```
usage: isort [-h] [--src SRC_PATHS] [-a ADD_IMPORTS] [--append] [--ac] [--af]
             [-b KNOWN_STANDARD_LIBRARY]
             [--extra-builtin EXTRA_STANDARD_LIBRARY] [-c] [--ca] [--cs] [-d]
             [--df] [--ds] [-e] [-f KNOWN_FUTURE_LIBRARY] [--fas] [--fass]
             [--ff FROM_FIRST] [--fgw [FORCE_GRID_WRAP]] [--fss] [-i INDENT]
             [-j JOBS] [--lai LINES_AFTER_IMPORTS] [--lbt LINES_BETWEEN_TYPES]
             [--le LINE_ENDING] [--ls] [--lss]
             [-m
{GRID,VERTICAL,HANGING_INDENT,VERTICAL_HANGING_INDENT,VERTICAL_GRID,VERTICAL_GRID_GROUPED,VERT
ICAL_GRID_GROUPED_NO_COMMA,NOQA,VERTICAL_HANGING_INDENT_BRACKET,VERTICAL_PREFIX_FROM_MODU
LE_IMPORT,HANGING_INDENT_WITH_PARENTHESES,BACKSLASH_GRID,0,1,2,3,4,5,6,7,8,9,10,11}]
             [-n] [--nis] [--nlb NO_LINES_BEFORE] [-o KNOWN_THIRD_PARTY]
             [--ot] [--dt] [-p KNOWN_FIRST_PARTY]
             [--known-local-folder KNOWN_LOCAL_FOLDER] [-q]
             [--rm REMOVE_IMPORTS] [--rr] [-s SKIP] [--sd DEFAULT_SECTION]
             [--sg SKIP_GLOB] [--gitignore] [--sl]
             [--nsl SINGLE_LINE_EXCLUSIONS] [--sp SETTINGS_PATH]
             [-t FORCE_TO_TOP] [--tc] [--up] [-V] [-v]
             [--virtual-env VIRTUAL_ENV] [--conda-env CONDA_ENV] [--vn]
             [-l LINE_LENGTH] [--wl WRAP_LENGTH] [--ws] [--case-sensitive]
             [--filter-files] [--py {all,2,27,3,35,36,37,38,39,auto}]
             [--profile PROFILE] [--interactive] [--old-finders]
             [--show-config] [--show-files] [--honor-noqa]
             [--remove-redundant-aliases] [--color] [--float-to-top]
             [--treat-comment-as-code TREAT_COMMENTS_AS_CODE]
             [--treat-all-comment-as-code] [--formatter FORMATTER]
             [--ext SUPPORTED_EXTENSIONS]
             [--blocked-extension BLOCKED_EXTENSIONS] [--dedup-headings]
             [--only-sections] [--only-modified]
             [files [files ...]]

Sort Python import definitions alphabetically within logical sections. Run
with no arguments to see a quick start guide, otherwise, one or more
files/directories/stdin must be provided. Use `-` as the first argument to
represent stdin. Use --interactive to use the pre 5.0.0 interactive behavior.
If you've used isort 4 but are new to isort 5, see the upgrading
guide:https://pycqa.github.io/isort/docs/upgrade_guides/5.0.0/.

positional arguments:
  files                 One or more Python source files that need their
                        imports sorted.

optional arguments:
  -h, --help            show this help message and exit
```

```
--src SRC_PATHS, --src-path SRC_PATHS
                        Add an explicitly defined source path (modules within
                        src paths have their imports automatically categorized
                        as first_party).
-a ADD_IMPORTS, --add-import ADD_IMPORTS
                        Adds the specified import line to all files,
                        automatically determining correct placement.
--append, --append-only
                        Only adds the imports specified in --add-imports if
                        the file contains existing imports.
--ac, --atomic        Ensures the output doesn't save if the resulting file
                        contains syntax errors.
--af, --force-adds    Forces import adds even if the original file is empty.
-b KNOWN_STANDARD_LIBRARY, --builtin KNOWN_STANDARD_LIBRARY
                        Force isort to recognize a module as part of Python's
                        standard library.
--extra-builtin EXTRA_STANDARD_LIBRARY
                        Extra modules to be included in the list of ones in
                        Python's standard library.
-c, --check-only, --check
                        Checks the file for unsorted / unformatted imports and
                        prints them to the command line without modifying the
                        file.
--ca, --combine-as    Combines as imports on the same line.
--cs, --combine-star  Ensures that if a star import is present, nothing else
                        is imported from that namespace.
-d, --stdout          Force resulting output to stdout, instead of in-place.
--df, --diff          Prints a diff of all the changes isort would make to a
                        file, instead of changing it in place
