

---

# **unihan-etl Documentation**

***Release 0.10.3***

**Tony Narlock**

**Aug 18, 2019**



# CONTENTS

<b>1</b>	<b>Tabular, “Flat” output</b>	<b>3</b>
<b>2</b>	<b>“Structured” output</b>	<b>5</b>
<b>3</b>	<b>Features</b>	<b>7</b>
<b>4</b>	<b>Usage</b>	<b>9</b>
<b>5</b>	<b>Code layout</b>	<b>11</b>
5.1	About unihan-etl . . . . .	11
5.2	About UNIHAN . . . . .	12
5.3	Command Line Interface . . . . .	15
5.4	API . . . . .	16
5.5	Frequently Asked Questions . . . . .	20
5.6	History . . . . .	20
	<b>Python Module Index</b>	<b>25</b>
	<b>Index</b>	<b>27</b>



*unihan-etl* - ETL tool for Unicode's Han Unification (UNIHAN) database releases. *unihan-etl* retrieves (downloads), extracts (unzips), and transforms the database from Unicode's website to a flat, tabular or structured, tree-like format.

*unihan-etl* can be used as a python library through its API, to retrieve data as a python object, or through the CLI to retrieve a CSV, JSON, or YAML file.

Part of the [cihai](#) project. Similar project: [libUnihan](#).

UNIHAN Version compatibility (as of *unihan-etl* v0.10.0): 11.0.0 (released 2018-05-08, revision 25).

UNIHAN's data is dispersed across multiple files in the format of:

U+3400	kCantonese	jau1
U+3400	kDefinition	(same <b>as</b> U+4E18 ) hillock <b>or</b> mound
U+3400	kMandarin	qiū
U+3401	kCantonese	tim2
U+3401	kDefinition	to lick; to taste, a mat, bamboo bark
U+3401	kHanyuPinyin	10019.020:tiàn
U+3401	kMandarin	tiàn

Values vary in shape and structure depending on their field type. *kHanyuPinyin* maps Unicode codepoints to Hànyǔ Dà Zìdiǎn, where 10019.020:tiàn represents an entry. Complicating it further, more variations:

U+5EFE	kHanyuPinyin	10513.110,10514.010,10514.020:gǒng
U+5364	kHanyuPinyin	10093.130:xī,lǚ 74609.020:lǚ,xī

*kHanyuPinyin* supports multiple entries delimited by spaces. “:” (colon) separate locations in the work from pinyin readings. “,” (comma) separate multiple entries/readings. This is just one of 90 fields contained in the database.



## TABULAR, “FLAT” OUTPUT

CSV (default), \$ unihan-etl:

```
char,ucn,kCantonese,kDefinition,kHanyuPinyin,kMandarin
,U+3400,jau1,(same as U+4E18 ) hillock or mound,,qiū
,U+3401,tim2,"to lick; to taste, a mat, bamboo bark",10019.020:tiàn,tiàn
```

With \$ unihan-etl -F yaml --no-expand:

```
- char:
  kCantonese: jau1
  kDefinition: (same as U+4E18 ) hillock or mound
  kHanyuPinyin: null
  kMandarin: qiū
  ucn: U+3400
- char:
  kCantonese: tim2
  kDefinition: to lick; to taste, a mat, bamboo bark
  kHanyuPinyin: 10019.020:tiàn
  kMandarin: tiàn
  ucn: U+3401
```

With \$ unihan-etl -F json --no-expand:

```
[
  {
    "char": "",
    "ucn": "U+3400",
    "kDefinition": "(same as U+4E18 ) hillock or mound",
    "kCantonese": "jau1",
    "kHanyuPinyin": null,
    "kMandarin": "qiū"
  },
  {
    "char": "",
    "ucn": "U+3401",
    "kDefinition": "to lick; to taste, a mat, bamboo bark",
    "kCantonese": "tim2",
    "kHanyuPinyin": "10019.020:tiàn",
    "kMandarin": "tiàn"
  }
]
```





## “STRUCTURED” OUTPUT

Codepoints can pack a lot more detail, unihan-etl carefully extracts these values in a uniform manner. Empty values are pruned.

To make this possible, unihan-etl exports to JSON, YAML, and python list/dicts.

---

### Why not CSV?

Unfortunately, CSV is only suitable for storing table-like information. File formats such as JSON and YAML accept key-values and hierarchical entries.

