```
Last login: Fri Feb 11 15:22:54 on ttys000
(base) bodiyang@BYang-MBP ~ % ls
Applications     Documents        Library          Music
Public
Desktop          Downloads        Movies           Pictures
(base) bodiyang@BYang-MBP ~ % cd Desktop
(base) bodiyang@BYang-MBP Desktop % cd taxcalc
(base) bodiyang@BYang-MBP taxcalc % ls
Tax-Calculator
(base) bodiyang@BYang-MBP taxcalc % cd Tax-Calculator
(base) bodiyang@BYang-MBP Tax-Calculator % ls
MANIFEST.in
Makefile
PSL_catalog.json
README.md
codecov.yml
conda.recipe
csv_show.sh
csv_vars.sh
df-20-#-tmp07q5r046-#-atr.html
df-20-#-tmp07q5r046-#-mtr.html
df-20-#-tmp07q5r046-#-pch.html
df-20-#-tmp0ksi86zu-#-atr.html
df-20-#-tmp0ksi86zu-#-mtr.html
df-20-#-tmp0ksi86zu-#-pch.html
df-20-#-tmp7clk1j4p-#-atr.html
df-20-#-tmp7clk1j4p-#-mtr.html
df-20-#-tmp7clk1j4p-#-pch.html
df-20-#-tmpabal1lbo-#-atr.html
df-20-#-tmpabal1lbo-#-mtr.html
df-20-#-tmpabal1lbo-#-pch.html
df-20-#-tmpgeb0bv6x-#-atr.html
df-20-#-tmpgeb0bv6x-#-mtr.html
df-20-#-tmpgeb0bv6x-#-pch.html
df-20-#-tmpk4y87wel-#-atr.html
df-20-#-tmpk4y87wel-#-mtr.html
df-20-#-tmpk4y87wel-#-pch.html
df-20-#-tmpk7uerrb1-#-atr.html
df-20-#-tmpk7uerrb1-#-mtr.html
df-20-#-tmpk7uerrb1-#-pch.html
df-20-#-tmpm11alx9o-#-atr.html
df-20-#-tmpm11alx9o-#-mtr.html
df-20-#-tmpm11alx9o-#-pch.html
df-20-#-tmpp3ypqnqo-#-atr.html
df-20-#-tmpp3ypqnqo-#-mtr.html
df-20-#-tmpp3ypqnqo-#-pch.html
df-20-#-tmppoc185mv-#-atr.html
df-20-#-tmppoc185mv-#-mtr.html
df-20-#-tmppoc185mv-#-pch.html
df-20-#-tmprbz06szz-#-atr.html
```

```
df-20-#-tmprbz06szz-#-mtr.html
df-20-#-tmprbz06szz-#-pch.html
df-20-#-tmpvjkd3x00-#-atr.html
df-20-#-tmpvjkd3x00-#-mtr.html
df-20-#-tmpvjkd3x00-#-pch.html
df-21-#-tmp07q5r046+tmp07q5r046-tmpgebzr2k0-doc.text
df-21-#-tmp07q5r046-tmpgebzr2k0.db
df-21-#-tmp0ksi86zu+tmp0ksi86zu-tmphyykqp5a-doc.text
df-21-#-tmp0ksi86zu-tmphyykqp5a.db
df-21-#-tmp7clk1j4p+tmp7clk1j4p-tmpyq8loxzm-doc.text
df-21-#-tmp7clk1j4p-tmpyq8loxzm.db
df-21-#-tmpabal1lbo+tmpabal1lbo-tmpa9pb0ykd-doc.text
df-21-#-tmpabal1lbo-tmpa9pb0ykd.db
df-21-#-tmpgeb0bv6x+tmpgeb0bv6x-tmpi49wnngo-doc.text
df-21-#-tmpgeb0bv6x-tmpi49wnngo.db
df-21-#-tmpk4y87wel+tmpk4y87wel-tmp33xds3v3-doc.text
df-21-#-tmpk4y87wel-tmp33xds3v3.db
df-21-#-tmpk7uerrb1+tmpk7uerrb1-tmpyly9pgti-doc.text
df-21-#-tmpk7uerrb1-tmpyly9pgti.db
df-21-#-tmpm11alx9o+tmpm11alx9o-tmpva9_wauz-doc.text
df-21-#-tmpm11alx9o-tmpva9_wauz.db
df-21-#-tmpp3ypqnqo+tmpp3ypqnqo-tmp4wb1g7wv-doc.text
df-21-#-tmpp3ypqnqo-tmp4wb1g7wv.db
df-21-#-tmppoc185mv+tmppoc185mv-tmpy6tmfwat-doc.text
df-21-#-tmppoc185mv-tmpy6tmfwat.db
df-21-#-tmprbz06szz+tmprbz06szz-tmp7apkg7xn-doc.text
df-21-#-tmprbz06szz-tmp7apkg7xn.db
df-21-#-tmpvjkd3x00+tmpvjkd3x00-tmpmk6godee-doc.text
df-21-#-tmpvjkd3x00-tmpmk6godee.db
docs
environment.yml
gitpr
gitpr.bat
gitsync
gitsync.bat
new_json.py
ppp.py
pytest.ini
setup.py
taxcalc
taxcalc.egg-info
tctest-nojit.sh
(base) bodiyang@BYang-MBP Tax-Calculator % conda activate taxcalc-dev
(taxcalc-dev) bodiyang@BYang-MBP Tax-Calculator % pytest -m 'not
requires_pufcsv'
/opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-packages/
pep8.py:110: FutureWarning: Possible nested set at position 1
  EXTRANEOUS_WHITESPACE_REGEX = re.compile(r'[[({] | []}),;:]')
============================== test session starts
==============================
platform darwin -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: /Users/bodiyang/Desktop/taxcalc/Tax-Calculator, configfile:
pytest.ini, testpaths: taxcalc
plugins: xdist-2.5.0, forked-1.4.0, harvest-1.10.3, anyio-3.4.0,
pep8-1.0.6
collected 389 items / 128 deselected / 261 selected

taxcalc/tests/test_4package.py ..
