:mod:`pickletools` --- Tools for pickle developers

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 1); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 4)**
>
> Unknown directive type "module".
>
> ```
> .. module:: pickletools
>    :synopsis: Contains extensive comments about the pickle protocols and
>               pickle-machine opcodes, as well as some useful functions.
> ```

Source code: :source:`Lib/pickletools.py`

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 8); *backlink***
>
> Unknown interpreted text role "source".

This module contains various constants relating to the intimate details of the :mod:`pickle` module, some lengthy comments about the implementation, and a few useful functions for analyzing pickled data. The contents of this module are useful for Python core developers who are working on the :mod:`pickle`; ordinary users of the :mod:`pickle` module probably won't find the :mod:`pickletools` module relevant.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 13); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 13); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 13); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 13); *backlink***
>
> Unknown interpreted text role "mod".

Command line usage

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)pickletools.rst`, line 23)**
>
> Unknown directive type "versionadded".
>
> ```
> .. versionadded:: 3.2
> ```

When invoked from the command line, `python -m pickletools` will disassemble the contents of one or more pickle files. Note that if you want to see the Python object stored in the pickle rather than the details of pickle format, you may want to use `-m pickle` instead. However, when the pickle file that you want to examine comes from an untrusted source, `-m pickletools` is a safer option because it does not execute pickle bytecode.

For example, with a tuple `(1, 2)` pickled in file `x.pickle`:

```
$ python -m pickle x.pickle
(1, 2)

$ python -m pickletools x.pickle
    0: \x80 PROTO      3
    2: K    BININT1    1
    4: K    BININT1    2
    6: \x86 TUPLE2
    7: q    BINPUT     0
    9: .    STOP
highest protocol among opcodes = 2
```

Command line options

Programmatic Interface

```
.. function:: dis(pickle, out=None, memo=None, indentlevel=4, annotate=0)

   Outputs a symbolic disassembly of the pickle to the file-like
   object *out*, defaulting to ``sys.stdout``.  *pickle* can be a
   string or a file-like object.  *memo* can be a Python dictionary
   that will be used as the pickle's memo; it can be used to perform
   disassemblies across multiple pickles created by the same
   pickler. Successive levels, indicated by ``MARK`` opcodes in the
   stream, are indented by *indentlevel* spaces.  If a nonzero value
   is given to *annotate*, each opcode in the output is annotated with
   a short description.  The value of *annotate* is used as a hint for
   the column where annotation should start.

   .. versionadded:: 3.2
      The *annotate* argument.
```

```
.. function:: genops(pickle)

   Provides an :term:`iterator` over all of the opcodes in a pickle, returning a
   sequence of ``(opcode, arg, pos)`` triples.  *opcode* is an instance of an
   :class:`OpcodeInfo` class; *arg* is the decoded value, as a Python object, of
   the opcode's argument; *pos* is the position at which this opcode is located.
   *pickle* can be a string or a file-like object.
```

```
.. function:: optimize(picklestring)

   Returns a new equivalent pickle string after eliminating unused ``PUT``
   opcodes. The optimized pickle is shorter, takes less transmission time,
   requires less storage space, and unpickles more efficiently.
```