---

JSON, \$ unihan-etl -F json:

```
[
  {
    "char": "",
    "ucn": "U+3400",
    "kDefinition": [
      "(same as U+4E18 ) hillock or mound"
    ],
    "kCantonese": [
      "jau1"
    ],
    "kMandarin": {
      "zh-Hans": "qiū",
      "zh-Hant": "qiū"
    }
  },
  {
    "char": "",
    "ucn": "U+3401",
    "kDefinition": [
      "to lick",
      "to taste, a mat, bamboo bark"
    ],
    "kCantonese": [
      "tim2"
    ],
    "kHanyuPinyin": [
      {
        "locations": [
          {
            "volume": 1,
            "page": 19,

```

(continues on next page)

(continued from previous page)

```
        "character": 2,
        "virtual": 0
      }
    ],
    "readings": [
      "tiàn"
    ]
  }
],
"kMandarin": {
  "zh-Hans": "tiàn",
  "zh-Hant": "tiàn"
}
}
```

YAML \$ unihan-etl -F yaml:

```
- char:
  kCantonese:
    - jau1
  kDefinition:
    - (same as U+4E18 ) hillock or mound
  kMandarin:
    zh-Hans: qiū
    zh-Hant: qiū
  ucn: U+3400
- char:
  kCantonese:
    - tim2
  kDefinition:
    - to lick
    - to taste, a mat, bamboo bark
  kHanyuPinyin:
    - locations:
        - character: 2
          page: 19
          virtual: 0
          volume: 1
        readings:
          - tiàn
  kMandarin:
    zh-Hans: tiàn
    zh-Hant: tiàn
  ucn: U+3401
```

## FEATURES

- automatically downloads UNIHAN from the internet
- strives for accuracy with the specifications described in [UNIHAN's database design](#)
- export to JSON, CSV and YAML (requires [pyyaml](#)) via `-F`
- configurable to export specific fields via `-f`
- accounts for encoding conflicts due to the Unicode-heavy content
- designed as a technical proof for future CJK (Chinese, Japanese, Korean) datasets
- core component and dependency of [cikai](#), a CJK library
- [data package](#) support
- expansion of multi-value delimited fields in YAML, JSON and python dictionaries
- supports python 2.7, >= 3.5 and pypy

If you encounter a problem or have a question, please [create an issue](#).



## USAGE

`unihan-etl` offers customizable builds via its command line arguments.

See [unihan-etl CLI arguments](#) for information on how you can specify columns, files, download URL's, and output destination.

To download and build your own UNIHAN export:

```
$ pip install --user unihan-etl
```

To output CSV, the default format:

```
$ unihan-etl
```

To output JSON:

```
$ unihan-etl -F json
```

To output YAML:

```
$ pip install --user pyyaml  
$ unihan-etl -F yaml
```

To only output the `kDefinition` field in a csv:

```
$ unihan-etl -f kDefinition
```

To output multiple fields, separate with spaces:

```
$ unihan-etl -f kCantonese kDefinition
```

To output to a custom file:

```
$ unihan-etl --destination ./exported.csv
```

To output to a custom file (templated file extension):

```
$ unihan-etl --destination ./exported.{ext}
```

See [unihan-etl CLI arguments](#) for advanced usage examples.



## CODE LAYOUT

```
# cache dir (Unihan.zip is downloaded, contents extracted)
{XDG cache dir}/unihan_etl/

# output dir
{XDG data dir}/unihan_etl/
    unihan.json
    unihan.csv
    unihan.yaml    # (requires pyyaml)

# package dir
unihan_etl/
    process.py      # argparse, download, extract, transform UNIHAN's data
    constants.py    # immutable data vars (field to filename mappings, etc)
    expansion.py    # extracting details baked inside of fields
    _compat.py      # python 2/3 compatibility module
    util.py         # utility / helper functions

# test suite
tests/*
```

### 5.1 About unihan-etl

unihan-etl provides configurable, self-serve data exports of the *About UNIHAN* database.

#### 5.1.1 Retrieval

unihan-etl will download and cache the raw database files for the user.

#### 5.1.2 No encoding headaches

Dealing with unicode encodings can be cumbersome across platforms. unihan-etl deals with handling output encoding issues that could come up if you were to try to export the data yourself.

#### 5.1.3 Python 2 and 3

Designed and tested to work across Python versions. View the [travis test matrix](#) for what this software is tested against.

## 5.1.4 Customizable output

### Formats

- CSV
- JSON
- YAML (requires [pyyaml](#))
- Python dict (via *API*)

### “Structured” output

*JSON, YAML, and python dict only*

Support for structured output of information in fields. unihan-etl refers to this as *expansion*.

Users can opt-out via `--no-expand`. This will preserve the values in each field as they are in the raw database.

Filters out empty values by default, opt-out via `--no-prune`.

### Filtering

Support for filtering by fields and files.