[  0%]
taxcalc/tests/test_benefits.py .
[  1%]
taxcalc/tests/
test_calcfunctions.py .................................... [ 14%]
..........
[ 18%]
taxcalc/tests/test_calculator.py ................................
[ 31%]
taxcalc/tests/test_compatible_data.py .
[ 31%]
taxcalc/tests/test_consumption.py .......
[ 34%]
taxcalc/tests/test_cpscsv.py F.
[ 34%]
taxcalc/tests/test_data.py .
[ 35%]
taxcalc/tests/test_decorators.py ................
[ 41%]
taxcalc/tests/test_growdiff.py .....
[ 43%]
taxcalc/tests/test_growfactors.py ....
[ 44%]
taxcalc/tests/test_parameters.py ..................................
[ 57%]
taxcalc/tests/test_policy.py .........................................
[ 73%]
taxcalc/tests/test_records.py ....................
[ 81%]
taxcalc/tests/test_reforms.py ...FFFF
[ 83%]
taxcalc/tests/test_responses.py .
[ 84%]
taxcalc/tests/test_taxcalcio.py ...................
[ 91%]
taxcalc/tests/test_utils.py ......................
[100%]taxcalc/tests/
test_parameters.py::test_json_file_contents[policy_current_law.json]
is slower than the current benchmark by 34451.653 ms
taxcalc/tests/test_taxcalcio.py::test_creation_with_aging is slower
than the current benchmark by 1039.031 ms
```

```
taxcalc/tests/test_benefits.py::test_benefits is faster than the
current benchmark by 1223.178 ms
taxcalc/tests/test_calculator.py::test_calculator_mtr is faster than
the current benchmark by 5164.196 ms
taxcalc/tests/test_calculator.py::test_ID_StateLocal_HC_vs_CRT is
faster than the current benchmark by 1161.884 ms
taxcalc/tests/test_calculator.py::test_ID_RealEstate_HC_vs_CRT is
faster than the current benchmark by 1410.165 ms
taxcalc/tests/test_calculator.py::test_reform_documentation is faster
than the current benchmark by 1523.44 ms
taxcalc/tests/test_calculator.py::test_distribution_tables is faster
than the current benchmark by 1035.374 ms
taxcalc/tests/test_calculator.py::test_calc_all_benefits_amounts is
faster than the current benchmark by 1514.931 ms
taxcalc/tests/test_calculator.py::test_cg_top_rate is faster than the
current benchmark by 1232.443 ms
taxcalc/tests/test_consumption.py::test_consumption_response is faster
than the current benchmark by 1229.197 ms
taxcalc/tests/test_cpscsv.py::test_agg is faster than the current
benchmark by 6940.475 ms
taxcalc/tests/test_policy.py::test_reform_with_removed_parameter is
faster than the current benchmark by 1114.387 ms
taxcalc/tests/test_policy.py::test_index_offset_reform is faster than
the current benchmark by 2642.899 ms
taxcalc/tests/test_policy.py::test_cpi_offset_affect_on_prior_years is
faster than the current benchmark by 1144.013 ms
taxcalc/tests/test_policy.py::test_cpi_offset_on_reverting_params is
faster than the current benchmark by 2591.739 ms
taxcalc/tests/test_policy.py::TestAdjust::test_simple_adj is faster
than the current benchmark by 1434.929 ms
taxcalc/tests/test_policy.py::TestAdjust::test_adj_without_index_1 is
faster than the current benchmark by 1506.262 ms
taxcalc/tests/test_policy.py::TestAdjust::test_adj_without_index_2 is
faster than the current benchmark by 1511.193 ms
taxcalc/tests/test_policy.py::TestAdjust::test_activate_index is
faster than the current benchmark by 1603.624 ms
taxcalc/tests/test_policy.py::TestAdjust::test_apply_cpi_offset is
faster than the current benchmark by 1347.947 ms
taxcalc/tests/test_policy.py::TestAdjust::test_multiple_cpi_swaps is
faster than the current benchmark by 1311.996 ms
taxcalc/tests/test_policy.py::TestAdjust::test_multiple_cpi_swaps2 is
faster than the current benchmark by 1363.608 ms
taxcalc/tests/
test_policy.py::TestAdjust::test_adj_CPI_offset_and_index_status is
faster than the current benchmark by 1905.896 ms
taxcalc/tests/test_policy.py::TestAdjust::test_indexed_status_parsing
is faster than the current benchmark by 1106.878 ms
taxcalc/tests/
test_policy.py::TestAdjust::test_cpi_offset_does_not_affect_wage_index
```

```
ed_params is faster than the current benchmark by 2270.551 ms
taxcalc/tests/test_reforms.py::test_2017_law_reform is faster than the
current benchmark by 1222.397 ms
taxcalc/tests/test_reforms.py::test_round_trip_reforms[2019] is faster
than the current benchmark by 2223.396 ms
taxcalc/tests/test_reforms.py::test_round_trip_reforms[2020] is faster
than the current benchmark by 2280.202 ms
taxcalc/tests/test_reforms.py::test_round_trip_reforms[2021] is faster
than the current benchmark by 2319.504 ms
taxcalc/tests/test_reforms.py::test_round_trip_reforms[2022] is faster
than the current benchmark by 1780.048 ms
taxcalc/tests/test_reforms.py::test_round_trip_reforms[2023] is faster
than the current benchmark by 2334.469 ms
taxcalc/tests/test_reforms.py::test_reform_json_and_output is faster
