PLD2flex v1.3 – USER'S MANUAL

Helena Wedig, Felix Theodor, Joshua Wieler, Eva Belke

April 17, 2024

In case of problems or questions, consult us with the help of the git repository:
https://github.com/FelixTheodor/PLD2flex

Table of Contents

PLD2flex – USER'S MANUAL

This manual introduces PLD2flex, a tool for establishing the phonological Levenshtein distance between phonological word forms. This can apply to pairs of word forms, word forms that are compared to lists of other words, including data bases like the SUBTLEX databases (Brysbaert et al., 2011; Keeulers et al., 2010) or CELEX (Baayen et al., 1995), and any other combination of word forms. PLD2flex takes written (orthographic) input that is transcribed to a phonological form by means of the Web Services of the Bavarian Archive of Speech Signals (BAS) (Reichel & Kisler, 2014) and computes PLD scores on the basis of these. By implication, PLD2flex can be used with any language that is supported by the BAS or for that phonological transcripts are provided.

In psycholinguistic research, there are many contexts in which PLD measures are useful, for instance, when phonological errors and the corresponding target utterances are to be compared or when when the phonological neighbourhood density of stimulus material needs to be calculated. One measure of phonological neighbourhood density is PLD20 (e. g. Suárez et al., 2009), which is the average phonological Levenshtein distance of the 20 words in a corpus-based database of words that are most similar to a target, be that another word or a pseudoword.

1. System Overview

PLD2flex was developed by a group of computational linguists and psycholinguists based or formerly based at Ruhr-University Bochum. It is freely accessible to any scientist in need of a tool to calculate or use the phonological Levenshtein distance between sets of words or the PLD20 of a word when assessed against a database.

PLD2flex can compute Levenshtein distances in one-to-one and multiple-one-to-one, one-to-many, and many-to-many comparisons of word forms. The input can be

- a list of words or a file containing a list of words to be compared to a target word,
- a list of words or a file containing a list of words to be compared to a data base of words, e. g. SUBTLEX-DE (Brysbaert et al., 2011).
- a file containing two columns with individual pairs of words per line that are compared pairwise, or
- a file that contains a list of words and their phonological Levenshtein distance related to the target word (as corpus for comparison).

Apart from processing files the graphical user interface enables the user to type in the target and/or the lists manually.

PLD2flex can be used with a database of word forms of the user's choice, such as CELEX (Baayen et al., 1995) or the SUBTLEX databases, which are available freely (Keeulers et al., 2010). Other databases can be loaded to and processed by the system, provided that the BAS provides the conversion for the given language (for details, see sections 2 and 3). By default, PLD2flex incorporates the SUBTLEX-DE database (Brysbaert et al., 2011); its entries were converted into phonemes via the G2P functionality at the BAS (Reichel & Kisler, 2014).

Please note that this manual documents **v1.3 of PLD2flex**, which includes a feature that is not yet reported on in the paper on PLD2flex (Wedig et al., 2024). In v1.3 we added a means for users to decide between a default computation of Levenshtein distances based on the number of phonetic signs in the strings to be compared and one that is based on the presegmented phoneme representation provided by the BAS (Reichel & Kisler, 2014). For more details, see the introductory section of Chapter 3.

This manual will introduce the requirements for using PLD2flex offline and the main functions. The *Getting Started* section explains in brief how to install PLD2flex, how to start it and which requirements are needed to use it. The following section describes how to use the system and presents examples. A section on reporting explains possible error messages and specifies how protocol and result files can be exported.

2. Getting Started

This section will provide a general walk-through of the system from system configuration through exit. Section 2.1 describes which modules need to be installed and how to download them to the local system. The next subsection explains how to start PLD2flex and how to use the system menu. The following subsection will introduce the main functions and the handling of the graphical user interface. Finally, the user will get to know how to exit the system.

2.1 System Installation

The easiest way to install the tool is to download the file matching your OS from the releases page: https://github.com/FelixTheodor/PLD2flex/release. It is a .zip file that you can unpack anywhere on your system. The archive contains the binary for execution (on Windows, .exe) as well as the folder structure needed to run the tool. Once all the files are unpacked, you can run the tool by clicking on the PLD2flex file in the unpacked folder.

If you want to run the tool from the source code, you can follow these steps:

The installation steps are as follows:

1 Install python3.12: https://www.python.org/downloads/

2 Install (ana)conda: https://conda.io/projects/conda/en/latest/index.html

3 Clone this repository, open a new terminal in the main directory of it

(recommended) If you want to use (ana)conda:

-3.1 Create a new env with this command:

```
conda create -n PLD2flex python=3.12
```

-3.2 Activate the new environment:

```
conda activate PLD2flex
```

-3.3 Install pip in your new environment:

```
conda install pip
```

4 Install all required packages via pip: `pip install -r requirements.txt`

5 Should the installation of a package fail, try to install the missing package manually via pip:

```
pip install missing_package
```

6 Update the config.txt (see section 2.2)
Among other updates, you need to specify the paths for the result / database files. Make sure that you use the right path format for your operating system.

7. Run the main.py: `python main.py`

2.2 System Configuration

The main folder of the tool contains a file called config.txt. In this file, users can configure settings for PLD2flex by putting the name of the setting on the left side of "=" (no spaces!) and a valid value for that setting on the right side. These are the parameters to be configured:

DBFOLDER: put here the path to the folder where your database files are.

Default: data/corpora

LANGUAGE: put here the language of your input data in the BAS-format[1].

Default: deu-DE

NEIGHBOURS: put here the number of neighbours you want to calculate with.

Default: 20

RESULTS_PATH: put here the path to the folder where you want your result data to be saved in.

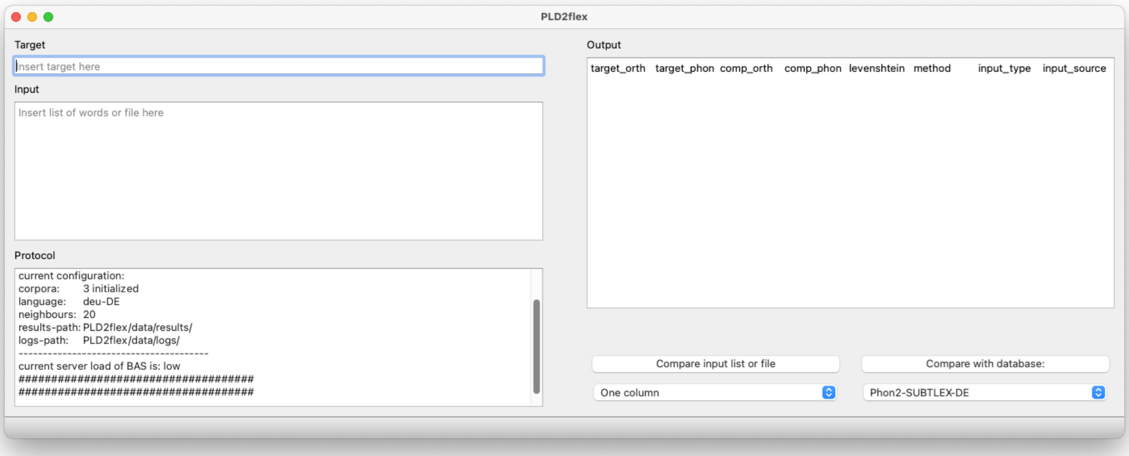
Default: data/results

LOGS_PATH: put here the path to the folder where you want your log data to be saved in.

Default: data/logs

2.3 Logging On + System Menu

To start PLD2flex the systems directory has to be changed via the command line to the location of the file 'PLD2flex/main.py' (`cd` *`path`*`/PLD2flex`). PLD2flex then starts by using the command `python3 main.py`[2] or `python main.py`. After starting PLD2flex, a window opens as shown in Figure 1.



```
target_orth   target_phon  comp_orth   comp_phon  levenshtein  method      input_type   input_source
```

```
current configuration:
corpora:      3 initialized
language:     deu-DE
neighbours:   20
results-path: PLD2flex/data/results/
logs-path:    PLD2flex/data/logs/
----------------------------------------
current server load of BAS is: low
#####################################
#####################################
```

One column

Phon2-SUBTLEX-DE

Figure 1. Main window of the program.

[1] for a full list of the available languages, see here: https://clarin.phonetik.uni-muenchen.de/BASWebServices/services/help

[2] in case python 2 is also installed on the system, e.g. on MacOS or Linux

2.4 System Functions

This subsection will describe the functions and attributes of the graphical user interface. Details on how to use each function will be introduced in section 3 ("Using the System").

2.4.1 Target

As shown in Figures 1 and 2, the GUI contains a text field named *Target* in the upper left corner of the window. This is where the target word or any other sequence of alphabetic characters has to be specified if a single target is to be compared with other words or pseudowords. Depending on the setting specified by the user, it will be compared to the forms specified in the Input section or to the words specified in a database. Users wishing to compute the PLD20 of a word will have to specify the target sequence here and compare it with one of the databases in the corpora folder, as configured in the config.txt-file (see section 2.2).

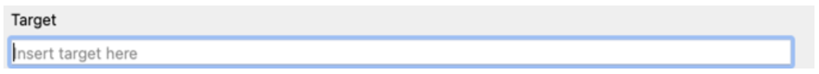


Figure 2. Text field to type in target word.

2.4.2 Input

The text field beneath the "Target"-field on the mid left side of the window is named "Input" (Figures 1 and 3). Here, data can be inserted by either typing in a list of words or by dragging and dropping a file. When using a list of words to be compared with a Target, there must not be more than one word or character sequence per line. The data provided in the Input section will then be compared to the target and the Levenshtein distance(s) will be established. For all other comparison scenarios, we will discuss the required format of files to be loaded in the Input field in section 3.

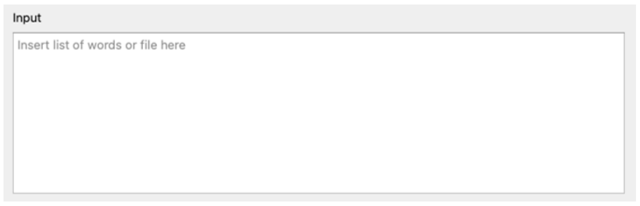
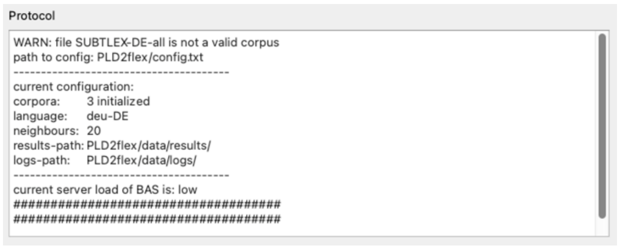


Figure 3. Text field to drag and drop a file or to type in list of words

2.4.3 Protocol

The text field "Protocol" is located in the lower left section of the window (Figure 4). It is not possible to drop files or write text in here, as this section is used for the protocol only. This field lists the input of the user and the steps of the ongoing process. Likewise, error messages are displayed in this text field, for instance, if the used data structure is not appropriate. A detailed list of the possible error messages can be found in section 4.1.

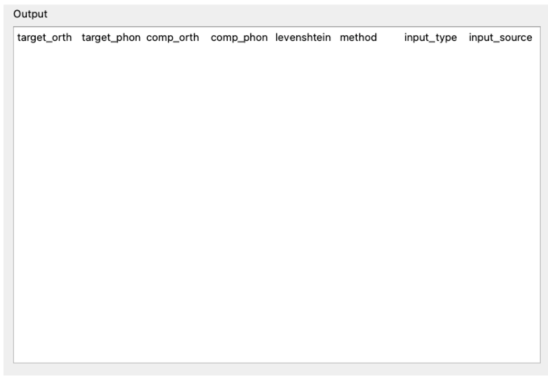


```
Protocol
WARN: file SUBTLEX-DE-all is not a valid corpus
path to config: PLD2flex/config.txt
---------------------------------------
current configuration:
corpora:      3 initialized
language:     deu-DE
neighbours:   20
results-path: PLD2flex/data/results/
logs-path:    PLD2flex/data/logs/
---------------------------------------
current server load of BAS is: low
####################################
####################################
```

Figure 4. Text field showing the protocol and logs for the init process.
The warning about SUBTLEX-DE in this screenshot highlights that this corpus does not contain a database in a format readable for the tool (see section 4.1).