To specify which fields to output, use `-f / --fields` and separate them in spaces. `-f kDefinition kCantonese kHanyuPinyin`.

For files, `-i / --input-files`. Example: `-i Unihan_DictionaryLikeData.txt Unihan_Readings.txt`.

## 5.2 About UNIHAN

### See also:

- [Wikipedia article](#)
- [UNIHAN database documentation](#)

### 5.2.1 Languages, Computers, and You

There are many languages and writing systems around the world. Computers internally use numbers to represent characters in writing systems. As computers became more prominent, hundreds of encoding systems were developed to handle writing systems from different regions.

No single encoding system covered all languages. Adding to the complexity, encodings conflicted with each other on the numbers assigned to characters. Any data decoded with the wrong standard would turn up as gibberish.

[Unicode](#) is a standard devised to provide a unique number for every character.

This entails pulling together minds from around the world to assign codepoints.

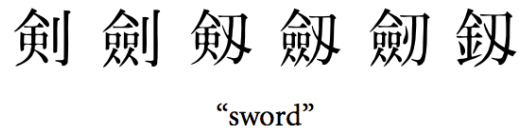
The *Unicode Consortium* is a non-profit organization founded to develop, extend and promote use of the Unicode Standard.



## 5.2.2 What is UNIHAN?

UNIHAN, short for [Han unification](#), is the effort of the consortium assign codepoints to CJK characters. Any single *han character* can multiple historical or regional variants to account for, hence “unification”.

Figure 18-4. CJK Source Separation



To do this, various sources of information are pulled together and cross-referenced to detail characteristics of the glyphs, and vet them through a thorough proofreading process. It’s an international effort, hallmarked by between researchers and groups like the [Ideographic Rapporteur Group](#). Glyphs once only noted in dictionaries and antiquity are set in stone with their own codepoints, carefully cross-referenced with information from, often multiple, distinct sources.

The advantage that UNIHAN provides to east asian researchers, including sinologists and japanologists, linguists, analysts, language learners, and hobbyists cannot be understated. Unbeknownst to users, its used under the hood in many applications and websites.

The resulting standard has industrial ramifications downstream to software developers and computer users. When a version of Unicode is released, it is then incorporated downstream in software projects.

## 5.2.3 The database

UNIHAN provides a database of its information, which is the culmination of CJK information that has been vetted and proofed painstakingly over years.

You can view the [UNIHAN Database documentation](#) to see where information on each field of information is derived from. For instance:

- **kCantonese**: The Cantonese pronunciation(s) for this character using the [jyutping romanization](#).

Bibliography:

1. Casey, G. Hugh, S.J. Ten Thousand Characters: An Analytic Dictionary. Hong Kong: Kelley and Walsh, 1980 (kPhonetic).
2. Cheung Kwan-hin and Robert S. Bauer, The Representation of Cantonese with Chinese Characters, Journal of Chinese Linguistics Monograph Series Number 18, 2002.
3. Roy T. Cowles, A Pocket Dictionary of Cantonese, Hong Kong: University Press, 1999 (kCowles).
4. Sidney Lau, A Practical Cantonese-English Dictionary, Hong Kong: Government Printer, 1977 (kLau).
5. Bernard F. Meyer and Theodore F. Wempe, Student’s Cantonese-English Dictionary, Maryknoll, New York: Catholic Foreign Mission Society of America, 1947 (kMeyerWempe).
6. , ed. , Hong Kong: Joint Publishing (H.K.) Co., Ltd., 1989.
7. , Hong Kong:, 1987.
8. , ed. , Hong Kong: The Commercial Press, 1991.
9. , Hong Kong: Longman, 2001.

- **kHanYu**: The position of this character in the Hanyu Da Zidian (HDZ) Chinese character dictionary.

Bibliography:

1. <Hanyu Da Zidian> [‘Great Chinese Character Dictionary’ (in 8 Volumes)]. XU Zhongshu (Editor in Chief). Wuhan, Hubei Province (PRC): Hubei and Sichuan Dictionary Publishing Collectives, 1986-1990. ISBN: 7-5403-0030-2/H.16.
- **kHanyuPinyin**: The Hànyǔ Pīnyīn reading(s) appearing in the edition of Hànyǔ Dà Zìdiǎn (HDZ) specified in the “kHanYu” property description (q.v.).

#### Bibliography:

- This data was originally input by Jǐng Zuòhéng
- proofed by Dān Māgē (Magda Danish, using software donated by Wénlín Institute, Inc. and tables prepared by Qū Lǐchá),
- and proofed again and prepared for the Unicode Consortium by Qū Lǐchá (2008-01-14).

Han Unification is a global effort. And it’s available free to the world.