than the current benchmark by 5088.071 ms
taxcalc/tests/test_taxcalcio.py::test_custom_dump_variables[\n
MARS;iitax\tpayrolltax|kombined,c00100\n    surtax\n    RECID\n
FLPDYR\n    -False-8] is faster than the current benchmark by 1075.592
ms
taxcalc/tests/test_taxcalcio.py::test_output_options is faster than
the current benchmark by 4596.778 ms
taxcalc/tests/test_taxcalcio.py::test_write_doc_file is faster than
the current benchmark by 2631.583 ms
taxcalc/tests/test_taxcalcio.py::test_no_tables_or_graphs is faster
than the current benchmark by 1159.527 ms
taxcalc/tests/test_taxcalcio.py::test_tables is faster than the
current benchmark by 1591.736 ms
taxcalc/tests/test_taxcalcio.py::test_graphs is faster than the
current benchmark by 1303.745 ms
taxcalc/tests/test_taxcalcio.py::test_analyze_warnings_print is faster
than the current benchmark by 1142.42 ms


=================================== FAILURES
===================================
___________________________________ test_agg
___________________________________

tests_path = '/Users/bodiyang/Desktop/taxcalc/Tax-Calculator/taxcalc/
tests'
cps_fullsample =         e00200  e00200p  e00200s  e00900  ...   RECID
agi_bin  pencon_p  pencon_s
0            0        0        0   ...        0
280004   69458    69458        0       0  ...  280005        9
0         0

[280005 rows x 68 columns]

    def test_agg(tests_path, cps_fullsample):
        """
        Test current-law aggregate taxes using cps.csv file.
        """
        # pylint: disable=too-many-statements,too-many-locals
        nyrs = 10
        # create a baseline Policy object with current-law policy
parameters
        baseline_policy = Policy()
        # create a Records object (rec) containing all cps.csv input
records
        recs = Records.cps_constructor(data=cps_fullsample)
        # create a Calculator object using baseline policy and cps
records
        calc = Calculator(policy=baseline_policy, records=recs)
        calc.advance_to_year(START_YEAR)
        calc_start_year = calc.current_year
        # create aggregate diagnostic table (adt) as a Pandas
DataFrame object
        adt = calc.diagnostic_table(nyrs).round(1)  # column labels
are int
        taxes_fullsample = adt.loc["Combined Liability ($b)"]
        # compare actual DataFrame, adt, with the expected DataFrame,
edt
        aggres_path = os.path.join(tests_path,
'cpscsv_agg_expect.csv')
        edt = pd.read_csv(aggres_path, index_col=False)  # column
labels are str
        edt.drop('Unnamed: 0', axis='columns', inplace=True)
        assert len(adt.columns.values) == len(edt.columns.values)
        diffs = False
        for icol in adt.columns.values:
            if not np.allclose(adt[icol], edt[str(icol)]):
                diffs = True
        if diffs:
            new_filename = '{}{}'.format(aggres_path[:-10],
'actual.csv')
            adt.to_csv(new_filename, float_format='%.1f')
            msg = 'CPSCSV AGG RESULTS DIFFER\n'
            msg += '-------------------------------------------------
\n'
            msg += '--- NEW RESULTS IN cpscsv_agg_actual.csv FILE ---
\n'
            msg += '--- if new OK, copy cpscsv_agg_actual.csv to  ---
\n'
            msg += '---                 cpscsv_agg_expect.csv     ---
\n'
            msg += '---            and rerun test.                ---
\n'
            msg += '---       (both are in taxcalc/tests)         ---
\n'
            msg += '-------------------------------------------------
```

```
\n'
>           raise ValueError(msg)
E           ValueError: CPSCSV AGG RESULTS DIFFER
E           -------------------------------------------------
E           --- NEW RESULTS IN cpscsv_agg_actual.csv FILE ---
E           --- if new OK, copy cpscsv_agg_actual.csv to  ---
E           ---                 cpscsv_agg_expect.csv     ---
E           ---            and rerun test.                ---
E           ---       (both are in taxcalc/tests)         ---
E           -------------------------------------------------

taxcalc/tests/test_cpscsv.py:63: ValueError
________________________ test_round_trip_reforms[2021]
_________________________

fyear = 2021
tests_path = '/Users/bodiyang/Desktop/taxcalc/Tax-Calculator/taxcalc/
tests'

    @pytest.mark.parametrize('fyear', [2019, 2020, 2021, 2022, 2023])
    def test_round_trip_reforms(fyear, tests_path):
        """
        Check that current-law policy has the same policy parameter
values in
        a future year as does a compound reform that first implements
the
        2017 tax law as specified in the 2017_law.json file and then
implements
        reforms that represents new tax legislation since 2017.