2.4.4 Output

After calculating the Levenshtein distance, the results can be seen on the upper right side of the window (Figure 5). It is not possible to drop files or write text in here, as this section is only used for displaying the output. For details about the structure of the output, see the relevant subsections of Section 3 ("Using the System").



```
Output
target_orth  target_phon  comp_orth  comp_phon  levenshtein  method  input_type  input_source
```

Figure 5. Text field containing the results

2.4.5 Using the Buttons

To use PLD2flex and start the process of establishing PLDs, clicking on one of the two buttons on the right hand side of the window is necessary (Figure 6). If files were specified as the input, the user can either choose "File (two columns)" or "File (one column)" depending on the given format. Beneath the “Compare to database” button, the user can choose one of the corpora listed below. They stem from the corpora folder specified in the config.txt.

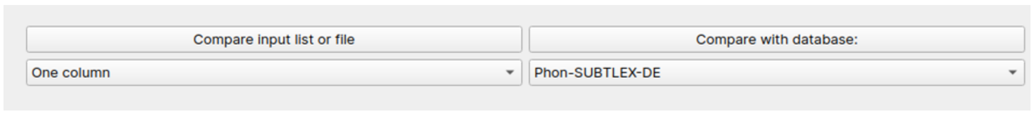


Figure 6. Buttons to start the process

2.5 Exit System

To exit the system, just close the window.

3. Using the System

This section provides a detailed description of the functions in PLD2flex. Each subsection will introduce the required input, the use and the resulting output. In general, all inputs need to be encoded as "utf-8", the output will be encoded likewise.

To transcribe the input and the target word, PLD2flex uses the web application G2P (Reichel & Kisler, 2014), provided by the Bavarian Archive of Speech Signals.[3] The output will be in SAMPA format which is an ASCII-based phonetic alphabet (Reichel & Kisler, 2014).

Having originally been equipped with only one standard character-based function for comparing phonetically transcribed (pseudo)word forms, we have recently augmented PLD2flex by a phoneme-based computation of phonological Levenshtein distances that is based on the phonological pre-segmentation provided by the BAS. For users to be able to set the method that best suits their needs, we added a new option called **CALC** in the config.txt, which can be set to **CHARACTER** or **SPACES**, respectively. In the CHARACTER method, all spaces delimiting individual phonemes are being removed and the resulting strings of SAMPA signs are compared. With this method, all elements of the strings are handled as individual segments, so the distance between *bit* [b I t] and *beet* [b i: t] is 2, whereas the distance between *bit* [b I t] and *bate* [b eI t] is 1. The SPACES makes use of the phonological pre-segmentation provided by the BAS and treats each element that is preceded and/or followed by a space as a single segment, regardless of whether it is represented by a single or multiple characters. With this calculation, the distances between [b I t], [b i: t], and [b eI t] are all 1.

3.1 Compare target and list of words

One of the most important functions of PLD2flex is the option to compare a target word with a manually given list of words. For such a comparison, the target word and a list of words have to be typed in as shown on the left side of Figure 7. It is required to type in one word per line. By default, a list of 20 words is expected, but the system also enables users to process more or less than 20 words (the default number of words can be set in the config.txt-file, see section 2.2). To prevent the user from specifying unintentionally either more or less than 20 words, the protocol notifies the user when more or less than 20 words will be tested.

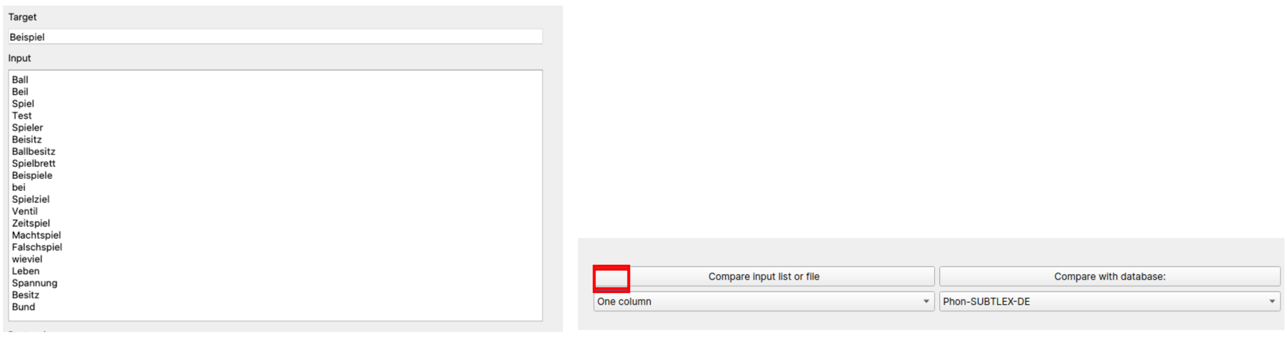
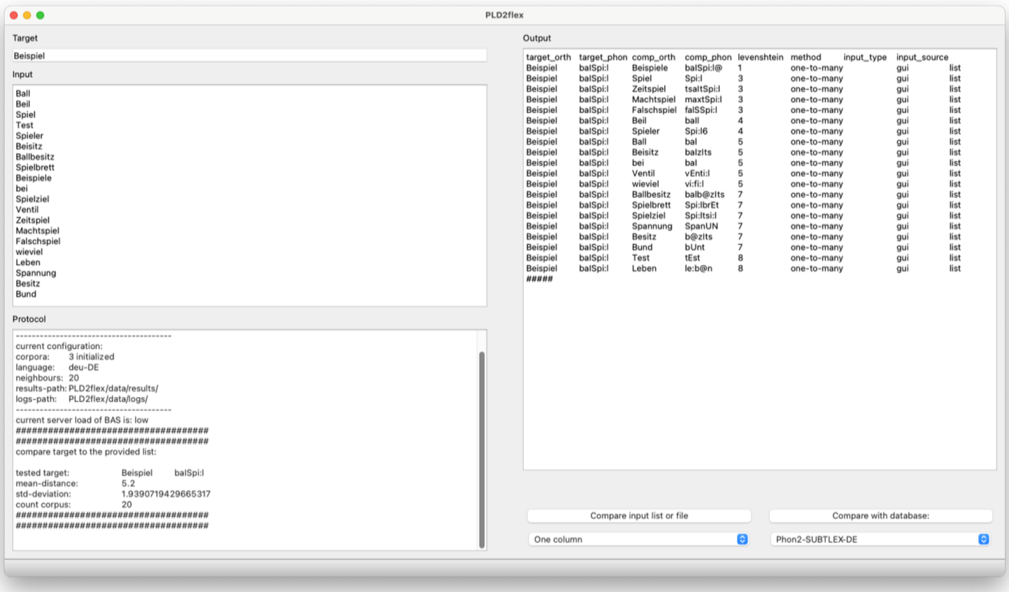

Target
Beispiel
Input
Ball
Beil
Spiel
Test
Spieler
Beisitz
Ballbesitz
Spielbrett
Beispiele
bei
Spielziel
Ventil
Zeitspiel
Machtspiel
Falschspiel
wieviel
Leben
Spannung
Besitz
Bund

Figure 7. Example showing how to type in target word and list of words on the left side. The right side shows the buttons to start PLD2flex.

When clicking on the button "Compare input with list or file", PLD2flex will first request the transcript of the given words via G2P. After that it will calculate the phonological Levenshtein distance between each word and the target. The results will be printed in the text field "Output", as shown on the left in Figure 8.

[3] see https://clarin.phonetik.uni-muenchen.de/BASWebServices/interface/Grapheme2Phoneme



```
Beispiel
```

```
Ball
Beil
Spiel
Test
Spieler
Beisitz
Ballbesitz
Spielbrett
Beispiele
bei
Spielziel
Ventil
Zeitspiel
Machtspiel
Falschspiel
wieviel
Leben
Spannung
Besitz
Bund
```

```
---------------------------------------
current configuration:
corpora:      3 initialized
language:     deu-DE
neighbours:   20
results-path: PLD2flex/data/results/
logs-path:    PLD2flex/data/logs/
---------------------------------------
current server load of BAS is: low
#####################################
#####################################
compare target to the provided list:

tested target:          Beispiel     balSpi:l
mean-distance:          5.2
std-deviation:          1.9390719429665317
count corpus:           20
#####################################
#####################################
```

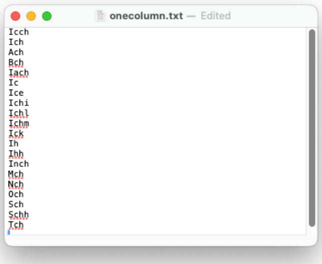
```
target_orth  target_phon  comp_orth    comp_phon   levenshtein  method       input_type  input_source
Beispiel     balSpi:l     Beispiele    balSpi:l@   1            one-to-many               gui         list
Beispiel     balSpi:l     Spiel        Spi:l       3            one-to-many               gui         list
Beispiel     balSpi:l     Zeitspiel    tsaltSpi:l  3            one-to-many               gui         list
Beispiel     balSpi:l     Machtspiel   maxtSpi:l   3            one-to-many               gui         list
Beispiel     balSpi:l     Falschspiel  falSSpi:l   3            one-to-many               gui         list
Beispiel     balSpi:l     Beil         ball        4            one-to-many               gui         list
Beispiel     balSpi:l     Spieler      Spi:l6      4            one-to-many               gui         list
Beispiel     balSpi:l     Ball         bal         5            one-to-many               gui         list
Beispiel     balSpi:l     Beisitz      balzIts     5            one-to-many               gui         list
Beispiel     balSpi:l     bei          bal         5            one-to-many               gui         list
Beispiel     balSpi:l     Ventil       vEnti:l     5            one-to-many               gui         list
Beispiel     balSpi:l     wieviel      vi:fi:l     5            one-to-many               gui         list
Beispiel     balSpi:l     Ballbesitz   balb@zIts   7            one-to-many               gui         list
Beispiel     balSpi:l     Spielbrett   Spi:lbrEt   7            one-to-many               gui         list
Beispiel     balSpi:l     Spielziel    Spi:ltsi:l  7            one-to-many               gui         list
Beispiel     balSpi:l     Spannung     SpanUN      7            one-to-many               gui         list
Beispiel     balSpi:l     Besitz       b@zIts      7            one-to-many               gui         list
Beispiel     balSpi:l     Bund         bUnt        7            one-to-many               gui         list
Beispiel     balSpi:l     Test         tEst        8            one-to-many               gui         list
Beispiel     balSpi:l     Leben        le:b@n      8            one-to-many               gui         list
#####
```

Figure 8. Example showing the output on the right side and the protocol on the left side.

In addition, the mean and standard deviation of the PLD computed across the Levenshtein distances of all pairings listed in the output are written to the field "Protocol", as shown on the right in Figure 8. All results are also saved in two different files, result and log, so they will not be lost when PLD2flex is closed.

3.2 Compare target with a one-column file

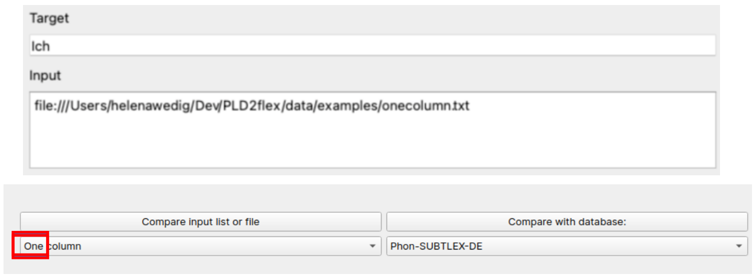
To compare the target word with a list of words, that is not given in the text field "Input", but in a text file, it is possible to drop the text file into the text field "Input". Therefore, the text file has to be encoded as "utf-8" and it has to have one word per line (as seen in Figure 9), which shows the contents of the file onecolumn.txt, provided by way of an example. (Note that this example file was created to demonstrate the performance of G2P with pseudowords, i. e. nonwords that adhere to the phonotactic principles of the target language, such as *Ichi* in German, and other nonwords, that do not do so, such as *Bch*. We will comment on its performance when discussing Figure 11, which shows G2P's output for the list of character sequences in the file.)



```
Icch
Ich
Ach
Bch
Iach
Ic
Ice
Ichi
Ichl
Ichm
Ick
Ih
Ihh
Inch
Mch
Nch
Och
Sch
Schh
Tch
```

Figure 9. Possible input file with one column.