## 5.2.4 The problem

It’s difficult to readily take advantage of UNIHAN database in its raw form.

UNIHAN comprises over 20 MB of character information, separated across multiple files. Within these files is 90 fields, spanning 8 general categories of data. Within some of fields, there are specific considerations to take account of to use the data correctly, for instance:

UNIHAN’s values place references to its own codepoints, such as *kDefinition*:

U+3400	kDefinition	(same as U+4E18 ) hillock or mound
--------	-------------	------------------------------------

And also by spaces, such as in *kCantonese*:

U+342B	kCantonese	gun3 hung1 zung1
--------	------------	------------------

And by spaces which specify different sources, like *kMandarin*, “When there are two values, then the first is preferred for zh-Hans (CN) and the second is preferred for zh-Hant (TW). When there is only one value, it is appropriate for both.”:

U+7E43	kMandarin	běng bēng
--------	-----------	-----------

Another, values are delimited in various ways, for instance, by rules, like *kDefinition*, “Major definitions are separated by semicolons, and minor definitions by commas.”:

U+3402	kDefinition	(J) non-standard form of U+559C , to like, love, enjoy; ↪ a joyful thing
--------	-------------	---

More complicated yet, *kHanyuPinyin*: “multiple locations for a given pīnyīn reading are separated by “,” (comma). The list of locations is followed by “:” (colon), followed by a comma-separated list of one or more pīnyīn readings. Where multiple pīnyīn readings are associated with a given mapping, these are ordered as in HDZ (for the most part reflecting relative commonality). The following are representative records.”:

U+3FCE	kHanyuPinyin	42699.050:fèn, fén
U+34D8	kHanyuPinyin	10278.080,10278.090:sù
U+5364	kHanyuPinyin	10093.130:xī, lǚ 74609.020:lǚ, xī
U+5EFE	kHanyuPinyin	10513.110,10514.010,10514.020:gǒng

Data could be exported to a CSV, but users wouldn’t be able to handle delimited values and structured information held within.

Since CSV does not support structured information, another format that supports needs to be found.

Even then, users may not want an export that expands the structured output of fields. So if a tool exists, exports should be configurable. Users could then export a field with `gun3 hung1 zung1` pristinely without turning it into list form.

## 5.3 Command Line Interface

Export UNIHAN to Python, Data Package, CSV, JSON and YAML

```
usage: unihan-etl [-h] [-v] [-s SOURCE] [-z ZIP_PATH] [-d DESTINATION]
                 [-w WORK_DIR] [-F {json,csv}] [--no-expand] [--no-prune]
                 [-f [FIELDS [FIELDS ...]]]
                 [-i [INPUT_FILES [INPUT_FILES ...]]]
                 [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
```

### 5.3.1 Named Arguments

<b>-v, --version</b>	show program's version number and exit
<b>-s, --source</b>	URL or path of zipfile. Default: <a href="http://www.unicode.org/Public/UNIDATA/Unihan.zip">http://www.unicode.org/Public/UNIDATA/Unihan.zip</a>
<b>-z, --zip-path</b>	Path the zipfile is downloaded to. Default: <code>/home/docs/.cache/unihan_etl/downloads/Unihan.zip</code>
<b>-d, --destination</b>	Output of .csv. Default: <code>/home/docs/.local/share/unihan_etl/unihan.{json,csv,yaml}</code>
<b>-w, --work-dir</b>	Default: <code>/home/docs/.cache/unihan_etl/downloads</code>
<b>-F, --format</b>	Possible choices: json, csv Default: csv
<b>--no-expand</b>	Don't expand values to lists in multi-value UNIHAN fields. Doesn't apply to CSVs. Default: True
<b>--no-prune</b>	Don't prune fields with empty keysDoesn't apply to CSVs. Default: True
<b>-f, --fields</b>	Fields to use in export. Separated by spaces. All fields used by default. Fields: kAccountingNumeric, kBigFive, kCCCCII, kCNS1986, kCNS1992, kCangjie, kCantonese, kCheungBauer, kCheungBauerIndex, kCihaiT, kCompatibilityVariant, kCowles, kDaeJaweon, kDefinition, kEACC, kFenn, kFennIndex, kFourCornerCode, kFrequency, kGB0, kGB1, kGB3, kGB5, kGB7, kGB8, kGSR, kGradeLevel, kHDZRadBreak, kHKGlyph, kHKSCS, kHanYu, kHangul, kHanyuPinlu, kHanyuPinyin, kIBMJapan, kIICore, kIRGDaeJaweon, kIRGDaiKanwaZiten, kIRGHanyuDaZidian, kIRGKangXi, kIRG_GSource, kIRG_HSource, kIRG_JSource, kIRG_KPSource, kIRG_KSource, kIRG_MSource, kIRG_TSource, kIRG_USource, kIRG_VSource, kJIS0213, kJa, kJapaneseKun, kJapaneseOn, kJinmeiyoKanji, kJis0, kJis1, kJoyoKanji, kKPS0, kKPS1, kKSC0, kKSC1, kKangXi, kKarlren, kKorean, kKoreanEducationHanja, kKoreanName, kLau, kMainlandTelegraph, kMandarin, kMatthews, kMeyerWempe, kMorohashi, kNelson, kOtherNumeric, kPhonetic, kPrimaryNumeric, kPseudoGB1, kRSAdobe_Japan1_6, kRSJapanese,