        This test checks that the future-year parameter values for
        current-law policy (which incorporates recent legislation such
as
        the TCJA, CARES Act, and ARPA) are the same as future-year
        parameter values for the compound round-trip reform.
        Doing this check ensures that the 2017_law.json
        and subsequent reform files that represent recent legislation
are
        specified in a consistent manner.
        """
        # pylint: disable=too-many-locals
        # create clp metadata dictionary for current-law policy in
fyear
        clp_pol = Policy()
        clp_pol.set_year(fyear)
        clp_mdata = dict(clp_pol.items())
        # create rtr metadata dictionary for round-trip reform in
fyear
        rtr_pol = Policy()
        # Revert to 2017 law
        reform_file = os.path.join(tests_path, '..', 'reforms',
'2017_law.json')
        with open(reform_file, 'r') as rfile:
            rtext = rfile.read()
        rtr_pol.implement_reform(Policy.read_json_reform(rtext))
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on TCJA
        reform_file = os.path.join(tests_path, '..', 'reforms',
'TCJA.json')
        with open(reform_file, 'r') as rfile:
            rtext = rfile.read()
        rtr_pol.implement_reform(Policy.read_json_reform(rtext))
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on the CARES Act
        rtr_pol.implement_reform(
            {'ID_Charity_crt_all': {2020: 1.0, 2021: 0.6},
             'STD_allow_charity_ded_nonitemizers': {2020: True, 2021:
False},
             'STD_charity_ded_nonitemizers_max': {2020: 300.0, 2021:
0.0}})
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on ARPA
        rtr_pol.implement_reform(
            {'RRC_c': {2021: 1400, 2022: 0},
             'RRC_ps': {2021: [75000, 150000, 75000, 112500,150000],
                        2022: [0, 0, 0, 0, 0]},
             'RRC_pe': {2021: [80000, 160000, 80000, 120000, 160000],
                        2022: [0, 0, 0, 0, 0]},
             'UI_em': {2020: 10200, 2021: 0},
             'UI_thd': {2020: [150000, 150000, 150000, 150000,
150000],
                        2021: [0, 0, 0, 0, 0]},
             'CTC_refundable': {2021: True, 2022: False},
             'CTC_include17': {2021: True, 2022: False},
             'CTC_new_c': {2021: 1000, 2022: 0},
             'CTC_new_c_under6_bonus': {2021: 600, 2022: 0},
             'CTC_new_for_all': {2021: True, 2022: False},
             'CTC_new_ps': {2021: [75000, 150000, 75000, 112500,
150000],
                            2022: [0, 0, 0, 0, 0]},
             'CTC_new_prt': {2021: 0.05, 2022: 0},
             'EITC_c': {2021: [1502.46, 3606.44, 5960.95, 6706.58],
                        2022: [546.21, 3640.7, 6017.58, 6770.29]},
             'EITC_rt': {2021: [0.153, 0.34, 0.4, 0.45],
                         2022: [0.0765, 0.34, 0.4, 0.45]},
             'EITC_ps': {2021: [11610, 19464.12, 19464.12, 19464.12],
                         2022: [8931.38, 19649.03, 19649.03,
19649.03]},
             'EITC_MinEligAge': {2021: 19, 2022: 25},
             'EITC_MaxEligAge': {2021: 125, 2022: 64},
             'EITC_InvestIncome_c': {2021: 10000},
             'EITC_sep_filers_elig': {2021: True},
             'CDCC_c': {2021: 8000, 2022: 3000},
             'CDCC_ps': {2021: 125000, 2022: 15000},
             'CDCC_ps2': {2021: 400000, 2022: 9e+99},
             'CDCC_crt': {2021: 50.0, 2022: 35.0},
             'CDCC_refundable': {2021: True, 2022: False},
             'ALD_BusinessLosses_c': {
                 2026: [283535.22, 567070.42, 283535.22, 283535.22,
567070.42],
                 2027: [9e+99, 9e+99, 9e+99, 9e+99, 9e+99]}})
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        rtr_pol.set_year(fyear)
        rtr_mdata = dict(rtr_pol.items())
        # compare fyear policy parameter values
        assert clp_mdata.keys() == rtr_mdata.keys()
        fail_dump = False
        if fail_dump:
            rtr_fails = open('fails_rtr', 'w')
            clp_fails = open('fails_clp', 'w')
        fail_params = list()
        msg = '\nRound-trip-reform and current-law-policy param values
differ for:'
        for pname in clp_mdata.keys():
            rtr_val = rtr_mdata[pname]
            clp_val = clp_mdata[pname]
            if not np.allclose(rtr_val, clp_val):
                fail_params.append(pname)
                msg += '\n  {} in {} : rtr={} clp={}'.format(
                    pname, fyear, rtr_val, clp_val
                )
                if fail_dump:
                    rtr_fails.write('{} {} {}\n'.format(pname, fyear,
rtr_val))
                    clp_fails.write('{} {} {}\n'.format(pname, fyear,
clp_val))
        if fail_dump:
            rtr_fails.close()
            clp_fails.close()
        if fail_params:
>           raise ValueError(msg)
E           ValueError:
E           Round-trip-reform and current-law-policy param values
differ for:
E             EITC_c in 2021 : rtr=[[1502.46 3606.44 5960.95 6706.58]]
clp=[[1502. 3618. 5980. 6728.]]
