**Parser für LaTeX. valepTeX Grundideen.**

**Objekt com generieren, das entweder eine Instanz der Klasse com ist, falls diese existiert, sonst eine Instanz von Command**

# Assuming 'com' contains the name of the class

com = "STRING" # or "Command" if there's no subclass named STRING

# Check if the class exists in the global namespace

if com in globals() and issubclass(globals()[com], Command):

# If the class exists and is a subclass of Command, create an object of that class

obj = globals()[com]("Some content specific to the class")

else:

# If the class is not a subclass of Command, create an object of Command

obj = Command("Some generic content")

type(a).\_\_name\_\_ Name der Klasse von a

type(a).\_\_mro\_\_ tupel mit allen Superklassen einschließlich type(a) als erstes Element

To Do List:

    - Put all declarations on a stack and use them during the parsing process:

    -- replace commands and environments with the content of their declaration

 Ist eine eigene Sache wenn Macros Argumente haben (fraglich, ob das überhaupt behandelt werden soll?)

    -- replace theorems with the respective text

    -- distribute counter values

* Packages
	+ Load not only \input files with declarations but also style and documentclass files
	+ Load python packages
		- Check e.g. if a subclass chapter(Command) exists and assign chapter to that class
	+ Take care to all these things during the parsing process
* sectioning
	+ Identify those commands who define sections as type “section”
	+ Determine a value for an additional attribute “section\_level”
		- Number between 0 and 10 (or so)
		- 0 = part, 1 = chapter, 2 = section, 3 = subsection …
		8 = paragraph, 9 = subparagraph, 10 = subsubparagtraph

    - Process tables and tabbings and store them as table objects

* Verbatim content der eins zu eins beibehalten werden soll
* Grafiken, Blöcke und dergleichen …
* Umgang mit INhalt, der nicht konvertiert wird (raw html usw.)
* Whitespace was folgt auf \MACRO: Whitespace ignorieren, \WHITESPACE und {}WHITESPACE bedeuten, dass sein Whitespace also ‘ ‘ folgt

    - Deal with all kinds of irregular macros, special characters, escape sequences and the like e.g. \/ \@ \^ALPH\`ALPH \´ALPH \^ALPH \=y \~n \.p

      and store them all in specific objects

**Performance:** das Parsen ist wesentlich schneller, wenn der Prozess bei großen Files auf Portionen aufgeteilt wird. \include Anweisungen werden daher erst nach dem ersten Parsingdurchlauf aufgelöst. Bei \input Anweisungen, die sich ihrerseits in durch \include eingebundenen Files befinden, würde sich das auch anbieten. Eventuell als Option
--deepinputlazy die tief geschachtelte input Anweisungen wie include Anweisungen behandelt

*Schritt 1:* FUNKTION start\_parse(DOKUMENT)

Umwandeln aller Zeilenumbrüche und Escapesequenzen in Standardmakros (können in bestimmten Umbebungen [verbatim, Formeln, …] am Ende wieder zurückverwandelt werden)

\{ \} \$ \& \# \% \\_ (\@ siehe unten!!) \@@vallbrc \@@valrbrc \@@valdlr \@@valand \@@valhash \@@valpct \textunderscore

% …\n \ \n Backslash + Leerzeichen + \n

 ALLE FOLGENDEN MAKROS IMMER GEFOLGT VON EINEM LEERZEICHEN, außer sie haben ein Argument:

\n mehr als zwei Mal \par \n …

\n \@@ valnl{FILEPATH,zeilennr}
oder: \n einfach löschen bzw. durch Leerzeichen ersetzen

\\\*[LÄNGE] \@@vallb[LÄNGE] \* u. LÄNGE optional

**Wahrscheinlich sollten diese Dinge schon zu Beginn gemacht werden:**

"a "o "u "A "O "U "s nicht aber \”o \oe usw. ä ö ü Ä Ö Ü ß

,, `` ´´ ?´! usw. sensitiv! Die entsprechenden Unicode Zeichen

Auch einzelne Anführungszeichen sensitiv! Unicode

SCHRITT 4: Aufräumen Idiosynkratische Valep Makros werden verarbeitet (Zeilenumbrüche, Sonderzeichen usw.)

SCHRITT 5: Rendering Das DOM Objekt wird als Json String abgespeichert und/oder in ein oder mehrere andere Formate gerendert

ENDE valeptex