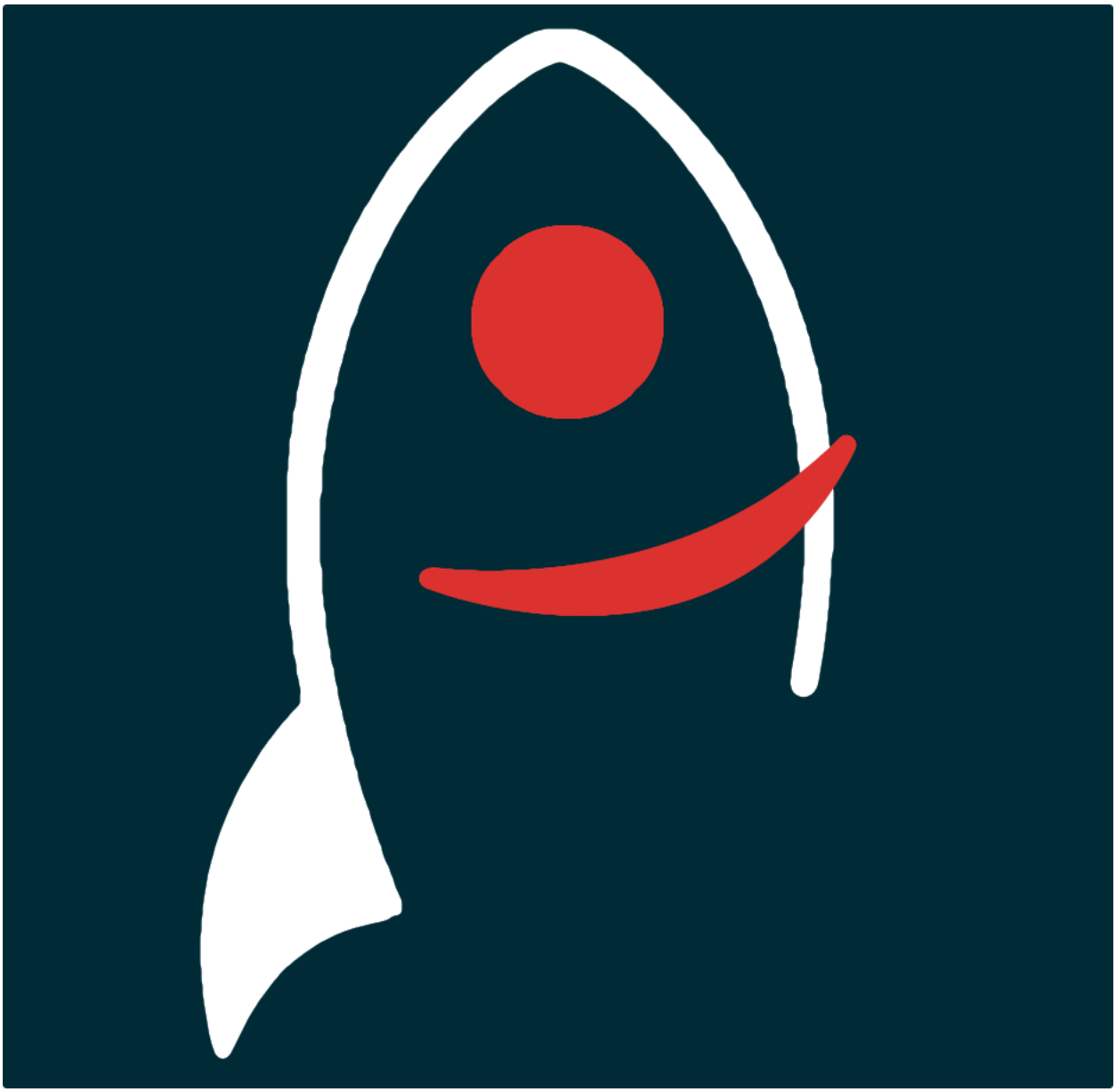

polyglot Documentation

Release 1.0.0

Dave Young

2017

Getting Started

A python package and command-line tools for translating documents and webpages to various markup languages and document formats (html, epub, mobi ..). Here's a summary of what's included in the python package:

Classes

`polyglot.ebook`	*The worker class for the ebook module*
`polyglot.htmlCleaner`	*A parser/cleaner to strip a webpage article of all cruft and neatly present it with some nice css*
`polyglot.kindle`	*Send documents or webpage articles to a kindle device or app*
`polyglot.printpdf`	*PDF printer*
`polyglot.webarchive`	*Generate a macOS webarchive given the URL to a webpage *
`polyglot.markdown.kindle_notebook`	*convert the HTML export of kindle notebooks (from kindle apps) to markdown*
`polyglot.markdown.translate`	*The Multimarkdown translator object*

Functions

CHAPTER 1

Installation

The easiest way to install polyglot is to use `pip`:

```
pip install polyglot
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/polyglot.git
cd polyglot
python setup.py install
```

To upgrade to the latest version of polyglot use the command:

```
pip install polyglot --upgrade
```

Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/polyglot.git
cd polyglot
python setup.py develop
```

Pull requests are welcomed!

Sublime Snippets

If you use Sublime Text as your code editor, and you're planning to develop your own python code with polyglot, you might find my Sublime Snippets useful.

Issues

Please report any issues here.

CHAPTER 2

Command-Line Usage

```
Documentation for polyglot can be found here: http://pypolyglot.readthedocs.org/en/
↪stable

Translate documents and webpages to various markup languages and document formats␣
↪(html, epub, mobi ..)

Usage:
    polyglot init
    polyglot [-oc] (pdf html epub mobi) <url> [<destinationFolder> -f <filename> -s
↪<pathToSettingsFile>]
    polyglot kindle <url> [-f <filename> -s <pathToSettingsFile>]
    polyglot [-o] (epub mobi) <docx> [<destinationFolder> -f <filename> -s
↪<pathToSettingsFile>]
    polyglot kindle <docx> [-f <filename> -s <pathToSettingsFile>]
    polyglot [-o] kindleNB2MD <notebook> [<destinationFolder> -s <pathToSettingsFile>]

Options:
    init                                                              setup the␣
↪polyglot settings file for the first time
    pdf                                                               print webpage to␣
↪pdf
    html                                                              parse and␣
↪download webpage to a local HTML document
    epub                                                              generate an epub␣
↪format book from a webpage URL
    kindle                                                            send webpage␣
↪article straight to kindle

    -h, --help                                                        show this help␣
↪message
    -v, --version                                                     show version
    -o, --open                                                        open the document␣
↪after creation
    -c, --clean                                                       add polyglot's␣
↪clean styling to the output document
```

```
    <url>                                                    the url of the␣
↪article's webpage
    <docx>                                                   path to a DOCX␣
↪file
    -s <pathToSettingsFile>, --settings <pathToSettingsFile>        path to␣
↪alternative settings file (optional)
    <destinationFolder>                                      the folder to␣
↪save the parsed PDF or HTML document to (optional)
    -f <filename>, --filename <filename>                     the name of the␣
↪file to save, otherwise use webpage title as filename (optional)
```

CHAPTER 3

Documentation

Documentation for polyglot is hosted by Read the Docs (last stable version and latest version).

CHAPTER 4

Command-Line Tutorial

Before you begin using polyglot you will need to populate some custom settings within the polyglot settings file.

To setup the default settings file at `~/.config/polyglot/polyglot.yaml` run the command:

```
polyglot init
```

This should create and open the settings file; follow the instructions in the file to populate the missing settings values (usually given an `XXX` placeholder).

polyglot often relies on a bunch on other excellent tools to get it's results like electron-pdf, pandoc and kidlegen. Depending on how you use polyglot, these tools may need to be install on your system.

To read the basic usage intructions just run `polyglot -h`.

Webpage Article to HTML document

To generate a parsed, cleaned local HTML document from a webpage at a given URL use polyglot's `html` command:

```
polyglot html https://en.wikipedia.org/wiki/Volkswagen
```

The filename for the output file is take from the webpage's title and output in the current directory. Here's the result.

To instead give both a destination output and a specified filename for the resulting document:

```
polyglot html https://en.wikipedia.org/wiki/Volkswagen ~/Desktop -f cars_and_stuff
```

To style the result with polyglots simple styling and easy to read fonts, use the `-c` flag:

```
polyglot -c html https://en.wikipedia.org/wiki/Volkswagen -f Volkswagen_Styled
```

See the result here.

Webpage Article to PDF

To instead print the webpage to PDF, you can either just print the original webpage:

```
polyglot pdf https://en.wikipedia.org/wiki/Volkswagen
```

with this result, or you can choose again to use polyglot's styling:

```
polyglot -c pdf https://en.wikipedia.org/wiki/Volkswagen -f Volkswagen_Styled
```

resulting in this PDF.

Note if you are going to be running polyglot in a windowless environment, to generate the PDFs with `electron-pdf <`https://github.com/fraserxu/electron-pdf>`_ you will need to install xvfb. To install and setup do something similar to the following (depending on your flavour of OS):

```
sudo apt-get install xvfb
```

then in whatever bash scripts you write add this before any polyglot commands:

```
export DISPLAY=':99.0'
Xvfb :99 -screen 0 1024x768x24 > /dev/null 2>&1 &
```

Webpage Article to eBook

To generate an epub book from a webpage article run the command:

```
polyglot epub http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-index.html
```

Here is the output of this command.

If you prefer a mobi output, use the command:

```
polyglot mobi http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-index.html
```

To get this mobi book.

Send a Webpage Article Straight to Your Kindle

Polyglot can go even further than creating a mobi ebook from the web-article; it can also send the ebook straight to your kindle device or smart phone app (or both at the same time) as long as you have the email settings populated in the polyglot settings file.

```
polyglot kindle http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-index.html
```

And here's the book appearing on a smart phone kindle app:

Converting Kindle Notebook HTML Exports to Markdown

On the Kindle app for iOS you can export an HTML document of your notes and annotations via email.

The colors of the annotation convert to markdown with the following color-key:

“blue”: “code”, “yellow”: “text”, “orange”: “quote”, “pink”: “header”

Todo

- add a tutorial to convert kindle annotations

CHAPTER 5

Installation

The easiest way to install polyglot is to use `pip`:

```
pip install polyglot
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/polyglot.git
cd polyglot
python setup.py install
```

To upgrade to the latest version of polyglot use the command:

```
pip install polyglot --upgrade
```

Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/polyglot.git
cd polyglot
python setup.py develop
```

Pull requests are welcomed!

Sublime Snippets

If you use Sublime Text as your code editor, and you're planning to develop your own python code with polyglot, you might find my Sublime Snippets useful.

Issues

Please report any issues here.

