```python
"""The helpers module provides functions useful to the Activity class methods.
Functions specific to parsing Garmin activity files reside in garmin_helpers.
"""

from datetime import datetime, timedelta, date


def td_cvt(duration: timedelta) -> tuple:
    """Return number of hours, minutes and seconds represented by a timedelta
    object."""

    # A timedelta objects is composed of days and seconds.  In oder to
    # represent cumulative exerise periods, which can exceed a day, we
    # use this function to convert the object into hours, minutes and
    # seconds to facilitate output.

    # A day has 86,400 day_seconds
    # An hour has 3,600 seconds
    # A minute has 60 seconds

    day_seconds = duration.days * 86400
    total_seconds = day_seconds + duration.seconds
    whole_hours = total_seconds // 3600
    adjusted_total_seconds = total_seconds % 3600
    whole_minutes = adjusted_total_seconds // 60
    remaining_seconds = adjusted_total_seconds % 60

    return (whole_hours, whole_minutes, remaining_seconds)


def is_date(string: str) -> bool:
    """ Returns True if the string is a valid date of the form YYYY-MM-DD."""

    if not isinstance(string, str):
        raise TypeError("string must be a string")

    format = "%Y-%m-%d"
    try:
        datetime.strptime(string, format).date()
        return True
    except ValueError:
        return False


def parse_date(string: str) -> date:
    """Parse a string of the form YYYY-MM-DD and return the corresponding
    datetime object."""

    if not isinstance(string, str):
        raise TypeError("string must be a string")

    format = "%Y-%m-%d"
    return datetime.strptime(string, format).date()


def none_factory() -> None:
    """Return None."""

    # py_athletics uses pickle to save state and pickle cannot save lambda
    # expressions, so we use this function instead of a lambda expression in
    # default dicts as the default factory.

    return None
```