After dropping the file there, its path will appear in the text field. As described in chapter 3.1 the target word has to be placed in the text field "Target". To read the file it is necessary to choose "One column" and then click on "Compare with file/list" (Figure 10).



Target

Ich

Input

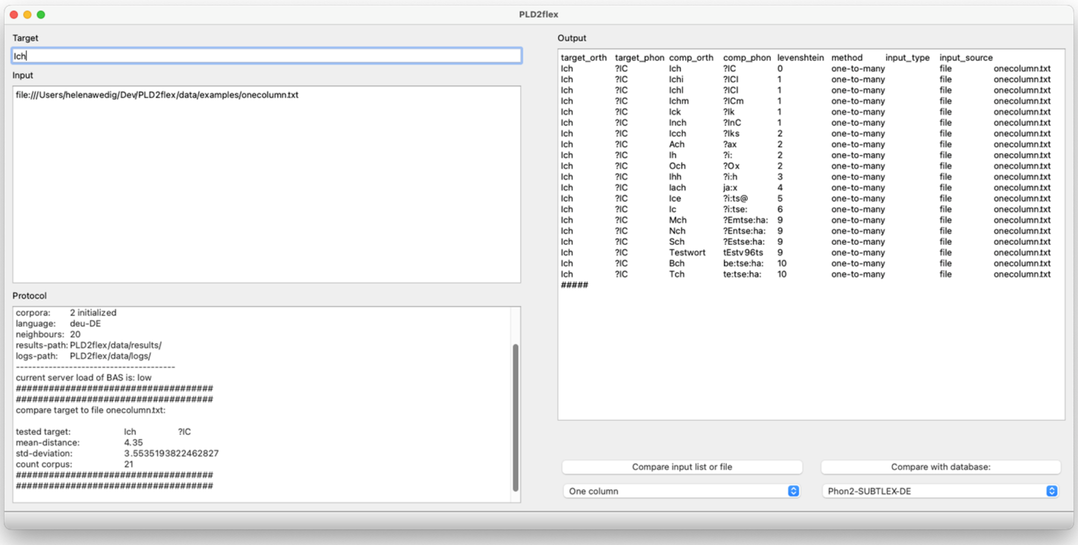
file:///Users/helenawedig/Dev/PLD2flex/data/examples/onecolumn.txt

Compare input list or file: One column

Compare with database: Phon-SUBTLEX-DE

Figure 10. Examples showing the input in the upper part and the settings in the lower part.

After processing the file and calculating the phonological Levenshtein distances of all pairings, PLD2flex prints the result into the text field "Output", into the field "Protocol", and into the corresponding text file (Figure 11).



Target

Ich

Input

file:///Users/helenawedig/Dev/PLD2flex/data/examples/onecolumn.txt

Protocol

```
corpora:      2 initialized
language:     deu-DE
neighbours:   20
results-path: PLD2flex/data/results/
logs-path:    PLD2flex/data/logs/
--------------------------------------
current server load of BAS is: low
####################################
####################################
compare target to file onecolumn.txt:

tested target:          Ich        ?IC
mean-distance:          4.35
std-deviation:          3.5535193822462827
count corpus:           21
####################################
####################################
```

Output

```
target_orth  target_phon  comp_orth  comp_phon   levenshtein  method       input_type  input_source
Ich          ?IC          Ich        ?IC         0            one-to-many  file        onecolumn.txt
Ich          ?IC          Ichi       ?ICI        1            one-to-many  file        onecolumn.txt
Ich          ?IC          Ichl       ?ICl        1            one-to-many  file        onecolumn.txt
Ich          ?IC          Ichm       ?ICm        1            one-to-many  file        onecolumn.txt
Ich          ?IC          Ick        ?Ik         1            one-to-many  file        onecolumn.txt
Ich          ?IC          Inch       ?InC        1            one-to-many  file        onecolumn.txt
Ich          ?IC          Icch       ?Iks        2            one-to-many  file        onecolumn.txt
Ich          ?IC          Ach        ?ax         2            one-to-many  file        onecolumn.txt
Ich          ?IC          Ih         ?i:         2            one-to-many  file        onecolumn.txt
Ich          ?IC          Och        ?Ox         2            one-to-many  file        onecolumn.txt
Ich          ?IC          Ihh        ?i:h        3            one-to-many  file        onecolumn.txt
Ich          ?IC          Iach       ja:x        4            one-to-many  file        onecolumn.txt
Ich          ?IC          Ice        ?i:ts@      5            one-to-many  file        onecolumn.txt
Ich          ?IC          Ic         ?i:tse:     6            one-to-many  file        onecolumn.txt
Ich          ?IC          Mch        ?Emtse:ha:  9            one-to-many  file        onecolumn.txt
Ich          ?IC          Nch        ?Entse:ha:  9            one-to-many  file        onecolumn.txt
Ich          ?IC          Sch        ?Estse:ha:  9            one-to-many  file        onecolumn.txt
Ich          ?IC          Testwort   tEstv96ts   9            one-to-many  file        onecolumn.txt
Ich          ?IC          Bch        be:tse:ha:  10           one-to-many  file        onecolumn.txt
Ich          ?IC          Tch        te:tse:ha:  10           one-to-many  file        onecolumn.txt
#####
```

Compare input list or file: One column

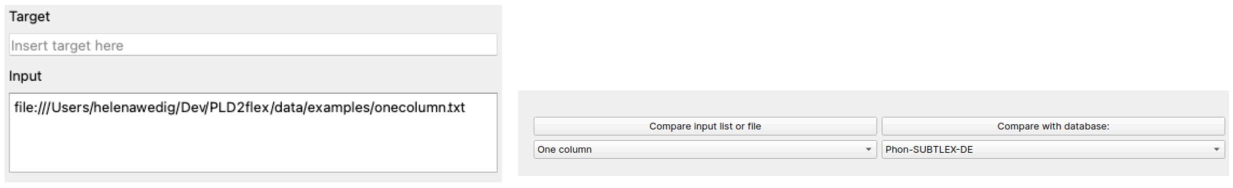
Compare with database: Phon2-SUBTLEX-DE

Figure 11. Example showing the output on the left side and the protocol on the right side.

Figure 11 gives an impression of how G2P deals with character sequences that cannot be pronounced with the G2P-rules it is equipped with: It simply spells out the letter names, as in 'Bch', transcribed as `be:tse:ha:` in German. We use this example to highlight this for users of PLD2flex who work with pseudowords. Such orthographically implausible sequences of letter names yield considerably longer transcripts, which has an immediate effect on the Levenshtein distances that are computed. It is hence paramount that users of PLD2flex check the output of G2P and indeed PLD2flex carefully before making use of the results.

3.3 Cross compare a one-column file

If given a one-column file with a list of words and no target word, PLD2flex will calculate the Levenshtein-distance for each possible pair of words in the list. Again, the user has to drop the file in the text field "Input", choose "One column" and then click on "Compare input list or file" (Figure 12). The following examples use the same text file as shown in Figure 9.



Target

Input

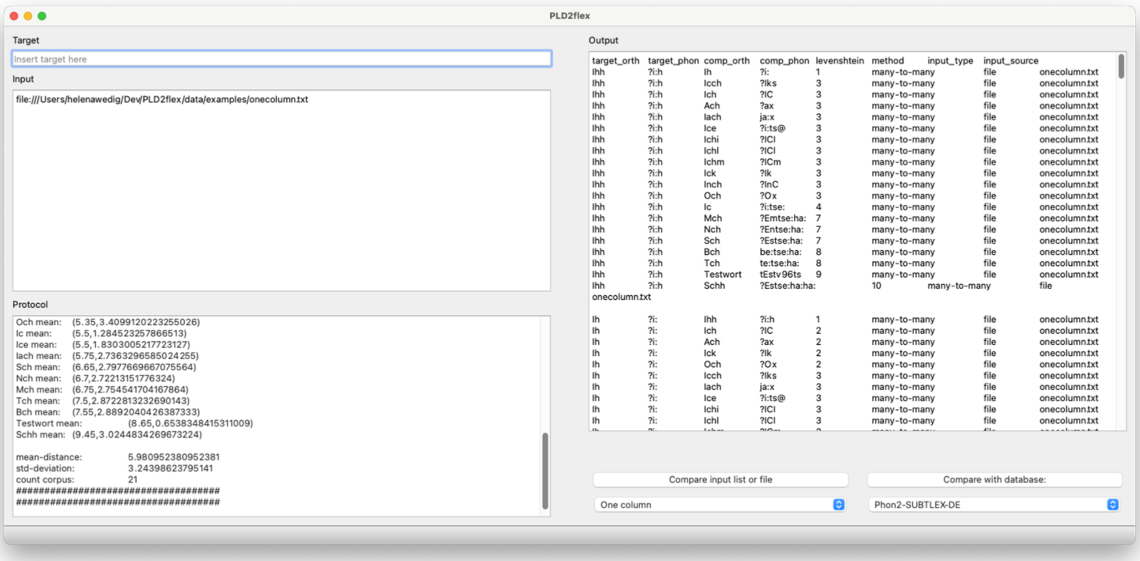
```
file:///Users/helenawedig/Dev/PLD2flex/data/examples/onecolumn.txt
```

One column

Phon-SUBTLEX-DE

Figure 12. Examples showing the input on the left and the settings on the right side.

Finishing the calculation, PLD2flex will present the result in the text fields "Output" and "Protocol" as well as in the corresponding files. Because cross-comparing would lead to double entries in the output as of the second entry, PLD2flex ignores pairs that have already been compared (Figure 13).



Target

Input

```
file:///Users/helenawedig/Dev/PLD2flex/data/examples/onecolumn.txt
```

Protocol

```
Och mean:   (5.35,3.4099120223255026)
Ic mean:    (5.5,1.284523257866513)
Ice mean:   (5.5,1.8303005217723127)
Iach mean:  (5.75,2.7363296585024255)
Sch mean:   (6.65,2.7977669667075564)
Nch mean:   (6.7,2.72213151776324)
Mch mean:   (6.75,2.754541704167864)
Tch mean:   (7.5,2.8722813232690143)
Bch mean:   (7.55,2.8892040426387333)
Testwort mean:          (8.65,0.6538348415311009)
Schh mean:  (9.45,3.0244834269673224)

mean-distance:          5.980952380952381
std-deviation:          3.24398623795141
count corpus:           21
######################################
######################################
```

Output

```
target_orth  target_phon  comp_orth  comp_phon  levenshtein  method  input_type  input_source
Ihh          ?i:h         Ih         ?i:         1           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Icch       ?Iks        3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ich        ?IC         3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ach        ?ax         3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Iach       ja:x        3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ice        ?i:ts@      3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ichi       ?ICI        3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ichl       ?ICl        3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ichm       ?ICm        3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ick        ?Ik         3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Inch       ?InC        3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Och        ?Ox         3           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Ic         ?i:tse:     4           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Mch        ?Emtse:ha:  7           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Nch        ?Entse:ha:  7           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Sch        ?Estse:ha:  7           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Bch        be:tse:ha:  8           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Tch        te:tse:ha:  8           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Testwort   tEstv96ts   9           many-to-many        file        onecolumn.txt
Ihh          ?i:h         Schh       ?Estse:ha:ha:           10          many-to-many        file
onecolumn.txt

Ih           ?i:          Ihh        ?i:h        1           many-to-many        file        onecolumn.txt
Ih           ?i:          Ich        ?IC         2           many-to-many        file        onecolumn.txt
Ih           ?i:          Ach        ?ax         2           many-to-many        file        onecolumn.txt
Ih           ?i:          Ick        ?Ik         2           many-to-many        file        onecolumn.txt
Ih           ?i:          Och        ?Ox         2           many-to-many        file        onecolumn.txt
Ih           ?i:          Icch       ?Iks        3           many-to-many        file        onecolumn.txt
Ih           ?i:          Iach       ja:x        3           many-to-many        file        onecolumn.txt
Ih           ?i:          Ice        ?i:ts@      3           many-to-many        file        onecolumn.txt
Ih           ?i:          Ichi       ?ICI        3           many-to-many        file        onecolumn.txt
Ih           ?i:          Ichl       ?ICl        3           many-to-many        file        onecolumn.txt
```

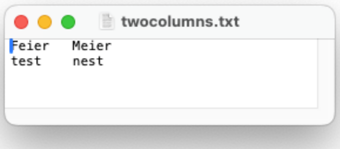
One column

Phon2-SUBTLEX-DE

Figure 13. Example showing the output on the left side and the protocol on the right side.