CHAPTER 6

Command-Line Usage

```
Documentation for polyglot can be found here: http://pypolyglot.readthedocs.org/en/
↪stable

Translate documents and webpages to various markup languages and document formats␣
↪(html, epub, mobi ..)

Usage:
    polyglot init
    polyglot [-oc] (pdf html epub mobi) <url> [<destinationFolder> -f <filename> -s
↪<pathToSettingsFile>]
    polyglot kindle <url> [-f <filename> -s <pathToSettingsFile>]
    polyglot [-o] (epub mobi) <docx> [<destinationFolder> -f <filename> -s
↪<pathToSettingsFile>]
    polyglot kindle <docx> [-f <filename> -s <pathToSettingsFile>]
    polyglot [-o] kindleNB2MD <notebook> [<destinationFolder> -s <pathToSettingsFile>]

Options:
    init                                                              setup the␣
↪polyglot settings file for the first time
    pdf                                                               print webpage to␣
↪pdf
    html                                                              parse and␣
↪download webpage to a local HTML document
    epub                                                              generate an epub␣
↪format book from a webpage URL
    kindle                                                            send webpage␣
↪article straight to kindle

    -h, --help                                                        show this help␣
↪message
    -v, --version                                                     show version
    -o, --open                                                        open the document␣
↪after creation
    -c, --clean                                                       add polyglot's␣
↪clean styling to the output document
```

```
    <url>                                                    the url of the␣
↪article's webpage
    <docx>                                                   path to a DOCX␣
↪file
    -s <pathToSettingsFile>, --settings <pathToSettingsFile>        path to␣
↪alternative settings file (optional)
    <destinationFolder>                                      the folder to␣
↪save the parsed PDF or HTML document to (optional)
    -f <filename>, --filename <filename>                     the name of the␣
↪file to save, otherwise use webpage title as filename (optional)
```

CHAPTER 7

Documentation

Documentation for polyglot is hosted by Read the Docs (last stable version and latest version).

CHAPTER 8

Command-Line Tutorial

Before you begin using polyglot you will need to populate some custom settings within the polyglot settings file.

To setup the default settings file at `~/.config/polyglot/polyglot.yaml` run the command:

```
polyglot init
```

This should create and open the settings file; follow the instructions in the file to populate the missing settings values (usually given an `XXX` placeholder).

polyglot often relies on a bunch on other excellent tools to get it's results like electron-pdf, pandoc and kidlegen. Depending on how you use polyglot, these tools may need to be install on your system.

To read the basic usage intructions just run `polyglot -h`.

Webpage Article to HTML document

To generate a parsed, cleaned local HTML document from a webpage at a given URL use polyglot's `html` command:

```
polyglot html https://en.wikipedia.org/wiki/Volkswagen
```

The filename for the output file is take from the webpage's title and output in the current directory. Here's the result.

To instead give both a destination output and a specified filename for the resulting document:

```
polyglot html https://en.wikipedia.org/wiki/Volkswagen ~/Desktop -f cars_and_stuff
```

To style the result with polyglots simple styling and easy to read fonts, use the `-c` flag:

```
polyglot -c html https://en.wikipedia.org/wiki/Volkswagen -f Volkswagen_Styled
```

See the result here.

Webpage Article to PDF

To instead print the webpage to PDF, you can either just print the original webpage:

```
polyglot pdf https://en.wikipedia.org/wiki/Volkswagen
```

with this result, or you can choose again to use polyglot's styling:

```
polyglot -c pdf https://en.wikipedia.org/wiki/Volkswagen -f Volkswagen_Styled
```

resulting in this PDF.

Note if you are going to be running polyglot in a windowless environment, to generate the PDFs with `electron-pdf <`https://github.com/fraserxu/electron-pdf>`_ you will need to install xvfb. To install and setup do something similar to the following (depending on your flavour of OS):

```
sudo apt-get install xvfb
```

then in whatever bash scripts you write add this before any polyglot commands:

```
export DISPLAY=':99.0'
Xvfb :99 -screen 0 1024x768x24 > /dev/null 2>&1 &
```

Webpage Article to eBook

To generate an epub book from a webpage article run the command:

```
polyglot epub http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-index.html
```

Here is the output of this command.

If you prefer a mobi output, use the command:

```
polyglot mobi http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-index.html
```

To get this mobi book.

Send a Webpage Article Straight to Your Kindle

Polyglot can go even further than creating a mobi ebook from the web-article; it can also send the ebook straight to your kindle device or smart phone app (or both at the same time) as long as you have the email settings populated in the polyglot settings file.

```
polyglot kindle http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-index.html
```

And here's the book appearing on a smart phone kindle app:

Converting Kindle Notebook HTML Exports to Markdown

On the Kindle app for iOS you can export an HTML document of your notes and annotations via email.