	kRSKanWa, kRSKangXi, kRSKorean, kRSUnicode, kSBGY, kSemanticVariant, kSimplifiedVariant, kSpecializedSemanticVariant, kTGH, kTaiwanTelegraph, kTang, kTotalStrokes, kTraditionalVariant, kVietnamese, kXHC1983, kXerox, kZVariant
<b>-i, --input-files</b>	Files inside zip to pull data from. Separated by spaces. All files used by default. Files: Unihan_DictionaryIndices.txt, Unihan_DictionaryLikeData.txt, Unihan_IRGSources.txt, Unihan_NumericValues.txt, Unihan_OtherMappings.txt, Unihan_RadicalStrokeCounts.txt, Unihan_Readings.txt, Unihan_Variants.txt
<b>-l, --log_level</b>	Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

## 5.4 API

Build Unihan into tabular / structured format and export it.

**class** `unihan_etl.process.Packager` (*options*)  
Download and generate a tabular release of **UNIHAN**.

**download** (*urlretrieve\_fn=<function urlretrieve>*)  
Download raw UNIHAN data if not exists.

**Parameters** *urlretrieve\_fn* (*function*) – function to download file

**export** ()  
Extract zip and process information into CSV's.

**classmethod** `from_cli` (*argv*)  
Create Packager instance from CLI `argparse` arguments.

**Parameters** *argv* (*list*) – Arguments passed in via CLI.

**Returns** builder

**Return type** *Packager*

`unihan_etl.process.ALLOWED_EXPORT_TYPES = ['json', 'csv']`  
Allowed export types

`unihan_etl.process.DESTINATION_DIR = '/home/docs/.local/share/unihan_etl'`  
Filepath to output built CSV file to.

**class** `unihan_etl.process.Packager` (*options*)  
Download and generate a tabular release of **UNIHAN**.

**download** (*urlretrieve\_fn=<function urlretrieve>*)  
Download raw UNIHAN data if not exists.

**Parameters** *urlretrieve\_fn* (*function*) – function to download file

**export** ()  
Extract zip and process information into CSV's.

**classmethod** `from_cli` (*argv*)  
Create Packager instance from CLI `argparse` arguments.

**Parameters** *argv* (*list*) – Arguments passed in via CLI.

**Returns** builder

**Return type** *Packager*

`unihan_etl.process.UNIHAN_FIELDS = ('kAccountingNumeric', 'kBigFive', 'kCCCCII', 'kCNS1986', ...)`  
 Default Unihan fields

`unihan_etl.process.UNIHAN_FILES = dict_keys(['Unihan_DictionaryIndices.txt', 'Unihan_DictionaryData.txt', ...])`  
 Default Unihan Files

`unihan_etl.process.UNIHAN_URL = 'http://www.unicode.org/Public/UNIDATA/Unihan.zip'`  
 URI of Unihan.zip data.

`unihan_etl.process.UNIHAN_ZIP_PATH = '/home/docs/.cache/unihan_etl/downloads/Unihan.zip'`  
 Filepath to download Zip file.

`unihan_etl.process.WORK_DIR = '/home/docs/.cache/unihan_etl/downloads'`  
 Directory to use for processing intermittent files.

`unihan_etl.process.download(url, dest, urlretrieve_fn=<function urlretrieve>, reporthook=None)`  
 Download file at URL to a destination.

#### Parameters

- **url** (*str*) – URL to download from.
- **dest** (*str*) – file path where download is to be saved.
- **urlretrieve\_fn** (*callable*) – function to download file
- **reporthook** (*function*) – Function to write progress bar to stdout buffer.

**Returns** destination where file downloaded to.

**Return type** `str`

`unihan_etl.process.expand_delimiters(normalized_data)`  
 Return expanded multi-value fields in UNIHAN.