```

```
taxcalc/tests/test_reforms.py:181: ValueError
________________________ test_round_trip_reforms[2022]
_________________________

fyear = 2022
tests_path = '/Users/bodiyang/Desktop/taxcalc/Tax-Calculator/taxcalc/
tests'

    @pytest.mark.parametrize('fyear', [2019, 2020, 2021, 2022, 2023])
    def test_round_trip_reforms(fyear, tests_path):
        """
        Check that current-law policy has the same policy parameter
values in
        a future year as does a compound reform that first implements
the
        2017 tax law as specified in the 2017_law.json file and then
implements
        reforms that represents new tax legislation since 2017.
        This test checks that the future-year parameter values for
        current-law policy (which incorporates recent legislation such
as
        the TCJA, CARES Act, and ARPA) are the same as future-year
        parameter values for the compound round-trip reform.
        Doing this check ensures that the 2017_law.json
        and subsequent reform files that represent recent legislation
are
        specified in a consistent manner.
        """
        # pylint: disable=too-many-locals
        # create clp metadata dictionary for current-law policy in
fyear
        clp_pol = Policy()
        clp_pol.set_year(fyear)
        clp_mdata = dict(clp_pol.items())
        # create rtr metadata dictionary for round-trip reform in
fyear
        rtr_pol = Policy()
        # Revert to 2017 law
        reform_file = os.path.join(tests_path, '..', 'reforms',
'2017_law.json')
        with open(reform_file, 'r') as rfile:
            rtext = rfile.read()
        rtr_pol.implement_reform(Policy.read_json_reform(rtext))
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on TCJA
        reform_file = os.path.join(tests_path, '..', 'reforms',
'TCJA.json')
        with open(reform_file, 'r') as rfile:
            rtext = rfile.read()
        rtr_pol.implement_reform(Policy.read_json_reform(rtext))
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on the CARES Act
        rtr_pol.implement_reform(
            {'ID_Charity_crt_all': {2020: 1.0, 2021: 0.6},
             'STD_allow_charity_ded_nonitemizers': {2020: True, 2021:
False},
             'STD_charity_ded_nonitemizers_max': {2020: 300.0, 2021:
0.0}})
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on ARPA
        rtr_pol.implement_reform(
            {'RRC_c': {2021: 1400, 2022: 0},
             'RRC_ps': {2021: [75000, 150000, 75000, 112500,150000],
                        2022: [0, 0, 0, 0, 0]},
             'RRC_pe': {2021: [80000, 160000, 80000, 120000, 160000],
                        2022: [0, 0, 0, 0, 0]},
             'UI_em': {2020: 10200, 2021: 0},
             'UI_thd': {2020: [150000, 150000, 150000, 150000,
150000],
                        2021: [0, 0, 0, 0, 0]},
             'CTC_refundable': {2021: True, 2022: False},
             'CTC_include17': {2021: True, 2022: False},
             'CTC_new_c': {2021: 1000, 2022: 0},
             'CTC_new_c_under6_bonus': {2021: 600, 2022: 0},
             'CTC_new_for_all': {2021: True, 2022: False},
             'CTC_new_ps': {2021: [75000, 150000, 75000, 112500,
150000],
                            2022: [0, 0, 0, 0, 0]},
             'CTC_new_prt': {2021: 0.05, 2022: 0},
             'EITC_c': {2021: [1502.46, 3606.44, 5960.95, 6706.58],
                        2022: [546.21, 3640.7, 6017.58, 6770.29]},
             'EITC_rt': {2021: [0.153, 0.34, 0.4, 0.45],
                         2022: [0.0765, 0.34, 0.4, 0.45]},
             'EITC_ps': {2021: [11610, 19464.12, 19464.12, 19464.12],
                         2022: [8931.38, 19649.03, 19649.03,
19649.03]},
             'EITC_MinEligAge': {2021: 19, 2022: 25},
             'EITC_MaxEligAge': {2021: 125, 2022: 64},
             'EITC_InvestIncome_c': {2021: 10000},
             'EITC_sep_filers_elig': {2021: True},
             'CDCC_c': {2021: 8000, 2022: 3000},
             'CDCC_ps': {2021: 125000, 2022: 15000},
             'CDCC_ps2': {2021: 400000, 2022: 9e+99},
             'CDCC_crt': {2021: 50.0, 2022: 35.0},
             'CDCC_refundable': {2021: True, 2022: False},
             'ALD_BusinessLosses_c': {
                 2026: [283535.22, 567070.42, 283535.22, 283535.22,
567070.42],
                 2027: [9e+99, 9e+99, 9e+99, 9e+99, 9e+99]}})
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        rtr_pol.set_year(fyear)
        rtr_mdata = dict(rtr_pol.items())
        # compare fyear policy parameter values