The colors of the annotation convert to markdown with the following color-key:

“blue”: “code”, “yellow”: “text”, “orange”: “quote”, “pink”: “header”

Todo

- add a tutorial to convert kindle annotations

Subpackages

polyglot	
polyglot.commonutils	*common tools used throughout package*
polyglot.markdown	*Generate a markdown rendering from various sources and formats*

polyglot (*subpackage*)

polyglot.commonutils (*subpackage*)

common tools used throughout package

polyglot.markdown (*subpackage*)

Generate a markdown rendering from various sources and formats

Modules

polyglot.cl_utils	Documentation for polyglot can be found here: http://pypolyglot.readthedocs.org/en/stable
polyglot.utKit	*Unit testing tools*

polyglot.cl_utils (*module*)

Documentation for polyglot can be found here: http://pypolyglot.readthedocs.org/en/stable

Translate documents and webpages to various markup languages and document formats (html, epub, mobi ..)

Usage: polyglot init polyglot [-oc] (pdf|html|epub|mobi) <url> [<destinationFolder> -f <filename> -s <pathToSettingsFile>] polyglot kindle <url> [-f <filename> -s <pathToSettingsFile>] polyglot [-o] (epub|mobi) <docx> [<destinationFolder> -f <filename> -s <pathToSettingsFile>] polyglot kindle <docx> [-f <filename> -s <pathToSettingsFile>] polyglot [-o] kindleNB2MD <notebook> [<destinationFolder> -s <pathToSettingsFile>]

Options: init setup the polyglot settings file for the first time pdf print webpage to pdf html parse and download webpage to a local HTML document epub generate an epub format book from a webpage URL kindle send webpage article straight to kindle

-h, --help	show this help message
-v, --version	show version
-o, --open	open the document after creation

-c, --clean add polyglot's clean styling to the output document

<url> the url of the article's webpage <docx> path to a DOCX file -s <pathToSettingsFile>, –settings <pathToSettingsFile> path to alternative settings file (optional) <destinationFolder> the folder to save the parsed PDF or HTML document to (optional) -f <filename>, –filename <filename> the name of the file to save, otherwise use webpage title as filename (optional)

polyglot.cl_utils.**main**(*arguments=None*)
The main function used when ''cl_utils.py'' is run as a single script from the cl, or when installed as a cl command

polyglot.utKit (*module*)

Unit testing tools

class polyglot.utKit.**utKit**(*moduleDirectory*)
Override dryx utKit

Classes

polyglot.ebook	*The worker class for the ebook module*
polyglot.htmlCleaner	*A parser/cleaner to strip a webpage article of all cruft and neatly present it with some nice css*
polyglot.kindle	*Send documents or webpage articles to a kindle device or app*
polyglot.printpdf	*PDF printer*
polyglot.webarchive	*Generate a macOS webarchive given the URL to a webpage *
polyglot.markdown.kindle_notebook	*convert the HTML export of kindle notebooks (from kindle apps) to markdown*
polyglot.markdown.translate	*The Multimarkdown translator object*
polyglot.utKit.utKit	*Override dryx utKit*

polyglot.ebook (*class*)

class polyglot.**ebook**(*log*, *settings*, *urlOrPath*, *outputDirectory*, *bookFormat*, *title=False*, *header=False*, *footer=False*)
The worker class for the ebook module

Key Arguments:

- log – logger
- settings – the settings dictionary
- urlOrPath – the url or path to the content source
- bookFormat – the output format (epub, mobi)
- outputDirectory – path to the directory to save the output html file to.
- title – the title of the output document. I. False then use the title of the original source. Default *False*
- header – content to add before the article/book content in the resulting ebook. Default *False*
- footer – content to add at the end of the article/book content in the resulting ebook. Default *False*

Usage:

