Task 1

Count the number of cities with populations that are greater than 50000.

Solution

```
def problem1(populations):
"""
the function takes a parameter populations (list of numbers)
and return number of populations greater than 50000 (integer)
"""
# initial counter
count = 0
# initial value measure with population
measure = 50000
 # loop through list with populations
for i in populations:
# if population greater than measure
        if i > measure:
        # add to count 1
        count += 1
# return count populations greater than 50000
return count
# now call the function
problem1(pop_populations)
```

Answer:

48

Task 2:

Find the mean population of all the given towns. Then find how many towns have a population within 3000 (plus or minus) of that number.

Solution:

```
def problem2(populations, diff=3000):
  """
  The function takes parameter populations (list of numbers),
  finds the mean population and quantity towns have a population
  within 3000 (plus or minus) of mean.
  Returns number (integer)
  """
  # find mean population
  mean = sum(populations)/len(populations)
  # define counter with value 0
  counter = 0
  # for each population in list
  for i in populations:
    # if population is in interval mean +- diff
    if (mean - diff) < i < (mean + diff):
      # increase counter by 1
      counter += 1
  return counter
```

```
# now call the function
problem2(pop_populations)
```

Answer:
55

Task 3

How many of the towns in the dataset are in Rhode Island (RI)?

Sulution:

```
def problem3(towns, st_abbrv='RI'):
    """
    Function finds how many of the towns
    in the dataset are in Rhode Island.
    Takes one parameter towns (list of strings)
    and returns number (integer)
    """
    # define counter with value 0
    counter = 0
    # for each town in list of towns
    for town in towns:
        # if town in RI
        if town.endswith(st_abbrv):
            # increase counter by 1
            counter += 1
    return counter

# now call the function
problem3(pop_towns)
```

Answer:
39

Task 4

What is the **number** of **unique** town names in the towns list when the state is not considered.

Solution:

```
def problem4(towns):
    """
    Function takes parameter towns (list of strings)
    and finds number of unique town names.
    Return number (integer)
    """
    # define empty list uniques to store here fuond towns
    uniques = []
    # for each tjwn in towns
    for town in towns:
        # split by ',' to separate town's name from state
        town = town.split(', ')
        # if towns name was found in first time
        if town[0] not in uniques:
            # add it to list uniques
            uniques.append(town[0])
    # return length of list uniques that is count of all unique town's names
    return len(uniques)
# now call the function
problem4(pop_towns)
```

Answer:
510

Task 5

What is the name of the town with the smallest population in this dataset?

Solution:

```
def problem5(towns, populations):
    """
    The function takes two parameters towns (list of strings)
    and populations (list of numbers). Finds name of the town with the smallest population.
    Return (string)
    """
    # add code here
    # find smallest number in list of populations and it's index
    index = populations.index(min(populations))
    # return town from list of towns by this index
    return towns[index]

# now call the function
problem5(pop_towns, pop_populations)
```

Answer:
'Gosnold, MA'

Task 6

Compute all town names that appear in **exactly two** states in this dataset.

Solution:

```
def extra_credit(towns):
    """
    Function takes one parameter towns (list of strings),
    finds all town names that appear in exactly two states.
    Return list of strings
    """
    # add code here
    # define empty dict dict_towns to store towns(keys) and states(values)
    dict_towns = {}
    # efine empty list _towns to store found towns
    _towns = []
    # for each town
    for town in towns:
        # find name and state of town
        name, state = town.split(', ')
        # if town not in dict_towns
        if name not in dict_towns:
            # store it in dict as key and state as value
            dict_towns[name] = [state]
        # if name in dict
        else:
            # add to it's values new state
            dict_towns[name].append(state)
    # for each town in dict
    for town in dict_towns:
        # if town appears in two states
        if len(dict_towns[town]) == 2:
            # append it to list _towns
            _towns.append(town)
    return _towns

# now call the function
extra_credit(pop_towns)
```

Answer:

```
['Westport',
 'Cheshire',
 'Sharon',
 'Chester',
 'Hopkinton',
 'Salisbury',
 'Winchester',
 'Weston',
 'Bolton',
 'Warwick',
 'Franklin',
 'Plainfield',
 'Washington',
 'Bristol',
 'Clinton',
 'Berlin',
 'Oxford',
 'Mansfield',
 'Plymouth',
 'Milford',
 'Avon',
 'Brookfield',
 'Watertown',
 'Groton',
 'Windsor',
 'Canton',
 'Essex',
 'Bridgewater',
 'Sterling',
 'Tolland',
 'Middletown',
     'Easton',
     'Andover',
     'Granby',
     'Salem',
     'Scituate',
     'Monroe',
     'Norfolk',
     'Goshen',
     'Lincoln',
     'Burlington',
     'Middlefield',
     'Plainville',
     'Marlborough',
     'Coventry',
     'Orange',
     'Richmond']
```

Tests:

```
......
----------------------------------------------------------------------
Ran 6 tests in 0.049s

OK
```

Out[11]:

```
<unittest.runner.TextTestResult run=6 errors=0 failures=0>
```