**Parameters** **normalized\_data** (*list of dict*) – Expects data in list of hashes, per `process.normalize()`

**Returns** Items which have fields with delimiters and custom separation rules, will be expanded. Including multi-value fields not using both fields (so all fields stay consistent).

**Return type** list of dict

`unihan_etl.process.extract_zip(zip_path, dest_dir)`  
 Extract zip file. Return `zipfile.ZipFile` instance.

#### Parameters

- **zip\_file** (*str*) – filepath to extract.
- **dest\_dir** (*str*) – directory to extract to.

**Returns** The extracted zip.

**Return type** `zipfile.ZipFile`

`unihan_etl.process.files_exist(path, files)`  
 Return True if all files exist in specified path.

`unihan_etl.process.filter_manifest(files)`  
 Return filtered UNIHAN\_MANIFEST from list of file names.

`unihan_etl.process.get_fields(d)`  
 Return list of fields from dict of {filename: ['field', 'field1']}.

`unihan_etl.process.get_parser()`  
 Return `argparse.ArgumentParser` instance for CLI.

**Returns** argument parser for CLI use.

**Return type** `argparse.ArgumentParser`

`unihan_etl.process.has_valid_zip(zip_path)`

Return True if valid zip exists.

**Parameters** `zip_path(str)` – absolute path to zip

**Returns** True if valid zip exists at path

**Return type** `bool`

`unihan_etl.process.in_fields(c, fields)`

Return True if string is in the default fields.

`unihan_etl.process.listify(data, fields)`

Convert tabularized data to a CSV-friendly list.

**Parameters**

- **data** (*list of dict*) –
- **params** (*list of str*) – keys/columns, e.g. ['kDictionary']

`unihan_etl.process.load_data(files)`

Extract zip and process information into CSV's.

**Parameters** `files(list of str)` –

**Returns** combined data from files

**Return type** `str`

`unihan_etl.process.normalize(raw_data, fields)`

Return normalized data from a UNIHAN data files.

**Parameters**

- **raw\_data** (*str*) – combined text files from UNIHAN
- **fields** (*list of str*) – list of columns to pull

**Returns** list of unihan character information

**Return type** `list`

`unihan_etl.process.not_junk(line)`

Return False on newlines and C-style comments.

`unihan_etl.process.setup_logger(logger=None, level='DEBUG')`

Setup logging for CLI use.

**Parameters**

- **logger** (`Logger`) – instance of logger
- **level** (*str*) – logging level, e.g. 'DEBUG'

`unihan_etl.process.zip_has_files(files, zip_file)`

Return True if zip has the files inside.

**Parameters**

- **files** (*list of str*) – files inside zip file
- **zip\_file** (`zipfile.ZipFile`) –

**Returns** True if files inside of `:py:meth: 'zipfile.ZipFile.namelist()`

**Return type** `bool`

### 5.4.1 Constants

`unihan_etl.constants.CUSTOM_DELIMITED_FIELDS = ('kDefinition', 'kDaeJaweon', 'kHDZRadBreak`  
 FIELDS with multiple values via custom delimiters

`unihan_etl.constants.INDEX_FIELDS = ('ucn', 'char')`  
 Default index fields for unihan csv's. You probably want these.

`unihan_etl.constants.SPACE_DELIMITED_DICT_FIELDS = ('kHanYu', 'kXHC1983', 'kMandarin', 'kT`  
 Fields with multiple values UNIHAN delimits by spaces -> dict

`unihan_etl.constants.SPACE_DELIMITED_FIELDS = ('kAccountingNumeric', 'kCantonese', 'kCCCI`  
 Any space delimited field regardless of expanded form

`unihan_etl.constants.SPACE_DELIMITED_LIST_FIELDS = ('kAccountingNumeric', 'kCantonese', 'I`  
 Fields with multiple values UNIHAN delimits by spaces -> list

`unihan_etl.constants.UNIHAN_MANIFEST = {'Unihan_DictionaryIndices.txt': ('kCheungBauerInde`  
 Dictionary of tuples mapping locations of files to fields

### 5.4.2 Expansion

Functions to uncompact details inside field values.

#### Notes

`re.compile()` operations are inside of expand functions:

1. readability
2. module-level function bytecode is cached in python
3. the last used compiled regexes are cached

`unihan_etl.expansion.N_DIACRITICS = 'ㄹㄺ'`  
 diacritics from kHanyuPinlu

`unihan_etl.expansion.expand_field(field, fvalue)`  
 Return structured value of information in UNIHAN field.

#### Parameters

- **field** (`str`) – field name
- **fvalue** (`str`) – value of field

**Returns** expanded field information per UNIHAN's documentation

**Return type** `list` or `dict`

### 5.4.3 Utilities and test helpers

Utility and helper methods for script.