WebToEpub

To generate an ebook from an article found on the web, using the webpages's title as the filename for the book:

```
from polyglot import ebook
epub = ebook(
    log=log,
    settings=settings,
    urlOrPath="http://www.thespacedoctor.co.uk/blog/2016/09/26/
↪mysqlSucker-index.html",
    title=False,
    bookFormat="epub",
    outputDirectory="/path/to/output/folder"
)
pathToEpub = epub.get()
```

To add a header and footer to the epub book, and specify the title/filename for the book:

```
from polyglot import ebook
epub = ebook(
    log=log,
    settings=settings,
    urlOrPath="http://www.thespacedoctor.co.uk/blog/2016/09/26/
↪mysqlSucker-index.html",
    title="MySQL Sucker",
    bookFormat="epub",
    outputDirectory="/path/to/output/folder",
    header='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>',
    footer='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>'
)
pathToEpub = epub.get()
```

WebToMobi

To generate a mobi version of the webarticle, just switch *epub* for *mobi*:

```
from polyglot import ebook
mobi = ebook(
    log=log,
    settings=settings,
    urlOrPath="http://www.thespacedoctor.co.uk/blog/2016/09/26/
↪mysqlSucker-index.html",
    title="MySQL Sucker",
    bookFormat="mobi",
    outputDirectory="/path/to/output/folder",
    header='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>',
    footer='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>'
)
pathToMobi = mobi.get()
```

DocxToEpub

To instead convert a DOCX document to epub, simply switch out the URL for the path to the DOCX file, like so:

```
from polyglot import ebook
epub = ebook(
```

```
    log=log,
    settings=settings,
    urlOrPath="/path/to/Volkswagen.docx",
    title="A book about a car",
    bookFormat="epub",
    outputDirectory="/path/to/output/folder",
    header='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>',
    footer='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>'
)
pathToEpub = epub.get()
```

DocxToMobi

You can work it out yourself by now!

__init__(*log*, *settings*, *urlOrPath*, *outputDirectory*, *bookFormat*, *title=False*, *header=False*, *footer=False*)

Methods

__init__(log, settings, urlOrPath, ...[, ...])	
get()	*get the ebook object*

polyglot.htmlCleaner (*class*)

class polyglot.**htmlCleaner**(*log*, *settings*, *url*, *outputDirectory=False*, *title=False*, *style=True*, *metadata=True*, *h1=True*)

A parser/cleaner to strip a webpage article of all cruft and neatly present it with some nice css

Key Arguments:

- log – logger
- settings – the settings dictionary
- url – the URL to the HTML page to parse and clean
- outputDirectory – path to the directory to save the output html file to
- title – title of the document to save. If *False* will take the title of the HTML page as the filename. Default *False*.
- style – add polyglot's styling to the HTML document. Default *True*
- metadata – include metadata in generated HTML. Default *True*
- h1 – include title as H1 at the top of the doc. Default *True*

Usage:

To generate the HTML page, using the title of the webpage as the filename:

```
from polyglot import htmlCleaner
cleaner = htmlCleaner(
    log=log,
    settings=settings,
    url="http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-
↪index.html",
    outputDirectory="/tmp"
```

```
)
cleaner.clean()
```

Or specify the title of the document and remove styling, metadata and title:

```
from polyglot import htmlCleaner
cleaner = htmlCleaner(
    log=log,
    settings=settings,
    url="http://www.thespacedoctor.co.uk/blog/2016/09/26/mysqlSucker-
↪index.html",
    outputDirectory="/tmp",
    title="my_clean_doc",
    style=False,
    metadata=False,
    h1=False
)
cleaner.clean()
```

__init__(*log*, *settings*, *url*, *outputDirectory=False*, *title=False*, *style=True*, *metadata=True*, *h1=True*)

Methods

__init__(log, settings, url[, ...])	
clean()	*parse and clean the html document with Mercury Parser*

polyglot.kindle (*class*)

class polyglot.**kindle**(*log*, *settings*, *urlOrPath*, *title=False*, *header=False*, *footer=False*)