3.4 Compare target within a two-column file

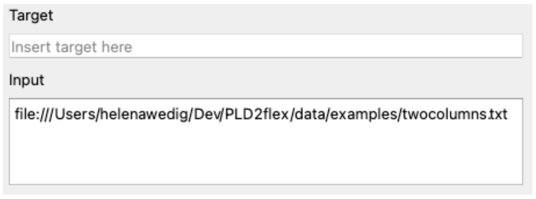
PLD2flex enables users to compare the Levenshtein distance of pairs of words specified in a two-column file. The first column has to include the target word, the second column the corresponding input (Figure 14). Processing the "utf-8"-encoded file, the result will be the Levenshtein distance between the token in the first and the token in the second column. Columns need to be separated either by a space or a tab.



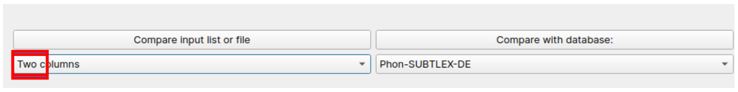
```
Feier   Meier
test    nest
```

Figure 14. Possible input file with two columns.

To start the process, it is required to drag and drop the file specifying the input in the text field "Input" and to choose the mode "Two Columns" (Figure 15). It is not necessary to put a target word into the text field "Target".



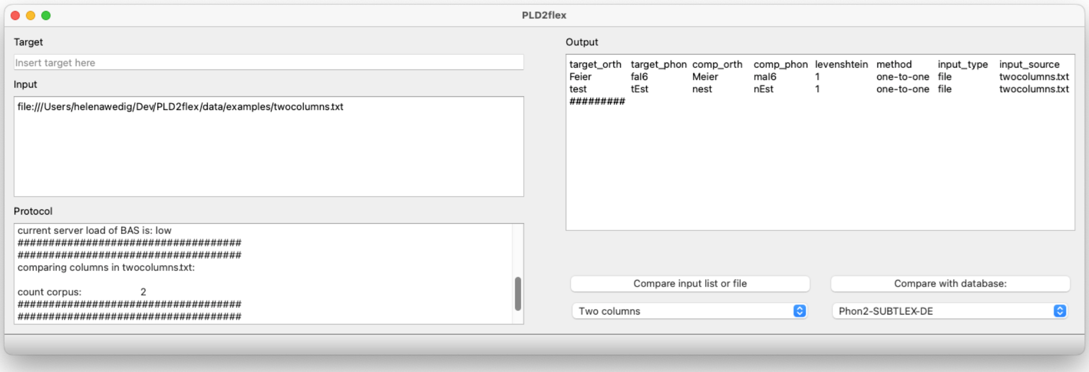
```
file:///Users/helenawedig/Dev/PLD2flex/data/examples/twocolumns.txt
```



Two columns | Phon-SUBTLEX-DE

Figure 15. Examples showing the input on the left and the settings on the right side for carrying out pairwise comparisons.

After dropping the text file, a click on the button "Compare input list or file" will make PLD2flex calculate the Levenshtein distance; it will display the results in the text fields "Output" and "Protocol" and save it in the corresponding text files (Figure 16). Users can also manually type in or copy and paste two columns of words in the input field to get the results. Please be aware that the pairwise Levenshtein computation is currently not optimized for bigger data sets; therefore, the process can take a long time to finish.



```
file:///Users/helenawedig/Dev/PLD2flex/data/examples/twocolumns.txt
```

```
current server load of BAS is: low
#####################################
#####################################
comparing columns in twocolumns.txt:

count corpus:             2
#####################################
#####################################
```

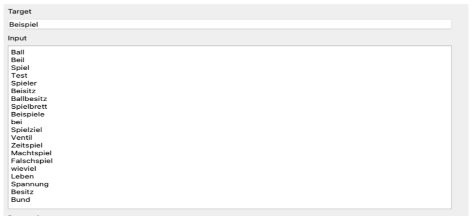
```
target_orth  target_phon  comp_orth  comp_phon  levenshtein  method      input_type  input_source
Feier        fal6         Meier      mal6       1            one-to-one  file        twocolumns.txt
test         tEst         nest       nEst       1            one-to-one  file        twocolumns.txt
#########
```

Two columns | Phon2-SUBTLEX-DE

Figure 16. Example showing the output on the left side and the protocol on the right side.

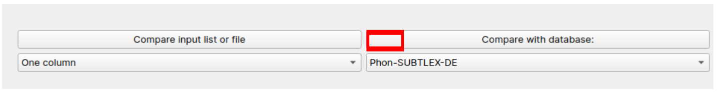
3.5 Compare target with SUBTLEX-DE

Another main function of PLD2flex is the ability to search for the 20 nearest neighbours in a given database. The mean Levenshtein distance computed over this set is a word's PLD20, as used by Suárez et al. (2011), for example. PLD2flex includes a G2P-transcribed version of the SUBTLEX-DE database (Brysbaert et al., 2011), stored in the file Phon-SUBTLEX-DE.txt (for preparing other databases in a similar fashion, see section 3.6). PLD2flex compares the transcribed target word with the G2P-pre-processed SUBTLEX-DE database and provides the 20 entries with the lowest Levenshtein distances to the target. To use this function, it is only required to type in a target word, choose SUBTLEX-DE as a database, and click on the button "Compare with database " (Figure 17).



Beispiel

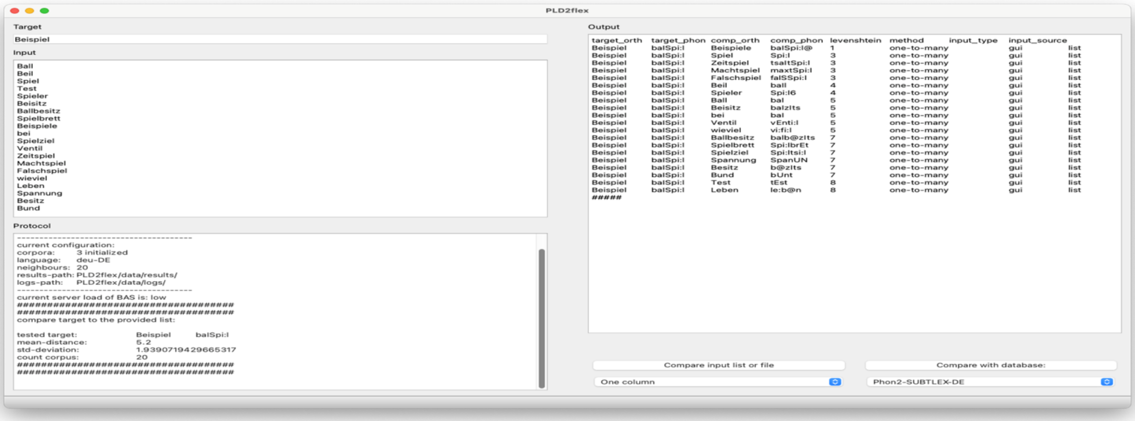
```
Ball
Beil
Spiel
Test
Spieler
Beisitz
Ballbesitz
Spielbrett
Beispiele
bei
Spielziel
Ventil
Zeitspiel
Machtspiel
Falschspiel
wieviel
Leben
Spannung
Besitz
Bund
```



One column | Phon-SUBTLEX-DE

Figure 17. Examples showing the input on the left and the settings on the right side.

After finding the 20 lowest Levenshtein distances, PLD2flex will create a text file that contains the 20 neighbours and their Levenshtein distances to the target word. Furthermore, the mean of the distances and the individual results will be printed into the text fields "Output" and "Protocol" as well as into the corresponding text file (Figure 18). We recommend inspecting the output for spurious entries and for deciding on how to deal with them. For instance, Figure 18 shows a SUBTLEX-DE entry called *fire*, which is homophonous to the German target word *Feier*. Given that the SUBTLEX-DE database was generated based on movie subtitles, it comes as no surprise that it includes many English words, so it is important to decide on how to best deal with them (see also section 5 on finding and dealing with homographs and homophones in a database).



Target

Beispiel

Input

```
Ball
Beil
Spiel
Test
Spieler
Beisitz
Ballbesitz
Spielbrett
Beispiele
bei
Spielziel
Ventil
Zeitspiel
Machtspiel
Falschspiel
wieviel
Leben
Spannung
Besitz
Bund
```

Protocol

```
----------------------------------------
current configuration:
corpora:       3 initialized
language:      deu-DE
neighbours:    20
results-path: PLD2flex/data/results/
logs-path:     PLD2flex/data/logs/
----------------------------------------
current server load of BAS is: low
##########################################
##########################################
compare target to the provided list:

tested target:               Beispiel        balSpi:l
mean-distance:               5.2
std-deviation:               1.939071942966531 7
count corpus:                20
##########################################
##########################################
```

Output

```
target_orth  target_phon  comp_orth    comp_phon   levenshtein  method       input_type  input_source
Beispiel     balSpi:l     Beispiele    balSpi:l@   1            one-to-many              gui          list
Beispiel     balSpi:l     Spiel        Spi:l       3            one-to-many              gui          list
Beispiel     balSpi:l     Zeitspiel    tsaltSpi:l  3            one-to-many              gui          list
Beispiel     balSpi:l     Machtspiel   maxtSpi:l   3            one-to-many              gui          list
Beispiel     balSpi:l     Falschspiel  falSSpi:l   3            one-to-many              gui          list
Beispiel     balSpi:l     Beil         ball        4            one-to-many              gui          list
Beispiel     balSpi:l     Spieler      Spi:l6      4            one-to-many              gui          list
Beispiel     balSpi:l     Ball         bal         5            one-to-many              gui          list
Beispiel     balSpi:l     Beisitz      balzIts     5            one-to-many              gui          list
Beispiel     balSpi:l     bei          bal         5            one-to-many              gui          list
Beispiel     balSpi:l     Ventil       vEnti:l     5            one-to-many              gui          list
Beispiel     balSpi:l     wieviel      vi:fi:l     5            one-to-many              gui          list
Beispiel     balSpi:l     Ballbesitz   balb@zIts   7            one-to-many              gui          list
Beispiel     balSpi:l     Spielbrett   Spi:lbrEt   7            one-to-many              gui          list
Beispiel     balSpi:l     Spielziel    Spi:ltsi:l  7            one-to-many              gui          list
Beispiel     balSpi:l     Spannung     SpanUN      7            one-to-many              gui          list
Beispiel     balSpi:l     Besitz       b@zIts      7            one-to-many              gui          list
Beispiel     balSpi:l     Bund         bUnt        7            one-to-many              gui          list
Beispiel     balSpi:l     Test         tEst        8            one-to-many              gui          list
Beispiel     balSpi:l     Leben        le:b@n      8            one-to-many              gui          list
######
```

Figure 18. Example showing the output on the left side and the protocol on the right side.

3.6 Compare target with a database of individual choice

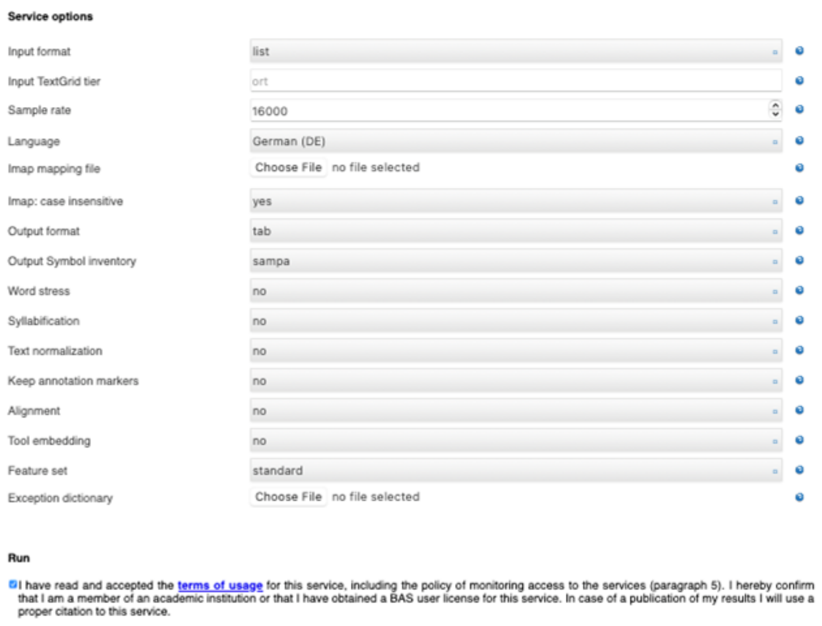
As stated in the previous subsection, it is also possible to compare a target word with a database of choice. To use this function PLD2flex needs to be provided with a transcribed database. Every database can be transcribed automatically via G2P as explained in subsection 3.8. It can then be placed in the corpus directory configured in the config.txt. This way, the database will be processed on start-up of PLD2flex and users can choose it as an alternative to SUBTLEX in the textbox. After typing in the target word in the text field "Target" and clicking on the button "Compare with database", PLD2flex will calculate the 20 nearest Levenshtein distances and print the result into the text fields "Output" and "Protocol" in addition to storing the corresponding text file.