## util

`unihan_etl.util.ucn_to_unicode(ucn)`

Return a python unicode value from a UCN.

Converts a Unicode Universal Character Number (e.g. “U+4E00” or “4E00”) to Python unicode (u’u4e00’)

`unihan_etl.util.ucnstring_to_python(ucn_string)`

Return string with Unicode UCN (e.g. “U+4E00”) to native Python Unicode (u’u4e00’).

`unihan_etl.util.ucnstring_to_unicode(ucn_string)`

Return ucnstring as Unicode.

Test helpers functions for downloading and processing Unihan data.

`unihan_etl.test.assert_dict_contains_subset(subset, dictionary, msg=None)`

Ported assertion for dict subsets in py.test.

### Parameters

- **subset** (*dict*) – needle
- **dictionary** (*dict*) – haystack
- **msg** (*str*, *optional*) – message display if assertion fails

## 5.5 Frequently Asked Questions

... **Why are some fields, e.g. *kTotalStrokes*, in lists when there’s seemingly not any multi-value data?** The word back from the developers of UNIHAN is they keep some fields multi-valued for future use.

Apparently at the moment there is only one record with two values for the *kTotalStrokes* field in the Unihan database. However, the maintainers of the data intend to populate the *kTotalStrokes* field as needed in the future, and as documented in UAX #38.

May 30, 2017 (Unicode 9.0)

unihan-etl is designed to handle fields correctly and consistently according to the documentation in the database.

## 5.6 History

- Fix flicker in download progress bar
- Add `project_urls` to `setup.py`
- Use plain `reStructuredText` for `CHANGES`
- Use `collections` that’s compatible with python 2 and 3
- PEP8 tweaks
- Add code links in API
- Add `__version__` to `unihan_etl`
- #91 New fields from UNIHAN [Revision 25](#).
  - `kJinmeiyoKanji`
  - `kJoyoKanji`
  - `kKoreanEducationHanja`



- kKoreanName
- kTGH

UNIHAN Revision 25 was released 2018-05-18 and issued for Unicode **11.0**:

- Add tests and example corpus for kCCCII
- Add configuration / make tests for isort, flake8
- Switch tmuxp config to use pipenv
- Add *Pipfile*
- Add `make sync_pipfile` task to sync *requirements/\*.txt* files with *Pipfile*
- Update and sync Pipfile
- Developer package updates (linting / docs / testing)
  - isort 4.2.15 to 4.3.4
  - flake8 3.3.0 to 3.5.0
  - vulture 0.14 to 0.27
  - sphinx 1.6.2 to 1.7.6
  - alagitpull 0.0.12 to 0.0.21
  - releases 1.3.1 to 1.6.1
  - sphinx-argparse 0.2.1 to 1.6.2
  - pytest 3.1.2 to 3.6.4
- Move documentation over to numpy-style
- Add sphinxcontrib-napoleon 0.6.1
- Update LICENSE New BSD to MIT
- All future commits and contributions are licensed to the *cihai software foundation*. This includes commits by Tony Narlock (creator).
- Enhance support for locations on *kHDZRadBreak* fields.
- Fix kIRG\_GSource without location
- Fix kFenn output
- Fix kHanyuPinlu support output for n diacritics
- Add expansion for kIRGKangXi
- Normalize Radical-Stroke expansion for kRSUnicode
- Migrate more fields to regular expressions
- Normalize character field for kDaeJaweon, kHanyuPinyin, and kCheungBauer, kFennIndex, kCheung-BauerIndex, kIICore, kIRGHanyuDaZidian
- Support for expanding kGSR
- Convert some field expansions to use regexes
- Fix bug where destination file was made into directory on first run
- Rename from unihan-tabular to unihan-etl
- Support for expanding multi-value fields

- Support for pruning empty fields
- Improve help dialog
- Added a page about UNIHAN and the project to documentation
- Split constant values into their own module
- Split functionality for expanding unstructured values into its own module
- Update to add *kJa* and adjust source file of *kCompatibilityVariant* per Unicode 8.0.0.
- Support for configuring logging via options and CLI
- Convert all print statements to use logger
- Allow for local / file system sources for Unihan.zip
- Only extract zip if unextracted
- Update package classifiers
- Add back datapackage
- Fix python 2 CSV output
- Default to CSV output
- Move unicodescv module to dependency package
- Support for XDG directory specification
- Support for custom destination output, including replacing template variable `{ext}`
- Move `__about__.py` to module level
- Fix python package import
- Fix readme bug on pypi
- Support for exporting in YAML and JSON
- More internal factoring and simplification
- Return data as list
- Drop python 3.3 and 3.4 support
- Rename from `cihaidata_unihan` to `unihan_tabular`
- Drop datapackages in favor of a universal JSON, YAML and CSV export.
- Only use `UnicodeWriter` in Python 2, fixes issue with python would encode *b* in front of values
- Rename `scripts/` to `cihaidata_unihan/`
- Enable invoking tool via `$ cihaidata_unihan`
- Major internal refactor and simplification
- Convert to `pytest assert` statements
- Convert full test suite to `pytest` functions and fixtures
- Get CLI documentation up again
- Improve test coverage
- Lint code, remove unused imports
- Switch license BSD -> MIT