Send documents or webpage articles to a kindle device or app

Key Arguments:

- log – logger
- settings – the settings dictionary
- urlOrPath – the url or path to the content source
- title – the title of the output document. I. False then use the title of the original source. Default *False*
- header – content to add before the article/book content in the resulting ebook. Default *False*
- footer – content to add at the end of the article/book content in the resulting ebook. Default *False*

Usage:

To send content from a webpage article straight to your kindle device or smart phone app, you will first need to populate the email settings with polyglot's settings file at ~.config/polyglot/polyglot.yaml, then use the following code:

```
from polyglot import kindle
sender = kindle(
    log=log,
    settings=settings,
```

```
    urlOrPath="http://www.thespacedoctor.co.uk/blog/2016/09/26/
↪mysqlSucker-index.html",
    header='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>',
    footer='<a href="http://www.thespacedoctor.co.uk">thespacedoctor</a>'
)
success = sender.send()
```

Success is True or False depending on the success/failure of sending the email to the kindle email address(es).

__init__(*log*, *settings*, *urlOrPath*, *title=False*, *header=False*, *footer=False*)

Methods

__init__(log, settings, urlOrPath[, title, ...])	
get()	*get the ebook object*
get_attachment(file_path)	Get file as MIMEBase message
send()	*send the mobi book generated to kindle email address(es)*

polyglot.printpdf (*class*)

class polyglot.**printpdf**(*log*, *settings=False*, *url=False*, *title=False*, *folderpath=False*, *append=False*, *readability=True*)

PDF printer

Key Arguments:

- log – logger
- settings – the settings dictionary
- url – the webpage url
- title – title of pdf
- folderpath – path at which to save pdf
- append – append this at the end of the file name (not title)
- readability – clean text with Mercury Parser

Usage:

To print a webpage to PDF without any cleaning of the content using the title of the webpage as filename:

```
from polyglot import printpdf
pdf = printpdf(
    log=log,
    settings=settings,
    url="https://en.wikipedia.org/wiki/Volkswagen",
    folderpath="/path/to/output",
    readability=False
).get()
```

To give the PDF an alternative title use:

```
from polyglot import printpdf
pdf = printpdf(
    log=log,
    settings=settings,
    url="https://en.wikipedia.org/wiki/Volkswagen",
    folderpath="/path/to/output",
    title="Cars",
    readability=False
).get()
```

Or to append a string to the end of the filename before *.pdf* extension (useful for indexing or adding date created etc):

```
from datetime import datetime, date, time
now = datetime.now()
now = now.strftime("%Y%m%dt%H%M%S")

from polyglot import printpdf
pdf = printpdf(
    log=log,
    settings=settings,
    url="https://en.wikipedia.org/wiki/Volkswagen",
    folderpath="/path/to/output",
    append="_"+now,
    readability=False
).get()
```

To clean the content using the Mercury Parser and apply some simple styling and pretty fonts:

```
from polyglot import printpdf
pdf = printpdf(
    log=log,
    settings=settings,
    url="https://en.wikipedia.org/wiki/Volkswagen",
    folderpath=pathToOutputDir,
    readability=True
).get()
```

__init__(*log*, *settings=False*, *url=False*, *title=False*, *folderpath=False*, *append=False*, *readability=True*)

Methods

`__init__`(log[, settings, url, title, ...])	
`get`()	*get the PDF*

polyglot.webarchive (*class*)

class `polyglot.`**`webarchive`**(*log*, *settings=False*)

*Generate a macOS webarchive given the URL to a webpage *

Key Arguments:

- `log` – logger

- `settings` – the settings dictionary

Usage:

To setup your logger, settings and database connections, please use the `fundamentals` package (see tutorial here).