--ds, --no-sections   Put all imports into the same section bucket
-e, --balanced        Balances wrapping to produce the most consistent line
                        length possible
-f KNOWN_FUTURE_LIBRARY, --future KNOWN_FUTURE_LIBRARY
                        Force isort to recognize a module as part of Python's
                        internal future compatibility libraries. WARNING: this
                        overrides the behavior of __future__ handling and
                        therefore can result in code that can't execute. If
                        you're looking to add dependencies such as six a
                        better option is to create a another section below
                        --future using custom sections. See:
                        https://github.com/PyCQA/isort#custom-sections-and-
                        ordering and the discussion here:
                        https://github.com/PyCQA/isort/issues/1463.
--fas, --force-alphabetical-sort
```

```
                    Force all imports to be sorted as a single section
  --fass, --force-alphabetical-sort-within-sections
                    Force all imports to be sorted alphabetically within a
                    section
  --ff FROM_FIRST, --from-first FROM_FIRST
                    Switches the typical ordering preference, showing from
                    imports first then straight ones.
  --fgw [FORCE_GRID_WRAP], --force-grid-wrap [FORCE_GRID_WRAP]
                    Force number of from imports (defaults to 2 when
                    passed as CLI flag without value)to be grid wrapped
                    regardless of line length. If 0 is passed in (the
                    global default) only line length is considered.
  --fss, --force-sort-within-sections
                    Don't sort straight-style imports (like import sys)
                    before from-style imports (like from itertools import
                    groupby). Instead, sort the imports by module,
                    independent of import style.
  -i INDENT, --indent INDENT
                    String to place for indents defaults to "    " (4
                    spaces).
  -j JOBS, --jobs JOBS  Number of files to process in parallel.
  --lai LINES_AFTER_IMPORTS, --lines-after-imports LINES_AFTER_IMPORTS
  --lbt LINES_BETWEEN_TYPES, --lines-between-types LINES_BETWEEN_TYPES
  --le LINE_ENDING, --line-ending LINE_ENDING
                    Forces line endings to the specified value. If not
                    set, values will be guessed per-file.
  --ls, --length-sort   Sort imports by their string length.
  --lss, --length-sort-straight
                    Sort straight imports by their string length. Similar
                    to `length_sort` but applies only to straight imports
                    and doesn't affect from imports.
  -m
{GRID,VERTICAL,HANGING_INDENT,VERTICAL_HANGING_INDENT,VERTICAL_GRID,VERTICAL_GRID_GROUPED,VERT
ICAL_GRID_GROUPED_NO_COMMA,NOQA,VERTICAL_HANGING_INDENT_BRACKET,VERTICAL_PREFIX_FROM_MODU
LE_IMPORT,HANGING_INDENT_WITH_PARENTHESES,BACKSLASH_GRID,0,1,2,3,4,5,6,7,8,9,10,11}, --multi-line
{GRID,VERTICAL,HANGING_INDENT,VERTICAL_HANGING_INDENT,VERTICAL_GRID,VERTICAL_GRID_GROUPED,VERT
ICAL_GRID_GROUPED_NO_COMMA,NOQA,VERTICAL_HANGING_INDENT_BRACKET,VERTICAL_PREFIX_FROM_MODU
LE_IMPORT,HANGING_INDENT_WITH_PARENTHESES,BACKSLASH_GRID,0,1,2,3,4,5,6,7,8,9,10,11}
                    Multi line output (0-grid, 1-vertical, 2-hanging,
                    3-vert-hanging, 4-vert-grid, 5-vert-grid-grouped,
                    6-vert-grid-grouped-no-comma, 7-noqa, 8-vertical-
                    hanging-indent-bracket, 9-vertical-prefix-from-module-
                    import, 10-hanging-indent-with-parentheses).
  -n, --ensure-newline-before-comments
```

```
                    Inserts a blank line before a comment following an
                    import.
--nis, --no-inline-sort
                    Leaves `from` imports with multiple imports 'as-is'
                    (e.g. `from foo import a, c ,b`).
--nlb NO_LINES_BEFORE, --no-lines-before NO_LINES_BEFORE
                    Sections which should not be split with previous by
                    empty lines
-o KNOWN_THIRD_PARTY, --thirdparty KNOWN_THIRD_PARTY
                    Force isort to recognize a module as being part of a