- Rebooted
- Modernize *Makefile* in docs
- Add Makefile to main project
- Modernize package metadata to use `__about__.py`
- Update requirements to use *requirements/* folder for base, testing and doc dependencies.
- Update sphinx theme to alabaster with new logo.
- Update travis to use coverall
- Update links on README to use https
- Update travis to test up to python 3.6
- Add support for pypy (why not)
- Lock base dependencies
- Add dev dependencies for isort, vulture and flake8



## PYTHON MODULE INDEX

### U

- `unihan_etl`, [16](#)
- `unihan_etl.constants`, [19](#)
- `unihan_etl.expansion`, [19](#)
- `unihan_etl.process`, [16](#)
- `unihan_etl.test`, [20](#)
- `unihan_etl.util`, [19](#)



## A

ALLOWED\_EXPORT\_TYPES (in module *unihan\_etl.process*), 16  
 assert\_dict\_contains\_subset() (in module *unihan\_etl.test*), 20

## C

CUSTOM\_DELIMITED\_FIELDS (in module *unihan\_etl.constants*), 19

## D

DESTINATION\_DIR (in module *unihan\_etl.process*), 16  
 download() (in module *unihan\_etl.process*), 17  
 download() (*unihan\_etl.process.Packager* method), 16

## E

expand\_delimiters() (in module *unihan\_etl.process*), 17  
 expand\_field() (in module *unihan\_etl.expansion*), 19  
 export() (*unihan\_etl.process.Packager* method), 16  
 extract\_zip() (in module *unihan\_etl.process*), 17

## F

files\_exist() (in module *unihan\_etl.process*), 17  
 filter\_manifest() (in module *unihan\_etl.process*), 17  
 from\_cli() (*unihan\_etl.process.Packager* class method), 16

## G

get\_fields() (in module *unihan\_etl.process*), 17  
 get\_parser() (in module *unihan\_etl.process*), 17

## H

has\_valid\_zip() (in module *unihan\_etl.process*), 18

## I

in\_fields() (in module *unihan\_etl.process*), 18  
 INDEX\_FIELDS (in module *unihan\_etl.constants*), 19

## L

listify() (in module *unihan\_etl.process*), 18  
 load\_data() (in module *unihan\_etl.process*), 18

## N

N\_DIACRITICS (in module *unihan\_etl.expansion*), 19  
 normalize() (in module *unihan\_etl.process*), 18  
 not\_junk() (in module *unihan\_etl.process*), 18

## P

Packager (class in *unihan\_etl.process*), 16

## S

setup\_logger() (in module *unihan\_etl.process*), 18  
 SPACE\_DELIMITED\_DICT\_FIELDS (in module *unihan\_etl.constants*), 19  
 SPACE\_DELIMITED\_FIELDS (in module *unihan\_etl.constants*), 19  
 SPACE\_DELIMITED\_LIST\_FIELDS (in module *unihan\_etl.constants*), 19

## U

ucn\_to\_unicode() (in module *unihan\_etl.util*), 20  
 ucnstring\_to\_python() (in module *unihan\_etl.util*), 20  
 ucnstring\_to\_unicode() (in module *unihan\_etl.util*), 20  
 unihan\_etl (module), 16  
 unihan\_etl.constants (module), 19  
 unihan\_etl.expansion (module), 19  
 unihan\_etl.process (module), 16  
 unihan\_etl.test (module), 20  
 unihan\_etl.util (module), 19  
 UNIHAN\_FIELDS (in module *unihan\_etl.process*), 16  
 UNIHAN\_FILES (in module *unihan\_etl.process*), 17  
 UNIHAN\_MANIFEST (in module *unihan\_etl.constants*), 19  
 UNIHAN\_URL (in module *unihan\_etl.process*), 17  
 UNIHAN\_ZIP\_PATH (in module *unihan\_etl.process*), 17

## W

WORK\_DIR (in module *unihan\_etl.process*), 17

## Z

`zip_has_files()` (*in module unihan\_etl.process*), [18](#)