Generate a webarchive docuemnt from a URL, use the following:

```
from polyglot import webarchive
wa = webarchive(
    log=log,
    settings=settings
)
wa.create(url="https://en.wikipedia.org/wiki/Volkswagen",
          pathToWebarchive=pathToOutputDir + "Volkswagen.webarchive")
```

__init__(*log*, *settings=False*)

Methods

`__init__`(log[, settings])	
`create`(url, pathToWebarchive)	*create the webarchive object*

polyglot.markdown.kindle_notebook (*class*)

class `polyglot.markdown.`**kindle_notebook**(*log*, *kindleExportPath*, *outputPath*)

convert the HTML export of kindle notebooks (from kindle apps) to markdown

Key Arguments:

- `log` – logger
- `kindleExportPath` – path to the exported kindle HTML file
- `outputPath` – the output path to the md file.

Usage:

To convert the exported HTML file of annotation and notes from a kindle book or document to markdown, run the code:

```
from polyglot.markdown import kindle_notebook
nb = kindle_notebook(
    log=log,
    kindleExportPath="/path/to/kindle_export.html",
    outputPath="/path/to/coverted_annotations.md"
)
nb.convert()
```

The colours of the annotations convert to markdown attributes via the following key:

__init__(*log*, *kindleExportPath*, *outputPath*)

Methods

`__init__`(log, kindleExportPath, outputPath)	
`clean`(text)	
`convert`()	*convert the kindle_notebook object*
`convertToMD`(kindleNote)	
`find_component`(divtag, annotations)	

polyglot.markdown.translate (*class*)

class `polyglot.markdown.`**`translate`**(*log*, *settings=False*)

The Multimarkdown translator object

Key Arguments:

- `log` – logger
- `settings` – the settings dictionary

Usage:

To setup your logger, settings and database connections, please use the `fundamentals` package (see tutorial here).

To initiate a `translate` object, use the following:

```
from polyglot.markdown import translate
md = translate(
    log=log,
    settings=settings
)
```

`__init__`(*log*, *settings=False*)

Methods

`__init__`(log[, settings])	
`blockquote`(text)	*convert plain-text to MMD blockquote*
`bold`(text)	*convert plain-text to MMD bolded text*
`cite`(title[, author, year, url, publisher, ...])	*generate a MMD citation*
`code`(text)	*convert plain-text to MMD inline-code text*
`codeblock`(text[, lang])	*convert plain-text to MMD fenced codeblock*
`comment`(text)	*convert plain-text to MMD comment*
`definition`(text, definition)	*genarate a MMD definition*
`em`(text)	*convert plain-text to MMD italicised text*
`footnote`(text)	*convert plain-text to MMD footnote*
`glossary`(term, definition)	*genarate a MMD glossary*
`header`(text, level)	*convert plain-text to MMD header*
`headerLink`(headerText[, text])	*generate a link to a MMD header*
`hl`(text)	*convert plain-text to MMD critical markup highlighted text*
`image`(url[, title, width])	*create MMD image link*
`inline_link`(text, url)	*generate a MMD sytle link*
`math_block`(text)	*convert plain-text to MMD math block*
`math_inline`(text)	*convert plain-text to MMD inline math*

Continued on next page

Table 8.10 – continued from previous page

ol(text)	*convert plain-text to MMD ordered list*
strike(text)	*convert plain-text to HTML strike-through text*
ul(text)	*convert plain-text to MMD unordered list*
underline(text)	*convert plain-text to HTML underline text*
url(text)	*convert plain-text to MMD clickable URL*

polyglot.utKit.utKit (*class*)

class polyglot.utKit.**utKit**(*moduleDirectory*)

Override dryx utKit

__init__(*moduleDirectory*)

Methods

__init__(moduleDirectory)	
setupModule()	*The setupModule method*
tearDownModule()	*The tearDownModule method*

Functions

Indexes

- Module Index
- Full Index

Todo

- Todolist

Python Module Index

p

Index

Symbols

E

H

K

M

P

T

U

W