        assert clp_mdata.keys() == rtr_mdata.keys()
        fail_dump = False
        if fail_dump:
            rtr_fails = open('fails_rtr', 'w')
            clp_fails = open('fails_clp', 'w')
        fail_params = list()
        msg = '\nRound-trip-reform and current-law-policy param values
differ for:'
        for pname in clp_mdata.keys():
            rtr_val = rtr_mdata[pname]
            clp_val = clp_mdata[pname]
            if not np.allclose(rtr_val, clp_val):
                fail_params.append(pname)
                msg += '\n  {} in {} : rtr={} clp={}'.format(
                    pname, fyear, rtr_val, clp_val
                )
                if fail_dump:
                    rtr_fails.write('{} {} {}\n'.format(pname, fyear,
rtr_val))
                    clp_fails.write('{} {} {}\n'.format(pname, fyear,
clp_val))
        if fail_dump:
            rtr_fails.close()
            clp_fails.close()
        if fail_params:
>           raise ValueError(msg)
E           ValueError:
E           Round-trip-reform and current-law-policy param values
differ for:
E             EITC_c in 2022 : rtr=[[ 546.21 3640.7  6017.58 6770.29]]
clp=[[ 560. 3733. 6064. 6935.]]
E             EITC_ps in 2022 : rtr=[[ 8931.38 19649.03 19649.03
19649.03]] clp=[[ 9160. 20130. 20130. 20130.]]
E             EITC_MaxEligAge in 2022 : rtr=[64] clp=[125]
E             EITC_InvestIncome_c in 2022 : rtr=[10169.] clp=[10300.]

taxcalc/tests/test_reforms.py:181: ValueError
________________________ test_round_trip_reforms[2023]
_________________________

fyear = 2023
tests_path = '/Users/bodiyang/Desktop/taxcalc/Tax-Calculator/taxcalc/
tests'
```

```
    @pytest.mark.parametrize('fyear', [2019, 2020, 2021, 2022, 2023])
    def test_round_trip_reforms(fyear, tests_path):
        """
        Check that current-law policy has the same policy parameter
values in
        a future year as does a compound reform that first implements
the
        2017 tax law as specified in the 2017_law.json file and then
implements
        reforms that represents new tax legislation since 2017.
        This test checks that the future-year parameter values for
        current-law policy (which incorporates recent legislation such
as
        the TCJA, CARES Act, and ARPA) are the same as future-year
        parameter values for the compound round-trip reform.
        Doing this check ensures that the 2017_law.json
        and subsequent reform files that represent recent legislation
are
        specified in a consistent manner.
        """
        # pylint: disable=too-many-locals
        # create clp metadata dictionary for current-law policy in
fyear
        clp_pol = Policy()
        clp_pol.set_year(fyear)
        clp_mdata = dict(clp_pol.items())
        # create rtr metadata dictionary for round-trip reform in
fyear
        rtr_pol = Policy()
        # Revert to 2017 law
        reform_file = os.path.join(tests_path, '..', 'reforms',
'2017_law.json')
        with open(reform_file, 'r') as rfile:
            rtext = rfile.read()
        rtr_pol.implement_reform(Policy.read_json_reform(rtext))
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on TCJA
        reform_file = os.path.join(tests_path, '..', 'reforms',
'TCJA.json')
        with open(reform_file, 'r') as rfile:
            rtext = rfile.read()
        rtr_pol.implement_reform(Policy.read_json_reform(rtext))
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on the CARES Act
        rtr_pol.implement_reform(
            {'ID_Charity_crt_all': {2020: 1.0, 2021: 0.6},
             'STD_allow_charity_ded_nonitemizers': {2020: True, 2021:
```

```
False},
             'STD_charity_ded_nonitemizers_max': {2020: 300.0, 2021:
0.0}})
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        # Layer on ARPA
        rtr_pol.implement_reform(
            {'RRC_c': {2021: 1400, 2022: 0},
             'RRC_ps': {2021: [75000, 150000, 75000, 112500,150000],
                        2022: [0, 0, 0, 0, 0]},
             'RRC_pe': {2021: [80000, 160000, 80000, 120000, 160000],
                        2022: [0, 0, 0, 0, 0]},
             'UI_em': {2020: 10200, 2021: 0},
             'UI_thd': {2020: [150000, 150000, 150000, 150000,
150000],
                        2021: [0, 0, 0, 0, 0]},
             'CTC_refundable': {2021: True, 2022: False},
             'CTC_include17': {2021: True, 2022: False},
             'CTC_new_c': {2021: 1000, 2022: 0},
             'CTC_new_c_under6_bonus': {2021: 600, 2022: 0},
             'CTC_new_for_all': {2021: True, 2022: False},