3.7 Compare list of targets with a database

It is also possible to compare a list of targets one by one to a database derived from a corpus. To this end, users need to put the file or list of targets into the input fields as described above and click on “Compare to database”. Be aware that this probably leads to a lot of computation, therefore it might take some time till the process finishes.

3.8 Excursus: Modifying a database

To use any database, it is necessary to adjust its format for use with PLD2flex. Per line, the database file must include one word, a semicolon and the word's SAMPA version. To produce such a file, it has to be transcribed by G2P (see footnote 3). G2P requires a file that has a maximum of 100.000 tokens and one token per line. It can be used via a web application. Figure 19 shows the options that need to be chosen:



Service options

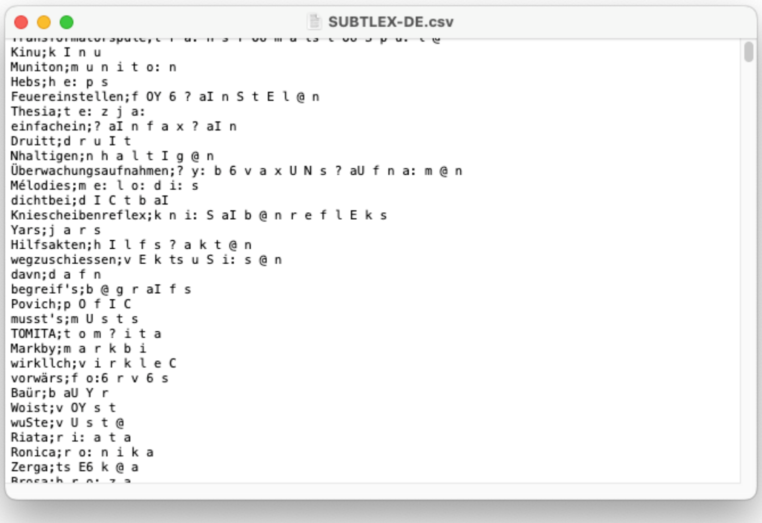
Input format	list
Input TextGrid tier	ort
Sample rate	16000
Language	German (DE)
Imap mapping file	Choose File no file selected
Imap: case insensitive	yes
Output format	tab
Output Symbol inventory	sampa
Word stress	no
Syllabification	no
Text normalization	no
Keep annotation markers	no
Alignment	no
Tool embedding	no
Feature set	standard
Exception dictionary	Choose File no file selected

Run

☑ I have read and accepted the terms of usage for this service, including the policy of monitoring access to the services (paragraph 5). I hereby confirm that I am a member of an academic institution or that I have obtained a BAS user license for this service. In case of a publication of my results I will use a proper citation to this service.

Figure 19. Necessary options to use G2P via the web application.

After using G2P, the file has the format shown in Figure 20 (csv with semicolon as separator) and can be used to calculate the 20 nearest Levenshtein distances.



SUBTLEX-DE.csv

```
Kinu;k I n u
Muniton;m u n i t o: n
Hebs;h e: p s
Feuereinstellen;f OY 6 ? aI n S t E l @ n
Thesia;t e: z j a:
einfachein;? aI n f a x ? aI n
Druitt;d r u I t
Nhaltigen;n h a l t I g @ n
Überwachungsaufnahmen;? y: b 6 v a x U N s ? aU f n a: m @ n
Mélodies;m e: l o: d i: s
dichtbei;d I C t b aI
Kniescheibenreflex;k n i: S aI b @ n r e f l E k s
Yars;j a r s
Hilfsakten;h I l f s ? a k t @ n
wegzuschiessen;v E k ts u S i: s @ n
davn;d a f n
begreif's;b @ g r aI f s
Povich;p O f I C
musst's;m U s t s
TOMITA;t o m ? i t a
Markby;m a r k b i
wirkllch;v i r k l e C
vorwärs;f o:6 r v 6 s
Baür;b aU Y r
Woist;v OY s t
wuSte;v U s t @
Riata;r i: a t a
Ronica;r o: n i k a
Zerga;ts E6 k @ a
Brosa;b r o: z a
```

Figure 20. Formatted database.

4. Producing the log and result-files

This section names and describes all error messages and results that can be generated by the system. Each protocol and result file will be saved in the folder "logs" or "results", respectively, but users can also configure a custom location for it in the config.txt-file. The names of the log and result files include exact timestamps.

4.1 Error messages and warnings

To ensure that PLD2flex works smoothly, several messages notify the user of possible sources of error. The following table gives an overview of them and their corresponding solution.

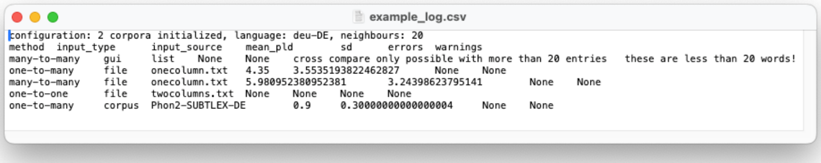
error message / warning	origin	solution
File not found, cannot read file	The file does not exist in the given path.	Check if the file exists in the given directory, adapt the path details, if necessary.
Comparing two columns is not possible with given target	The user stated a target word, but wants to compare a file with two columns.	Delete the target word.
Connection to BAS refused	Either the BAS-Server has problems, or your internet connection is not stable.	Check your internet connection and take a look at the start-up messages of the program: it tells you when the BAS-Server is overloaded
No list or file in input	User clicked “Compare to input/file”, but the input field is empty	Put something in the input field or click “Compare to database”
No database provided	There is no (working) database file in the configured corpus directory	Put a file with the right format in the configured directory (e.g. SUBTLEX-DE)
No data in target or input field, data in target and input field	Either both fields are filled and the program does not know which to choose, or none is filled.	Compare the inputs, delete the target or input, or fill in one of the fields
Comparing two columns is not possible with given target	The user checked button "File (two columns)" but a target is provided	Delete the target
Too many columns in input file, not every line has two columns	The given file or input does not have the right format regarding columns.	Beware that every tab or space in a line counts as a new column. Make sure that the number of columns is correct.
Cross compare only possible with more than x entries	The user tried to cross compare with less entries then the configured number of neighbours	Change the number of neighbours or add more entries
WARN: X is not a valid corpus	There is a file in the specified database path that does not contain a database in a format readable for the tool.	Ignore this if you do not need to use the named file as a database. If you want to use It, make sure that the format is valid (compare to examples).
RuntimeWarning: Couldn’t find ffmpeg or avconc – defaulting to ffmpeg but may not work	This is a warning pertaining to the sound that is played when PLD2flex has completed processing.	Ignore this warning and carry on using the tool albeit without the bell sound being played when calculations are finished; alternatively, use the Python based setup as explained in this Manual to implement a version that incluces the bell sound.

4.2. Printing the protocol

The error messages are a part of the log that will be produced in the process of using PLD2flex. Every time the user starts PLD2flex, a new log file will be produced in the configured path. For each action taken, another line will be appended to that file, containing following information in csv-format:

method, input_type, input_source, mean_pld, sd, errors, and warnings.

Furthermore, the configuration of the config.txt is summarized at the top of the file. Figure 21 shows an example of a csv-formatted log file.

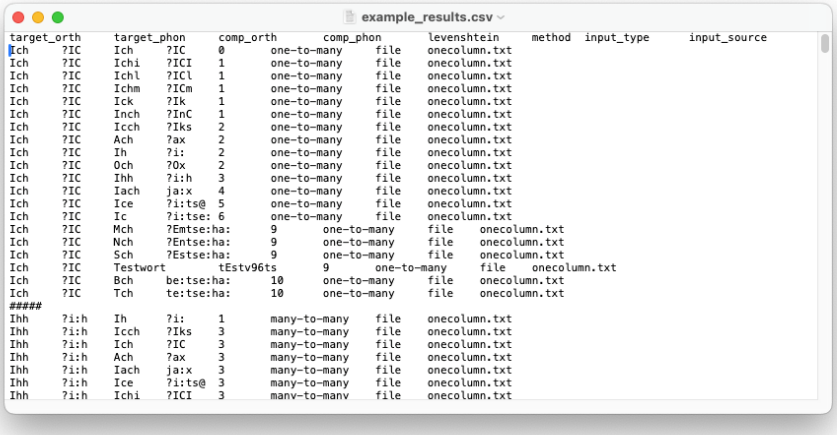


```
configuration: 2 corpora initialized, language: deu-DE, neighbours: 20
method  input_type      input_source    mean_pld        sd      errors  warnings
many-to-many    gui     list    None    None    cross compare only possible with more than 20 entries   these are less than 20 words!
one-to-many     file    onecolumn.txt   4.35    3.5535193822462827      None    None
many-to-many    file    onecolumn.txt   5.980952380952381       3.24398623795141        None    None
one-to-one      file    twocolumns.txt  None    None    None    None
one-to-many     corpus  Phon2-SUBTLEX-DE        0.9     0.30000000000000004     None    None
```

Figure 21. Log file that includes the error messages and information about the configuration

4.3. Printing the results

To separate the error messages and the results given by the system, messages and results will be written to different files. The result csv file is formatted as shown in Figure 22.



```
target_orth     target_phon     comp_orth       comp_phon       levenshtein     method  input_type      input_source
Ich     ?IC     Ich     ?IC     0       one-to-many     file    onecolumn.txt
Ich     ?IC     Ichi    ?ICI    1       one-to-many     file    onecolumn.txt
Ich     ?IC     Ichl    ?ICl    1       one-to-many     file    onecolumn.txt
Ich     ?IC     Ichm    ?ICm    1       one-to-many     file    onecolumn.txt
Ich     ?IC     Ick     ?Ik     1       one-to-many     file    onecolumn.txt
Ich     ?IC     Inch    ?InC    1       one-to-many     file    onecolumn.txt
Ich     ?IC     Icch    ?Iks    2       one-to-many     file    onecolumn.txt
Ich     ?IC     Ach     ?ax     2       one-to-many     file    onecolumn.txt
Ich     ?IC     Ih      ?i:     2       one-to-many     file    onecolumn.txt
Ich     ?IC     Och     ?Ox     2       one-to-many     file    onecolumn.txt
Ich     ?IC     Ihh     ?i:h    3       one-to-many     file    onecolumn.txt
Ich     ?IC     Iach    ja:x    4       one-to-many     file    onecolumn.txt
Ich     ?IC     Ice     ?i:ts@  5       one-to-many     file    onecolumn.txt
Ich     ?IC     Ic      ?i:tse: 6       one-to-many     file    onecolumn.txt
Ich     ?IC     Mch     ?Emtse:ha:      9       one-to-many     file    onecolumn.txt
Ich     ?IC     Nch     ?Entse:ha:      9       one-to-many     file    onecolumn.txt
Ich     ?IC     Sch     ?Estse:ha:      9       one-to-many     file    onecolumn.txt
Ich     ?IC     Testwort        tEstv96ts       9       one-to-many     file    onecolumn.txt
Ich     ?IC     Bch     be:tse:ha:      10      one-to-many     file    onecolumn.txt
Ich     ?IC     Tch     te:tse:ha:      10      one-to-many     file    onecolumn.txt
#####
Ihh     ?i:h    Ih      ?i:     1       many-to-many    file    onecolumn.txt
Ihh     ?i:h    Icch    ?Iks    3       many-to-many    file    onecolumn.txt
Ihh     ?i:h    Ich     ?IC     3       many-to-many    file    onecolumn.txt
Ihh     ?i:h    Ach     ?ax     3       many-to-many    file    onecolumn.txt
Ihh     ?i:h    Iach    ja:x    3       many-to-many    file    onecolumn.txt
Ihh     ?i:h    Ice     ?i:ts@  3       many-to-many    file    onecolumn.txt
Ihh     ?i:h    Ichi    ?ICI    3       many-to-many    file    onecolumn.txt
```

Figure 22. Result file corresponding to the protocol file shown in Figure 21. To show as much output as possible some results were hidden.