                    third party library.
--ot, --order-by-type
                    Order imports by type, which is determined by case, in
                    addition to alphabetically. **NOTE**: type here refers
                    to the implied type from the import name
                    capitalization. isort does not do type introspection
                    for the imports. These "types" are simply:
                    CONSTANT_VARIABLE, CamelCaseClass,
                    variable_or_function. If your project follows PEP8 or
                    a related coding standard and has many imports this is
                    a good default, otherwise you likely will want to turn
                    it off. From the CLI the `--dont-order-by-type` option
                    will turn this off.
--dt, --dont-order-by-type
                    Don't order imports by type, which is determined by
                    case, in addition to alphabetically. **NOTE**: type
                    here refers to the implied type from the import name
                    capitalization. isort does not do type introspection
                    for the imports. These "types" are simply:
                    CONSTANT_VARIABLE, CamelCaseClass,
                    variable_or_function. If your project follows PEP8 or
                    a related coding standard and has many imports this is
                    a good default. You can turn this on from the CLI
                    using `--order-by-type`.
-p KNOWN_FIRST_PARTY, --project KNOWN_FIRST_PARTY
                    Force isort to recognize a module as being part of the
                    current python project.
--known-local-folder KNOWN_LOCAL_FOLDER
                    Force isort to recognize a module as being a local
                    folder. Generally, this is reserved for relative
                    imports (from . import module).
-q, --quiet         Shows extra quiet output, only errors are outputted.
--rm REMOVE_IMPORTS, --remove-import REMOVE_IMPORTS
                    Removes the specified import from all files.
```

```
--rr, --reverse-relative
                  Reverse order of relative imports.
-s SKIP, --skip SKIP  Files that sort imports should skip over. If you want
                  to skip multiple files you should specify twice:
                  --skip file1 --skip file2.
--sd DEFAULT_SECTION, --section-default DEFAULT_SECTION
                  Sets the default section for import options:
                  ('FUTURE', 'STDLIB', 'THIRDPARTY', 'FIRSTPARTY',
                  'LOCALFOLDER')
--sg SKIP_GLOB, --skip-glob SKIP_GLOB
                  Files that sort imports should skip over.
--gitignore, --skip-gitignore
                  Treat project as a git repository and ignore files
                  listed in .gitignore
--sl, --force-single-line-imports
                  Forces all from imports to appear on their own line
--nsl SINGLE_LINE_EXCLUSIONS, --single-line-exclusions SINGLE_LINE_EXCLUSIONS
                  One or more modules to exclude from the single line
                  rule.
--sp SETTINGS_PATH, --settings-path SETTINGS_PATH, --settings-file SETTINGS_PATH, --settings SETTINGS_PATH
                  Explicitly set the settings path or file instead of
                  auto determining based on file location.
-t FORCE_TO_TOP, --top FORCE_TO_TOP
                  Force specific imports to the top of their appropriate
                  section.
--tc, --trailing-comma
                  Includes a trailing comma on multi line imports that
                  include parentheses.
--up, --use-parentheses
                  Use parentheses for line continuation on length limit
                  instead of slashes. **NOTE**: This is separate from
                  wrap modes, and only affects how individual lines that
                  are too long get continued, not sections of multiple
                  imports.
-V, --version         Displays the currently installed version of isort.
-v, --verbose         Shows verbose output, such as when files are skipped
                  or when a check is successful.
--virtual-env VIRTUAL_ENV
                  Virtual environment to use for determining whether a
                  package is third-party
