sequence of converters from one file to another.

Because the module uses **/bin/sh** command lines, a POSIX or compatible shell for `os.system()` and `os.popen()` is required.

The `pipes` module defines the following class:

class `pipes.`**`Template`**

An abstraction of a pipeline.

Example:

```
>>> import pipes
>>> t = pipes.Template()
>>> t.append('tr a-z A-Z', '--')
>>> f = t.open('pipefile', 'w')
>>> f.write('hello world')
>>> f.close()
>>> open('pipefile').read()
'HELLO WORLD'
```

35.10.1. Template Objects

Template objects following methods:

`Template.`**`reset`**()

Restore a pipeline template to its initial state.

`Template.`**`clone`**()

Return a new, equivalent, pipeline template.

`Template.`**`debug`**(*flag*)

If *flag* is true, turn debugging on. Otherwise, turn debugging off. When debugging is on, commands to be executed are printed, and the shell is given `set -x` command to be more verbose.

`Template.`**`append`**(*cmd*, *kind*)

Append a new action at the end. The *cmd* variable must be a valid bourne shell command. The *kind* variable consists of two letters.

The first letter can be either of `'-'` (which means the command reads its standard input), `'f'` (which means the commands reads a given file on the command line) or `'.'` (which means the commands reads no input, and hence must be first.)

Similarly, the second letter can be either of `'-'` (which means the command writes to standard output), `'f'` (which means the command writes a file on the command line) or `'.'` (which means the command does not write anything, and hence must be last.)

`Template.`**`prepend`**(*cmd*, *kind*)

Add a new action at the beginning. See `append()` for explanations of the arguments.

`Template.`**`open`**(*file*, *mode*)

Return a file-like object, open to *file*, but read from or written to by the pipeline. Note that only one of `'r'`, `'w'` may be given.

`Template.`**`copy`**(*infile*, *outfile*)

Copy *infile* to *outfile* through the pipe.