             'CTC_new_ps': {2021: [75000, 150000, 75000, 112500,
150000],
                            2022: [0, 0, 0, 0, 0]},
             'CTC_new_prt': {2021: 0.05, 2022: 0},
             'EITC_c': {2021: [1502.46, 3606.44, 5960.95, 6706.58],
                        2022: [546.21, 3640.7, 6017.58, 6770.29]},
             'EITC_rt': {2021: [0.153, 0.34, 0.4, 0.45],
                         2022: [0.0765, 0.34, 0.4, 0.45]},
             'EITC_ps': {2021: [11610, 19464.12, 19464.12, 19464.12],
                         2022: [8931.38, 19649.03, 19649.03,
19649.03]},
             'EITC_MinEligAge': {2021: 19, 2022: 25},
             'EITC_MaxEligAge': {2021: 125, 2022: 64},
             'EITC_InvestIncome_c': {2021: 10000},
             'EITC_sep_filers_elig': {2021: True},
             'CDCC_c': {2021: 8000, 2022: 3000},
             'CDCC_ps': {2021: 125000, 2022: 15000},
             'CDCC_ps2': {2021: 400000, 2022: 9e+99},
             'CDCC_crt': {2021: 50.0, 2022: 35.0},
             'CDCC_refundable': {2021: True, 2022: False},
             'ALD_BusinessLosses_c': {
                2026: [283535.22, 567070.42, 283535.22, 283535.22,
567070.42],
                2027: [9e+99, 9e+99, 9e+99, 9e+99, 9e+99]}})
        assert not rtr_pol.parameter_warnings
        assert not rtr_pol.errors
        rtr_pol.set_year(fyear)
        rtr_mdata = dict(rtr_pol.items())
        # compare fyear policy parameter values
        assert clp_mdata.keys() == rtr_mdata.keys()
        fail_dump = False
        if fail_dump:
            rtr_fails = open('fails_rtr', 'w')
            clp_fails = open('fails_clp', 'w')
        fail_params = list()
        msg = '\nRound-trip-reform and current-law-policy param values
differ for:'
        for pname in clp_mdata.keys():
            rtr_val = rtr_mdata[pname]
            clp_val = clp_mdata[pname]
            if not np.allclose(rtr_val, clp_val):
                fail_params.append(pname)
                msg += '\n  {} in {} : rtr={} clp={}'.format(
                    pname, fyear, rtr_val, clp_val
                )
                if fail_dump:
                    rtr_fails.write('{} {} {}\n'.format(pname, fyear,
rtr_val))
                    clp_fails.write('{} {} {}\n'.format(pname, fyear,
clp_val))
        if fail_dump:
            rtr_fails.close()
            clp_fails.close()
        if fail_params:
>           raise ValueError(msg)
E           ValueError:
E           Round-trip-reform and current-law-policy param values
differ for:
E             EITC_c in 2023 : rtr=[[ 556.21 3707.32 6127.7  6894.19]]
clp=[[ 570.25 3801.31 6174.97 7061.91]]
E             EITC_ps in 2023 : rtr=[[ 9094.82 20008.61 20008.61
20008.61]] clp=[[ 9327.63 20498.38 20498.38 20498.38]]
E             EITC_MaxEligAge in 2023 : rtr=[64] clp=[125]
E             EITC_InvestIncome_c in 2023 : rtr=[10355.09]
clp=[10488.49]

taxcalc/tests/test_reforms.py:181: ValueError
_________________________ test_reform_json_and_output
__________________________

tests_path = '/Users/bodiyang/Desktop/taxcalc/Tax-Calculator/taxcalc/
tests'

    def test_reform_json_and_output(tests_path):
        """
        Check that each JSON reform file can be converted into a
reform dictionary
        that can then be passed to the Policy class implement_reform
method that
```

```
        generates no parameter_errors.
        Then use each reform to generate static tax results for small
set of
        filing units in a single tax_year and compare those results
with
        expected results from a CSV-formatted file.
        """
        # pylint: disable=too-many-statements,too-many-locals

        # embedded function used only in test_reform_json_and_output
        def write_res_file(calc, resfilename):
            """
            Write calc output to CSV-formatted file with resfilename.
            """
            varlist = [
                'RECID', 'c00100', 'standard', 'c04800', 'iitax',
'payrolltax'
            ]
            # varnames  AGI    STD         TaxInc    ITAX     PTAX
            stats = calc.dataframe(varlist)
            stats['RECID'] = stats['RECID'].astype(int)
            with open(resfilename, 'w') as resfile:
                stats.to_csv(resfile, index=False,
float_format='%.2f')

        # embedded function used only in test_reform_json_and_output
        def res_and_out_are_same(base):
            """
            Return True if base.res.csv and base.out.csv file contents
are same;
            return False if base.res.csv and base.out.csv file
contents differ.
