```python
import os
from pathlib import Path
from zipfile import ZipFile

from src.settings import CWD, DATA_DIR, TEST_DIR
from src.settings import REGISTRY_INPUT_DIR, REGISTRY_OUTPUT_DIR, SUPPLEMENTAL_INPUT_DIR, SUPPLEMENTAL_OUTPUT_DIR


def get_data_dir_path():
    """
    Builds path to aircraft registry data directory

    :return: path to aircraft registry data directory
    """

    path = os.path.join(CWD, DATA_DIR)

    return path


def get_test_dir_path():
    """
    Builds path to aircraft registry test directory

    :return: path to aircrat registry test directory
    """

    path = os.path.join(CWD, TEST_DIR)

    return path


def get_registry_dir_path(is_test=False):
    """
    Determines which aircraft registry directory serves as the root for registry-related paths given the session context

    :param (bool) is_test: if 'True' the session is in a testing-context
    :return: root directory for registry-related paths
    """

    if not is_test:
        path = get_data_dir_path()
    else:
        path = get_test_dir_path()

    return path


def get_registry_inputdir_path(is_test=False):
    """
    Builds path to input registry data or test directory

    :param (bool) is_test: if 'True' the session is in a testing-context
    :return: path to input registry data or test directory
    """

    path = os.path.join(get_registry_dir_path(is_test), REGISTRY_INPUT_DIR)

    return path


def get_registry_outputdir_path(is_test=False):
    """
    Builds path to output registry data or test directory

    :param (bool) is_test: if 'True' the session is in a testing-context
    :return: path to output registry data or test directory
    """

    path = os.path.join(get_registry_dir_path(is_test), REGISTRY_OUTPUT_DIR)

    return path


def get_supplemental_inputdir_path(is_test=False):
    """
    Builds path to input supplemental data or test directory

    :param (bool) is_test: if 'True' the session is in a testing-context
    :return: path to input supplemental data or test directory
    """

    path = os.path.join(get_registry_dir_path(is_test), SUPPLEMENTAL_INPUT_DIR)

    return path


def get_supplemental_outputdir_path(is_test=False):
    """
    Builds path to output supplemental data or test directory

    :param (bool) is_test: if 'True' the session is in a testing-context
    :return: path to output supplemental data or test directory
    """

    path = os.path.join(get_registry_dir_path(is_test), SUPPLEMENTAL_OUTPUT_DIR)

    return path


def make_dir(path):
    """
    Initializes new directory if it doesn't exist

    :param (str) path: path of directory to initialize
    :return: effect - creates directory specified by 'path' argument
    """

    Path(path).mkdir(exist_ok=True)


def get_registry_codes():
    """
    Builds list of registry-codes for remote registry data sources whose ETL components have been implemented

    :return: list of uppercased registry-codes for implemented remote registry data sources
    """

    registry_codes = []

    for child in os.listdir(os.path.join(CWD, "registries")):
        if not child.endswith(".py") and child != "__pycache__":
            registry_codes.append(child.upper())

    return registry_codes


def init_registry_dirs(is_test=False):
    """
    Creates all input/output directories within the parent data or test directory, and all input/output sub-directories
    for each remote registry source

    :param (bool) is_test: if 'True' the session is in a testing-context
    :return: effect - creates [DATA_DIR|TEST_DIR]/[REGISTRY_INPUT_DIR]/[registry_code] directory for each remote source;
        effect - creates [DATA_DIR|TEST_DIR]/[REGISTRY_OUTPUT_DIR]/[registry_code] directory for each remote source
    """

    registry_input_path = get_registry_inputdir_path(is_test)
    registry_output_path = get_registry_outputdir_path(is_test)

    # initialize parent input/output registry data or test directory
    make_dir(registry_input_path)
    make_dir(registry_output_path)

    for registry_code in get_registry_codes():
        # initialize input/output registry data or test sub-directories
        make_dir(os.path.join(registry_input_path, registry_code.lower()))
        make_dir(os.path.join(registry_output_path, registry_code.lower()))

    # initialize parent input/output supplemental data or registry directory
    make_dir(get_supplemental_inputdir_path(is_test))
    make_dir(get_supplemental_outputdir_path(is_test))


def unzip(path):
    """
    Extracts zipped file into a new directory with the same name

    :param (str) path: path to zipped file to be extracted
    :return: effect - creates new directory adjacent to 'path' argument
    """

    with ZipFile(path, "r") as file:
        file.extractall(path[:-4])
```