The results always contain the following information: orthographic and phonological forms of the target and the word it is being compared to, the Levenshtein distance established for each pair, the method and input_type used (e.g., one-to-many and file), and the input source. In the example shown in Figure 24, this source is the file onecolumn.txt.

5. Suggestions for dealing with homographs and homophones in a database

In some contexts, for instance before computing its PLD20 using PLD2flex, it can be useful to find and eliminate homographic and non-homographic homophones to a target as they may need to be treated individually. This is relevant as G2P will provide only a single transcript of any given written form, so it will generate homophonic word forms to all homographic entries in a database. Some of these entries, however, are phonologically distinct, such as the noun *minute* vs. the adjective *minute* in English, and G2P may assign a transcript of the non-intended form (G2P's transcript of minute is mInIt). If need be, users may want to correct such entries manually. The same holds for non-homographic homophones. For instance, in German, nominalized verbs are often phonologically identical to their citation form (e. g. kochen (*koch-en,* cook-inf, 'to cook') vs. Kochen (cook[NMLZN], the act of cooking). Hence they differ only very subtly in meaning, yet such forms are often listed as separate entries in lexical databases by virtue of belonging to different POS classes. Whether such word forms should be treated as neighbours of each other or not when it comes to computing their PLD20 or related measures depends on the research question at hand. For homographic and non-homographic homonyms, i. e. words with identical phonological forms but very distinct meanings, such as ball (in the sense of toy vs. dance convention), the case appears to be clearer as such forms probably should count as phonological neighbours of each other.

Given these specifics of such forms, it seemed inappropriate to implement a solution for dealing with homographic and homophonic forms that deals with all forms in the same way. Instead, we opted for providing an additional script that identifies all homographic word forms in a database so as to enable users to make an informed decision from case to case whether to keep both or only one form in the input file or the database (the script identifies homophonous word forms in a database, too). Users can start the script by typing

```
python3 analyze.py
```

in the main directory of the project. The script works with the same config.txt-file as PLD2flex and checks a database for homophones and/or homographs (on both levels) to help users deal with them before calculating anything.

User's of the .exe-files for Windows/MacOS may want to use a spreadsheet-based tool to sort the data base by phonological word forms and identify double entries in this way.

6. Expandability

The code of the tool is designed in a way that should allow for expandability. If you want to use parts of the code and adapt to your own needs, the files PLD20.py, BASConnector.py, and WordObject.py should be the most useful for that. If you are looking for a good way to start expanding the script, you may want to look at the analyze.py script (see section 5); it interacts with and uses parts of the code of the tool. Please credit the authorship of the present code when adapting or expanding it.

7. References

Baayen, R H., R Piepenbrock, Gulikers, L. (1995). *CELEX2 LDC96L14*. Web Download. Philadelphia: Linguistic Data Consortium.

Brysbaert, M., Buchmeier, M., Conrad, M., Jacobs, A. M., Bölte, J., & Böhl, A. (2011). The word frequency effect: A review of recent developments and implications for the choice of frequency estimates in German. *Experimental Psychology, 58*(5), 412–424. https://doi.org/10.1027/1618-3169/a000123

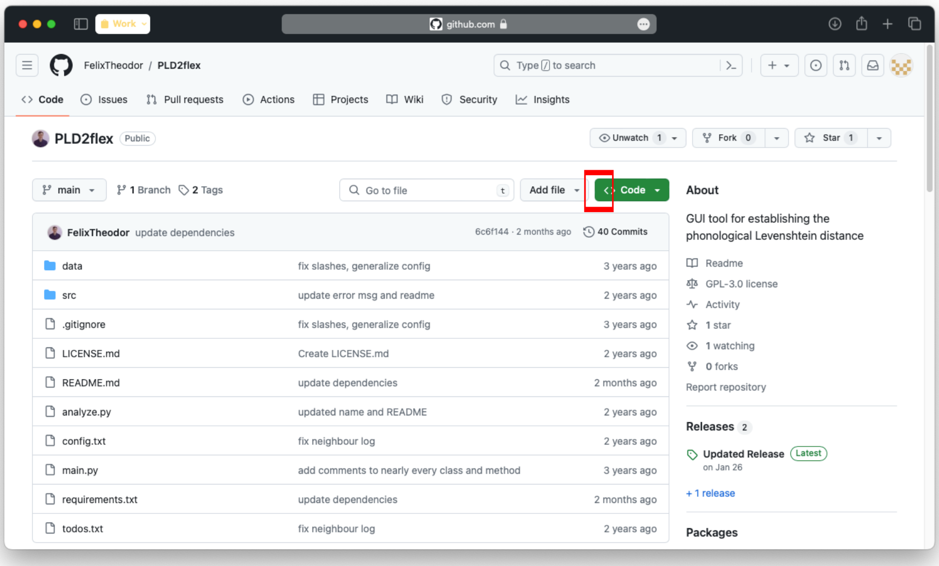
Keuleers, E., Brysbaert, M. & New, B. SUBTLEX-NL: A new measure for Dutch word frequency based on film subtitles. *Behavior Research Methods, 42*, 643–650 (2010). https://doi.org/10.3758/BRM.42.3.643

Reichel, U. D., Kisler, T. (2014). Language-independent grapheme-phoneme conversion and word stress assignment as a web service. In Hoffmann, R. (Ed.): *Elektronische Sprachverarbeitung. Studientexte zur Sprachkommunikation 71* (pp 42-49). Dresden: TUDpress.

Suárez, L., Tan, S. H., Yap, M. J., & Goh, W. D. (2011). Observing neighborhood effects without neighbors. *Psychonomic Bulletin & Review; 18,* 605–611. https://doi.org/10.3758/s13423-011-0078-9

Appendix A.1 – How to clone the repository

In order to use the tool, the repository (https://github.com/FelixTheodor/PLD2flex/) needs to be cloned. To do that, the user needs to open the github page and click on 'Code' which is marked red in Figure A.1.



FelixTheodor / PLD2flex

PLD2flex Public

main · 1 Branch · 2 Tags

FelixTheodor update dependencies · 6c6f144 · 2 months ago · 40 Commits

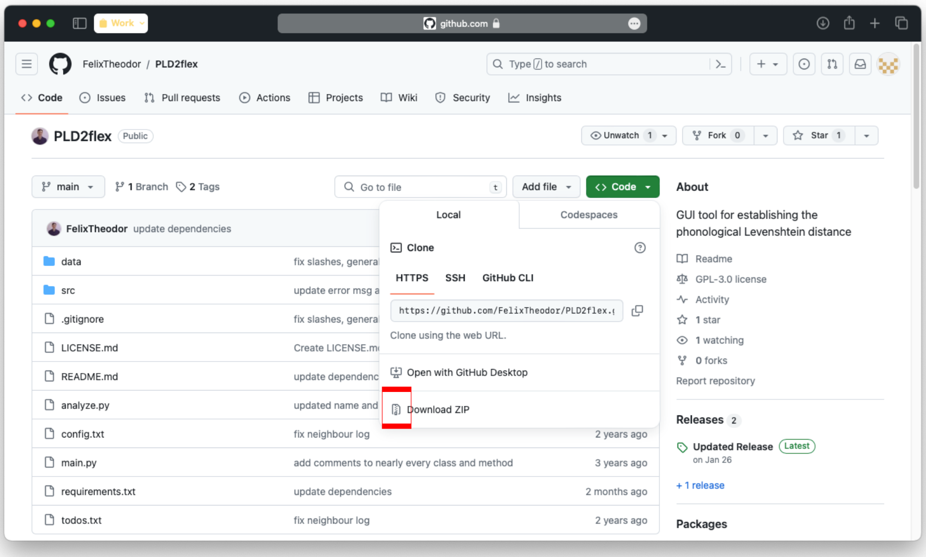
data	fix slashes, generalize config	3 years ago
src	update error msg and readme	2 years ago
.gitignore	fix slashes, generalize config	3 years ago
LICENSE.md	Create LICENSE.md	2 years ago
README.md	update dependencies	2 months ago
analyze.py	updated name and README	2 years ago
config.txt	fix neighbour log	2 years ago
main.py	add comments to nearly every class and method	3 years ago
requirements.txt	update dependencies	2 months ago
todos.txt	fix neighbour log	2 years ago

About

GUI tool for establishing the phonological Levenshtein distance

Figure A.1. Screenshot of the Github repository. The red marks the first step to clone the repository.

After clicking on "Code" a small window opens (Figure A.2). If the user clicks on 'Download ZIP'(as marked in red on Figure A.2), the download of the repository will be started. After that the folder needs to be unpacked and saved in the desired directory.



```
https://github.com/FelixTheodor/PLD2flex.(
```

Clone using the web URL.

Figure A.2. Screenshot of the Github repository. The red marks the second step to clone the repository.

Appendix A.2 – How to install the tool

We recommend to install and use PLD2flex in the virtual environment *conda*, and to thus install the presented modules via *conda*. In order to install PLD2flex, it is necessary to clone the Git repository or the zip-file onto your local machine (see Appendix A.1). Following, it is required to open the terminal in the main directory of the downloaded folder. Next, using conda a new environment has to be created with the command:

```
conda create -n PLD2flex python=3.12
```

The command

```
conda activate PLD2flex
```

allows users to work inside the newly created environment. To install *pip* the command

```
conda install pip
```

can be used. Lastly, the command

```
pip install -r requirements.txt
```

installs all necessary modules. Note that macOS users need to also install the package pyObjC (`pip install PyObjC).`

If the use of conda prompts the user to install python 3.12 first. It is recommended to install python 3.12 using the following website: https://www.python.org/downloads/windows (Windows).

If PLD2flex is used on Mac or Linux, installing is possible via homebrew by typing

```
brew install python.
```

Using the requirements.txt to install the needed modules, should allow for a easy install. However, if problems occure or the user does not want to use conda, the following modules need to be installed manually using the prompts

`python -m pip install module_name` (on Windows)

`brew install module_name` or `pip3 install module_name` (on MacOS and Linux).

Modules that need to be installed:

- `requests`
- `numpy`
- `textdistance`
- `playsound`
- `PyQt5`
- `PyObjC` (on MacOs)

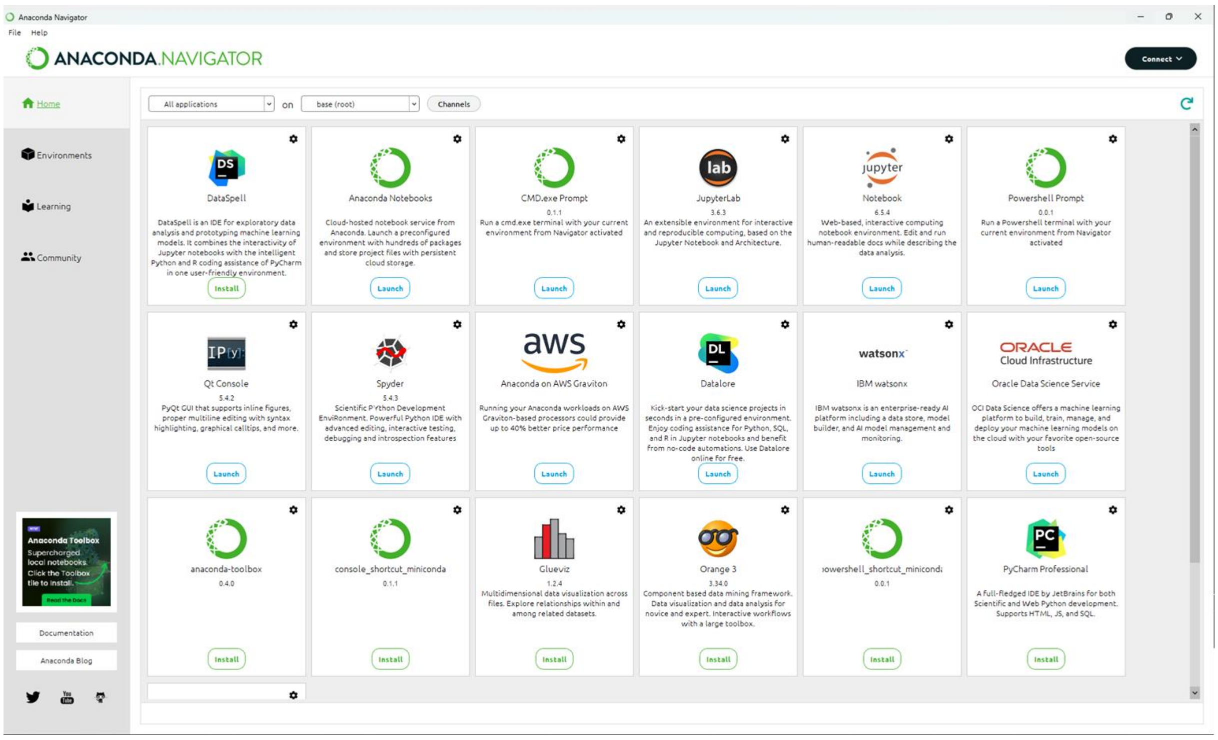
Aside from the installation of python and the corresponding modules, it may be necessary to install 'qt 5.15' via the following website (https://www.qt.io/offline-installers). While using PLD2flex a working internet connection is mandatory.