            """
            resdf = pd.read_csv(base + '.res.csv')
            outdf = pd.read_csv(base + '.out.csv')
            diffs = False
            for col in resdf:
                if col in outdf:
                    if not np.allclose(resdf[col], outdf[col]):
                        diffs = True
                else:
                    diffs = True
            return not diffs

        # specify Records object containing cases data
        tax_year = 2020
        cases_path = os.path.join(tests_path, '..', 'reforms',
'cases.csv')
        cases = Records(data=cases_path,
                        start_year=tax_year,  # set raw input data
year
                        gfactors=None,  # keeps raw data unchanged
                        weights=None,
                        adjust_ratios=None)
        # specify list of reform failures
        failures = list()
        # specify current-law-policy Calculator object
        calc = Calculator(policy=Policy(), records=cases,
verbose=False)
        calc.advance_to_year(tax_year)
        calc.calc_all()
        res_path = cases_path.replace('cases.csv', 'clp.res.csv')
        write_res_file(calc, res_path)
        if res_and_out_are_same(res_path.replace('.res.csv', '')):
            os.remove(res_path)
        else:
            failures.append(res_path)
        del calc
        # read 2017_law.json reform file and specify its parameters
dictionary
        pre_tcja_jrf = os.path.join(tests_path, '..', 'reforms',
'2017_law.json')
        pre_tcja = Policy.read_json_reform(pre_tcja_jrf)
        # check reform file contents and reform results for each
reform
        reforms_path = os.path.join(tests_path, '..', 'reforms',
'*.json')
        json_reform_files = glob.glob(reforms_path)
        for jrf in json_reform_files:
            # determine reform's baseline by reading contents of jrf
            with open(jrf, 'r') as rfile:
                jrf_text = rfile.read()
            pre_tcja_baseline = 'Reform_Baseline: 2017_law.json' in
jrf_text
            # implement the reform relative to its baseline
            reform = Policy.read_json_reform(jrf_text)
            pol = Policy()  # current-law policy
            if pre_tcja_baseline:
                pol.implement_reform(pre_tcja)
                assert not pol.parameter_errors
            pol.implement_reform(reform)
            assert not pol.parameter_errors
            calc = Calculator(policy=pol, records=cases,
verbose=False)
            calc.advance_to_year(tax_year)
            calc.calc_all()
            res_path = jrf.replace('.json', '.res.csv')
            write_res_file(calc, res_path)
            if res_and_out_are_same(res_path.replace('.res.csv', '')):
                os.remove(res_path)
            else:
                failures.append(res_path)
            del calc
        if failures:
            msg = 'Following reforms have res-vs-out differences:\n'
            for ref in failures:
                msg += '{}\n'.format(os.path.basename(ref))
>           raise ValueError(msg)
E           ValueError: Following reforms have res-vs-out differences:
E           clp.res.csv
E           TCJA.res.csv
E           Larson2019.res.csv
E           2017_law.res.csv
E           Renacci.res.csv
E           SandersDeFazio.res.csv
E           ptaxes3.res.csv
E           BrownKhanna.res.csv
E           ptaxes0.res.csv
E           Trump2017.res.csv
E           Trump2016.res.csv

taxcalc/tests/test_reforms.py:280: ValueError
=============================== warnings summary
===============================
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/marshmallow/__init__.py:17
  /opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-packages/
marshmallow/__init__.py:17: DeprecationWarning: distutils Version
classes are deprecated. Use packaging.version instead.
    __version_info__ = tuple(LooseVersion(__version__).version)

../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/fsspec/registry.py:188
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/fsspec/registry.py:188
  /opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-packages/fsspec/
registry.py:188: DeprecationWarning: distutils Version classes are
deprecated. Use packaging.version instead.
    minversions = {"s3fs": LooseVersion("0.3.0"), "gcsfs":
LooseVersion("0.3.0")}

../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/marshmallow/fields.py:181
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/marshmallow/fields.py:181
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/marshmallow/fields.py:181
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/marshmallow/fields.py:181
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
```

```
packages/marshmallow/fields.py:181
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/marshmallow/fields.py:181
../../../../../opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-
packages/marshmallow/fields.py:181
  /opt/miniconda3/envs/taxcalc-dev/lib/python3.9/site-packages/
marshmallow/fields.py:181: RemovedInMarshmallow4Warning: The 'missing'
argument to fields is deprecated. Use 'load_default' instead.
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/warnings.html
=========================== short test summary info
============================
FAILED taxcalc/tests/test_cpscsv.py::test_agg - ValueError: CPSCSV AGG
RESULT...
FAILED taxcalc/tests/test_reforms.py::test_round_trip_reforms[2021] -
ValueEr...
FAILED taxcalc/tests/test_reforms.py::test_round_trip_reforms[2022] -
ValueEr...
FAILED taxcalc/tests/test_reforms.py::test_round_trip_reforms[2023] -
ValueEr...
FAILED taxcalc/tests/test_reforms.py::test_reform_json_and_output -
ValueErro...
==== 5 failed, 256 passed, 128 deselected, 10 warnings in 706.02s
(0:11:46) ====
(taxcalc-dev) bodiyang@BYang-MBP Tax-Calculator %
```