--conda-env CONDA_ENV
                  Conda environment to use for determining whether a
                  package is third-party
--vn, --version-number
```

```
                        Returns just the current version number without the
                        logo
-l LINE_LENGTH, -w LINE_LENGTH, --line-length LINE_LENGTH, --line-width LINE_LENGTH
                        The max length of an import line (used for wrapping
                        long imports).
--wl WRAP_LENGTH, --wrap-length WRAP_LENGTH
                        Specifies how long lines that are wrapped should be,
                        if not set line_length is used. NOTE: wrap_length must
                        be LOWER than or equal to line_length.
--ws, --ignore-whitespace
                        Tells isort to ignore whitespace differences when
                        --check-only is being used.
--case-sensitive      Tells isort to include casing when sorting module
                        names
--filter-files        Tells isort to filter files even when they are
                        explicitly passed in as part of the CLI command.
--py {all,2,27,3,35,36,37,38,39,auto}, --python-version {all,2,27,3,35,36,37,38,39,auto}
                        Tells isort to set the known standard library based on
                        the specified Python version. Default is to assume any
                        Python 3 version could be the target, and use a union
                        of all stdlib modules across versions. If auto is
                        specified, the version of the interpreter used to run
                        isort (currently: 38) will be used.
--profile PROFILE     Base profile type to use for configuration. Profiles
                        include: black, django, pycharm, google, open_stack,
                        plone, attrs, hug. As well as any shared profiles.
--interactive         Tells isort to apply changes interactively.
--old-finders, --magic-placement
                        Use the old deprecated finder logic that relies on
                        environment introspection magic.
--show-config         See isort's determined config, as well as sources of
                        config options.
--show-files          See the files isort will be ran against with the
                        current config options.
--honor-noqa          Tells isort to honor noqa comments to enforce skipping
                        those comments.
--remove-redundant-aliases
                        Tells isort to remove redundant aliases from imports,
                        such as `import os as os`. This defaults to `False`
                        simply because some projects use these seemingly
                        useless aliases to signify intent and change
                        behaviour.
--color               Tells isort to use color in terminal output.
--float-to-top        Causes all non-indented imports to float to the top of
```

the file having its imports sorted (immediately below the top of file comment). This can be an excellent shortcut for collecting imports every once in a while when you place them in the middle of a file to avoid context switching. *NOTE*: It currently doesn't work with cimports and introduces some extra over-head and a performance penalty.

--treat-comment-as-code TREAT_COMMENTS_AS_CODE

Tells isort to treat the specified single line comment(s) as if they are code.

--treat-all-comment-as-code

Tells isort to treat all single line comments as if they are code.

--formatter FORMATTER

Specifies the name of a formatting plugin to use when producing output.

--ext SUPPORTED_EXTENSIONS, --extension SUPPORTED_EXTENSIONS, --supported-extension SUPPORTED_EXTENSIONS

Specifies what extensions isort can be ran against.

--blocked-extension BLOCKED_EXTENSIONS

Specifies what extensions isort can never be ran against.

--dedup-headings **Tells isort to only show an identical custom import heading comment once, even if there are multiple sections with the comment set.**

--only-sections, --os

Causes imports to be sorted only based on their sections like STDLIB,THIRDPARTY etc. Imports are unaltered and keep their relative positions within the different sections.

--only-modified, --om

Suppresses verbose output for non-modified files.