Appendix A.3 – A step-by-step guide to installing PLD2flex for Windows users

Skip this section if you got the PLD2flex tool to run as intended. This Appendix to section 2.1 of the PLD2flex user's manual is directed towards Windows users who have read through section 2.1 and were unsure what things like '*virtual environment*' or '*conda*' mean. If the installation process described in section 2.1 was completely clear to you and you have managed to get the program running, there is no need to read any further here. This section is written from the perspective of someone who is not very experienced with programming and thus found it quite challenging to follow some of the instructions given in the user's manual. For such users, additional steps not mentioned explicitly in section 2.1 were required to do so.

Let's start with the recommendation "*to install and use PLD2flex in a virtual environment […] and install the presented modules via conda*". *Conda* refers to a package that is included in the Anaconda distribution. You can download the Anaconda distribution on this site https://www.anaconda.com/download.

After installation you can start up the Anaconda Navigator, which should look similar to this:



DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

Anaconda Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.6.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.5.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
5.4.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
5.4.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Anaconda on AWS Graviton
Running your Anaconda workloads on AWS Graviton-based processors could provide up to 40% better price performance

Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

IBM watsonx
IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

anaconda-toolbox
0.4.0

console_shortcut_miniconda
0.1.1

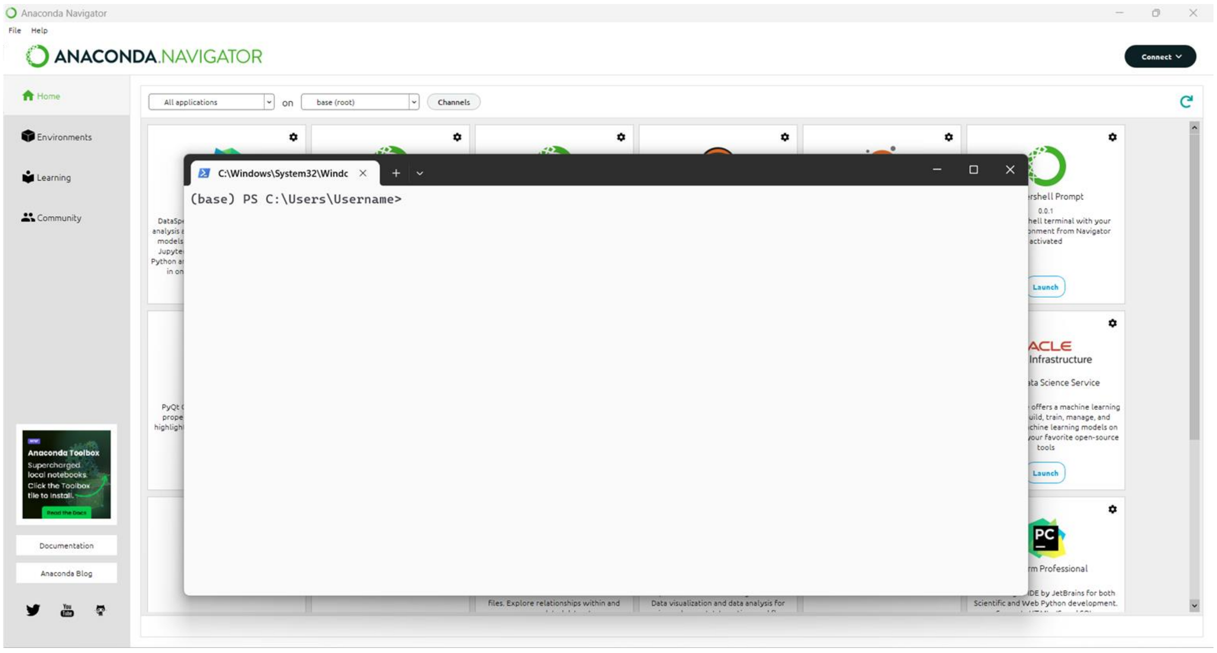
Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.34.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

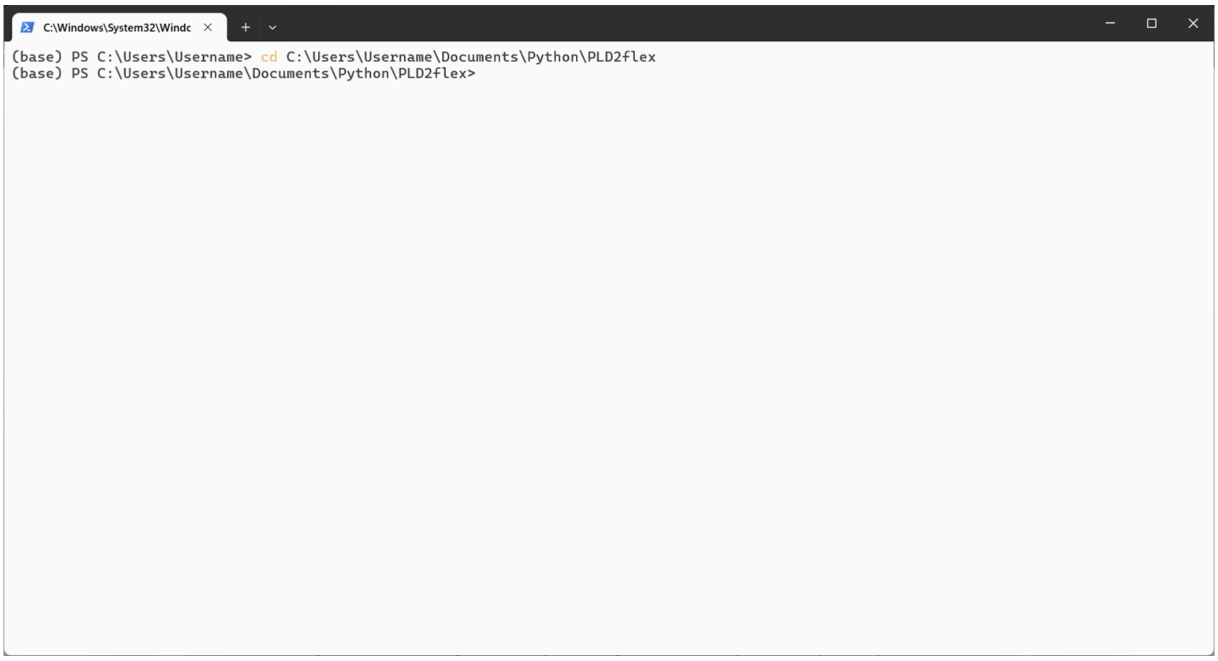
›owershell_shortcut_miniconda
0.0.1

PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

The user's manual gives the instruction to "*open the terminal in the main directory of the downloaded folder*", that is, the folder of the PLD2flex tool that contains the file *main.py*. To do this, you should launch the Powershell Prompt application in the Anaconda Navigator. This will open up a terminal where you can enter commands:


```
(base) PS C:\Users\Username>
```

Next, navigate towards the folder that contains the *main.py* file. You can achieve this by typing in '*cd*', followed by the directory. For me it looks like this:


```
(base) PS C:\Users\Username> cd C:\Users\Username\Documents\Python\PLD2flex
(base) PS C:\Users\Username\Documents\Python\PLD2flex>
```

At this point you can resume following the instructions given in section 2.1:

1 Create a new environment by typing in the command `conda create -n PLD2flex python=3.12`.

2 Type in the command `conda activate PLD2flex` to activate the environment.

3 Next, install pip using the command `conda install pip`.

Between each of these steps a bunch of things should be happening in the Powershell terminal. Sometimes you will be asked for confirmation to proceed by typing in '*y*'.

Next, the manual wants you to install the modules listed in section 2.1 by entering *`pip install -r requirements.txt`* into the Powershell terminal. At this point I encountered a problem which took me some time to solve. If all of the required modules install without issue: great! If not, here is a description of the problem I met:

At some point during the installation of the modules the process stopped returning this error:

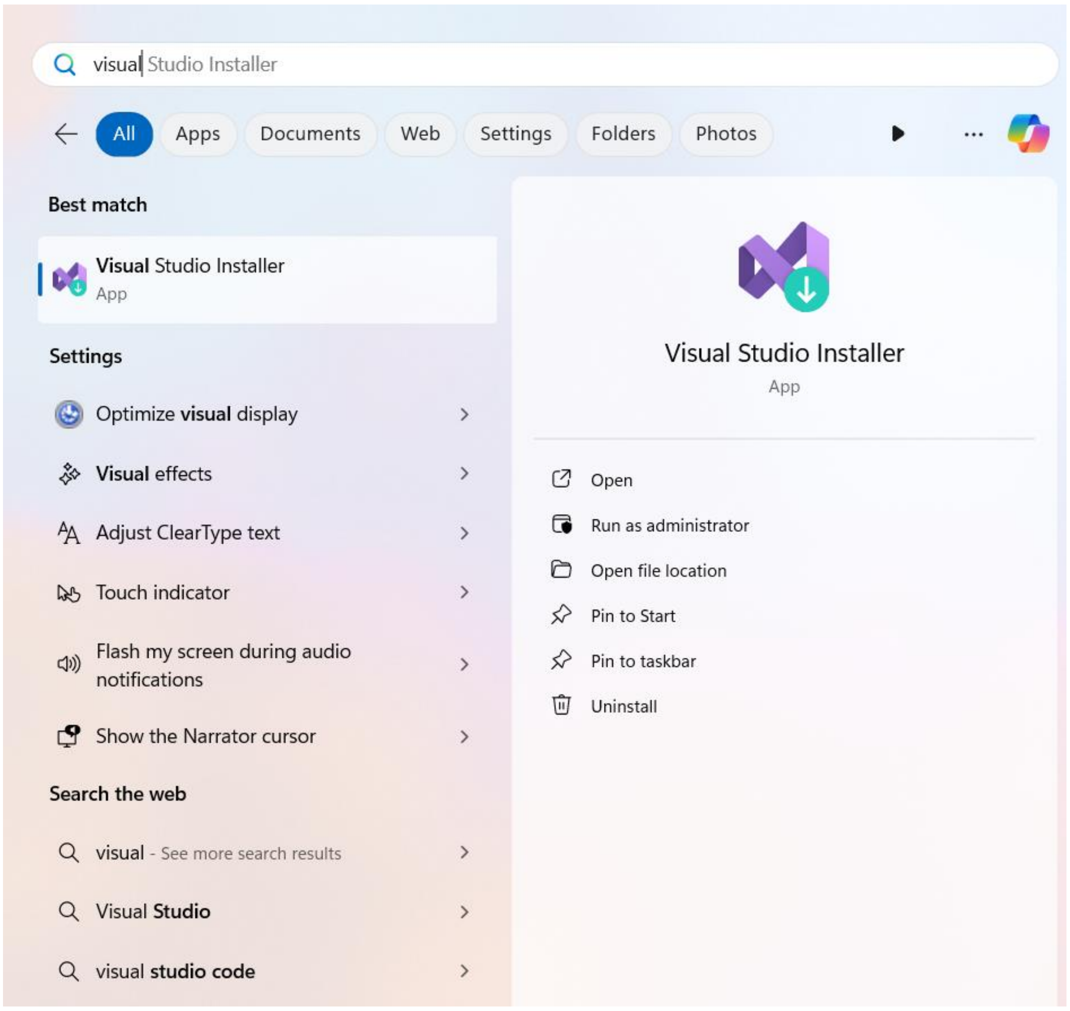
```
Successfully built playsound
Failed to build numpy
ERROR: Could not build wheels for numpy, which is required to install pyproject.toml-based projects
```

A little further above this message I found a clue of what turned out to be the issue:

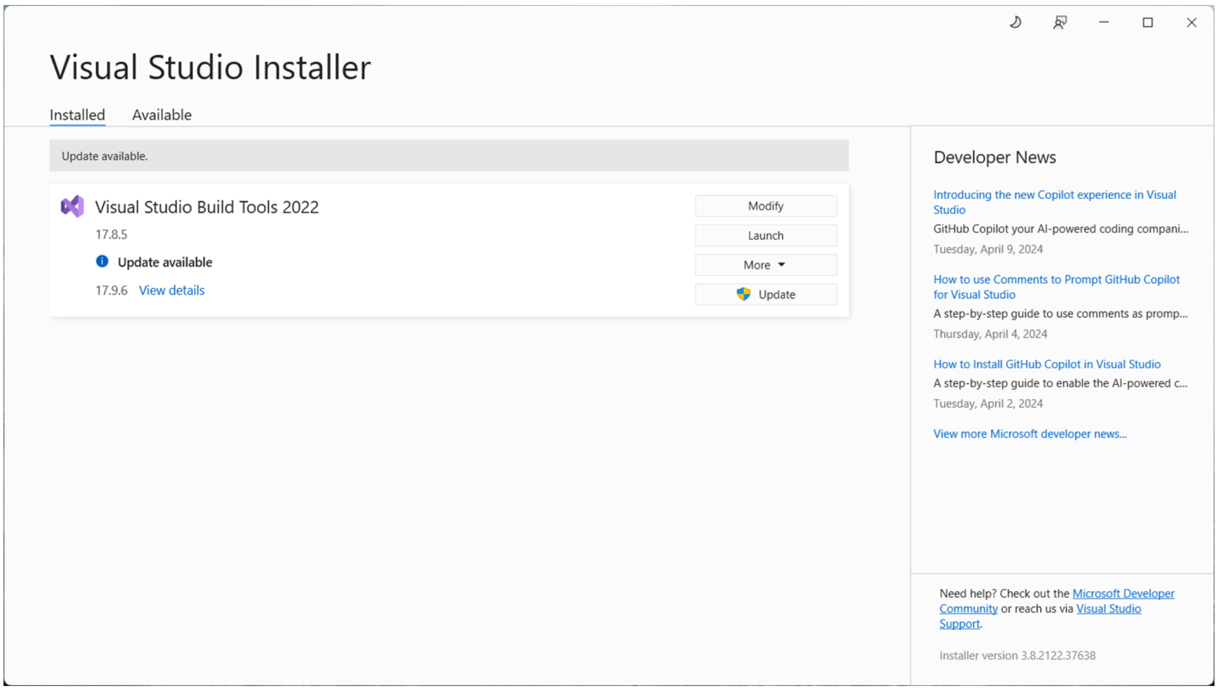
```
error: Microsoft Visual C++ 14.0 or greater is required. Get it with 'Microsoft C++ Build Tools': https://visualstudio.microsoft.com
/visual-cpp-build-tools/
[end of output]
```

If you come across this error as well, you should follow the link in that message and install Microsoft Build Tools. I installed PLD2flex on a relatively new Windows device that did not have Build Tools preinstalled. As another tester of PLD2flex also encountered this problem on a Windows device, I would assume this goes for many other Windows users as well.

After installing Build Tools start up a program called Visual Studio Installer. Find it in the Windows Start menu.



Start the installer. You should be looking at something like this:


Visual Studio Installer

Installed   Available

Update available.

Visual Studio Build Tools 2022
17.8.5
Update available
17.9.6 View details

Modify
Launch
More
Update

Developer News

Introducing the new Copilot experience in Visual Studio
GitHub Copilot your AI-powered coding compani...
Tuesday, April 9, 2024

How to use Comments to Prompt GitHub Copilot for Visual Studio
A step-by-step guide to use comments as promp...
Thursday, April 4, 2024

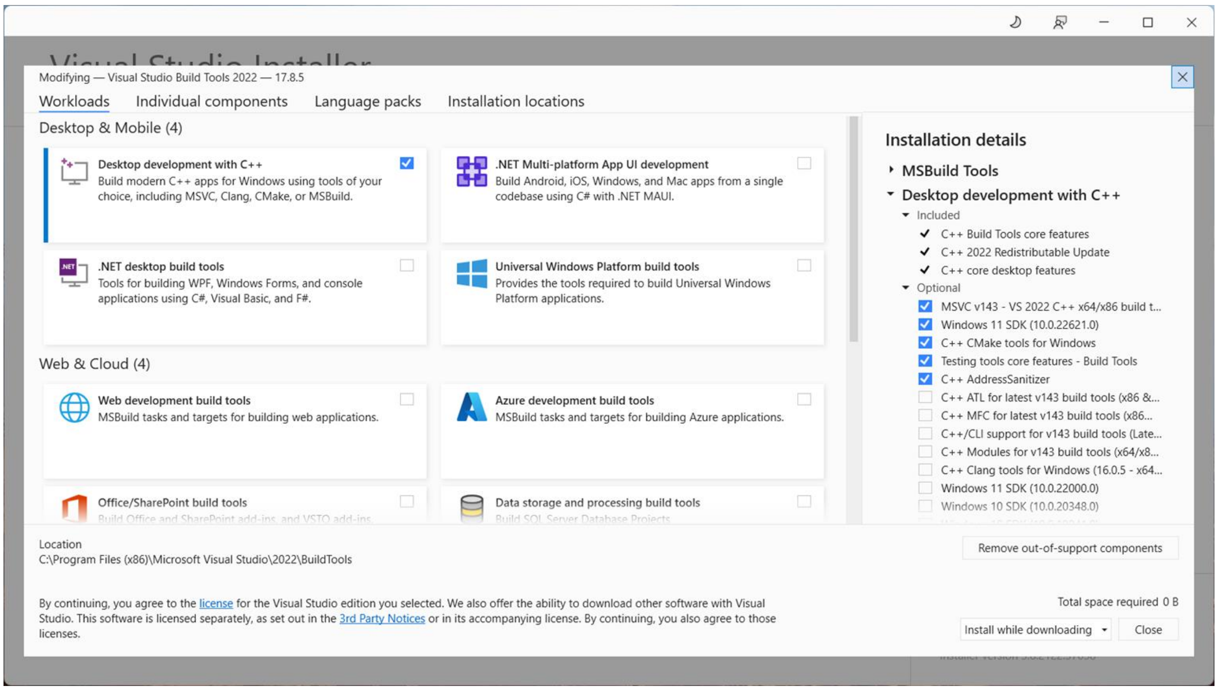
How to Install GitHub Copilot in Visual Studio
A step-by-step guide to enable the AI-powered c...
Tuesday, April 2, 2024

View more Microsoft developer news...

Need help? Check out the Microsoft Developer Community or reach us via Visual Studio Support.

Installer version 3.8.2122.37638

Click on *Modify*, which should open up another window.


Modifying — Visual Studio Build Tools 2022 — 17.8.5

Workloads   Individual components   Language packs   Installation locations

Desktop & Mobile (4)

Desktop development with C++ ✓
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

.NET Multi-platform App UI development
Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.

.NET desktop build tools
Tools for building WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

Universal Windows Platform build tools
Provides the tools required to build Universal Windows Platform applications.

Web & Cloud (4)

Web development build tools
MSBuild tasks and targets for building web applications.

Azure development build tools
MSBuild tasks and targets for building Azure applications.

Office/SharePoint build tools
Build Office and SharePoint add-ins, and VSTO add-ins

Data storage and processing build tools
Build SQL Server Database Projects

Installation details

- MSBuild Tools
- Desktop development with C++
  - Included
    - ✓ C++ Build Tools core features
    - ✓ C++ 2022 Redistributable Update
    - ✓ C++ core desktop features
  - Optional
    - ✓ MSVC v143 - VS 2022 C++ x64/x86 build t...
    - ✓ Windows 11 SDK (10.0.22621.0)
    - ✓ C++ CMake tools for Windows
    - ✓ Testing tools core features - Build Tools
    - ✓ C++ AddressSanitizer
    - ☐ C++ ATL for latest v143 build tools (x86 &...
    - ☐ C++ MFC for latest v143 build tools (x86...
    - ☐ C++/CLI support for v143 build tools (Late...
    - ☐ C++ Modules for v143 build tools (x64/x8...
    - ☐ C++ Clang tools for Windows (16.0.5 - x64...
    - ☐ Windows 11 SDK (10.0.22000.0)
    - ☐ Windows 10 SDK (10.0.20348.0)

Location
C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools

Remove out-of-support components

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

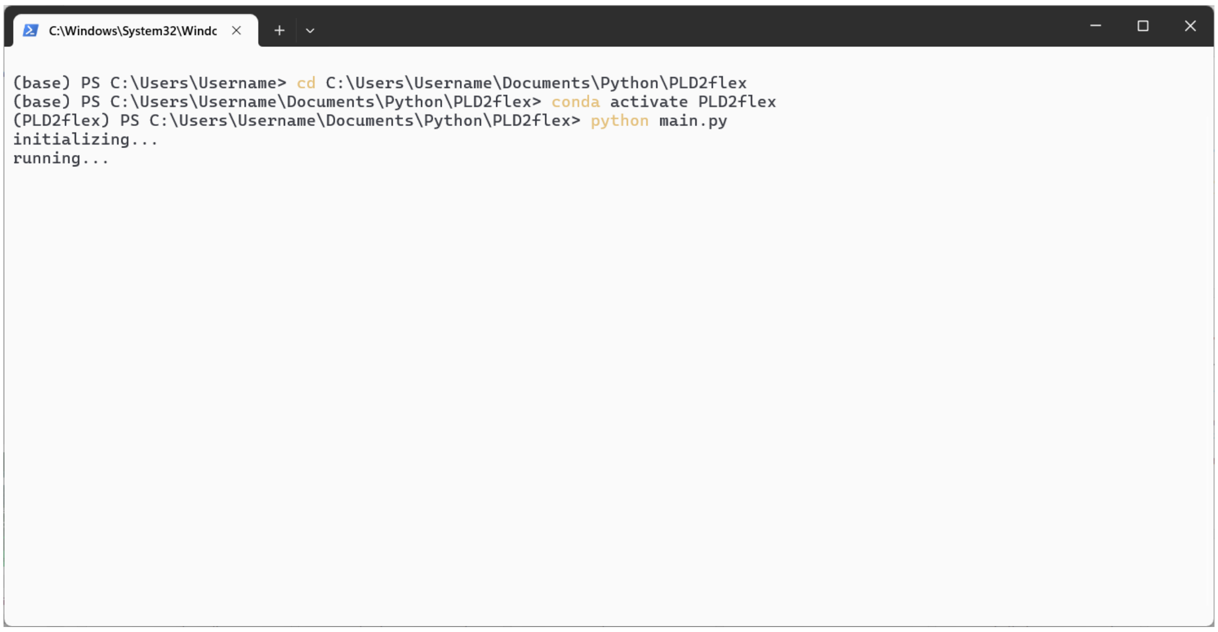
Total space required 0 B

Install while downloading   Close

Next, you need to set the check mark in the top right corner of *Desktop development with C++* and then install it. This installation is rather large so it might take some time.

After this you should return to the PLD2flex environment in your Powershell terminal and try to install the required modules again by entering `pip install -r requirements.txt`. This time all modules should install without issue (hopefully). This concludes the installation of the PLD2flex tool. Start up the tool by entering `python main.py` into you Powershell terminal.

One last remark: Each time you want to start the PLD2flex tool anew, you will have to navigate to the folder that contains the *main.py* file in the Powershell terminal, then activate the PLD2flex environment and finally type in `python main.py`. You will not have to install the required modules again though. The whole process to start up PLD2flex after its first use is shown once again in this screenshot:



```
(base) PS C:\Users\Username> cd C:\Users\Username\Documents\Python\PLD2flex
(base) PS C:\Users\Username\Documents\Python\PLD2flex> conda activate PLD2flex
(PLD2flex) PS C:\Users\Username\Documents\Python\PLD2flex> python main.py
initializing...
running...
```