# CS 115: Project 3
# Air Quality Index Calculator (Redux)

## Goals

In this project, you will revisit our prior Air Quality Index (AQI) project, re-implementing it using the below concepts.

You will be practicing the following concepts from prior labs:

- `for`-loops and conditionals
- lists and 2D lists
- min/max/avg calculations with lists
- writing functions from specifications
- writing new functions and their docstrings

## Summary

The Air Quality Index (AQI) is a simple, unitless index used to communicate air pollution to the general public. The concept and calculations for AQI are fully explained in the description for our Project 1.

In this project, you will re-implement your Project 1 assignment in its entirety, following the functions and comments in the provided template. Some functions must be implemented as specified, and some functions you will create and document yourself. The behavior of Project 3's Final Code should agree with the behavior and sample input/outputs of Project 1's Final Code. Pay attention to the grading rubric, which has changed since last time.

In all samples below, user input is shown in *italics and underlined*.

## Due Dates

- Checkpoint A: Due as a demo in any lab, drop-in tutoring or workshop before Thursday, Nov. 30 at 7 PM.
- Final Code: Due via Moodle on Thursday, Dec. 7 at 11:55 PM.

## Template

You will need the template for the project:

- Download the template, template_P3.py

The template contains functions and specifications for functions you will implement. The functions each have docstrings written for you holding the **function specifications**. Your implementation of the function should match the specifications provided in this template.

# Checkpoint A

Running the unmodified template demonstrates the following input/output behavior:

```
=== Air Quality Index (AQI) Calculator ===
Select the number of locations for the report: 2


Enter name of ** Location 1 **: A
         [None, None, None, None, None, None, None]
         [-1, -1, -1, -1, -1, -1, -1]
['A']
[-1]
[]


Enter name of ** Location 2 **: B
         [None, None, None, None, None, None, None]
         [-1, -1, -1, -1, -1, -1, -1]
['A', 'B']
[-1, -1]
[]
```

This will be the "skeleton" of the program that will give us insight into the values and calculations for the demo of Part A.

For Checkpoint A, you will need to modify the template to demonstrate a program that satisfies the following:

1. The functions `get_concentration()` and `get_per_pollutant_index()` are fully implemented as specified.
2. The program matches the sample input/output below.

   Any feedback given on your Project 1 code should be incorporated.
3. **Demo.** Demo Checkpoint A.

## Sample Input/Output

### Sample 8

```
=== Air Quality Index (AQI) Calculator ===
Select the number of locations for the report: 7


Enter name of ** Location 1 **: Cupertino
    -> PM-2.5 [ug/m3, 24-hr avg]: 12.3
    -> PM-10 [ug/m3, 24-hr avg]: 0.1
    -> NO2 [ppb, 1-hr avg]: 0.5
    -> SO2 [ppb, 1-hr avg]: 11
    -> CO [ppm, 8-hr avg]: 12
    -> O3 [ppb, 8-hr avg]: 110
    -> O3 [ppb, 1-hr avg]: 170
         [12.3, 0, 0, 11, 12.0, 110, 170]
         [51, 0, 0, 16, 143, 205, 157]
['Cupertino']
[205]
[12.3]


Enter name of ** Location 2 **: Davis
    -> PM-2.5 [ug/m3, 24-hr avg]: 130
    -> PM-10 [ug/m3, 24-hr avg]: 25.33
    -> NO2 [ppb, 1-hr avg]: 5.4
    -> SO2 [ppb, 1-hr avg]: 0
    -> CO [ppm, 8-hr avg]: 1.4
    -> O3 [ppb, 8-hr avg]: 69
    -> O3 [ppb, 1-hr avg]: 100
         [130.0, 25, 5, 0, 1.4, 69, 100]
         [189, 23, 5, 0, 16, 97, -1]
['Cupertino', 'Davis']
[205, 189]
```

```
[12.3, 130.0]


Enter name of ** Location 3 **: Woodland
    -> PM-2.5 [ug/m3, 24-hr avg]: 20
    -> PM-10 [ug/m3, 24-hr avg]: 10
    -> NO2 [ppb, 1-hr avg]: 40
    -> SO2 [ppb, 1-hr avg]: 15
    -> CO [ppm, 8-hr avg]: 1.4
    -> O3 [ppb, 8-hr avg]: 10
    -> O3 [ppb, 1-hr avg]: 123
         [20.0, 10, 40, 15, 1.4, 10, 123]
         [68, 9, 38, 21, 16, 9, -1]
['Cupertino', 'Davis', 'Woodland']
[205, 189, 68]
[12.3, 130.0, 20.0]

Enter name of ** Location 4 **: Vacaville
    -> PM-2.5 [ug/m3, 24-hr avg]: 10
    -> PM-10 [ug/m3, 24-hr avg]: 2
    -> NO2 [ppb, 1-hr avg]: 10
    -> SO2 [ppb, 1-hr avg]: 5
    -> CO [ppm, 8-hr avg]: 2.4
    -> O3 [ppb, 8-hr avg]: 8
    -> O3 [ppb, 1-hr avg]: 165
         [10.0, 2, 10, 5, 2.4, 8, 165]
         [42, 2, 9, 7, 27, 7, 151]
['Cupertino', 'Davis', 'Woodland', 'Vacaville']
[205, 189, 68, 151]
[12.3, 130.0, 20.0, 10.0]

Enter name of ** Location 5 **: Santa Cruz
    -> PM-2.5 [ug/m3, 24-hr avg]: 5
    -> PM-10 [ug/m3, 24-hr avg]: 12.5
```

```
    -> NO2 [ppb, 1-hr avg]: 20.8
    -> SO2 [ppb, 1-hr avg]: 4
    -> CO [ppm, 8-hr avg]: 1.1
    -> O3 [ppb, 8-hr avg]: -1
    -> O3 [ppb, 1-hr avg]: -1
         [5.0, 12, 20, 4, 1.1, -1, -1]
         [21, 11, 19, 6, 12, -1, -1]
['Cupertino', 'Davis', 'Woodland', 'Vacaville', 'Santa Cruz']
[205, 189, 68, 151, 21]
[12.3, 130.0, 20.0, 10.0, 5.0]

Enter name of ** Location 6 **: Watsonville
    -> PM-2.5 [ug/m3, 24-hr avg]: 12
    -> PM-10 [ug/m3, 24-hr avg]: -1
    -> NO2 [ppb, 1-hr avg]: -1
    -> SO2 [ppb, 1-hr avg]: -1
    -> CO [ppm, 8-hr avg]: -1
    -> O3 [ppb, 8-hr avg]: -1
    -> O3 [ppb, 1-hr avg]: -1
         [12.0, -1, -1, -1, -1, -1, -1]
         [50, -1, -1, -1, -1, -1, -1]
['Cupertino', 'Davis', 'Woodland', 'Vacaville', 'Santa Cruz', 'Watsonville']
[205, 189, 68, 151, 21, 50]
[12.3, 130.0, 20.0, 10.0, 5.0, 12.0]

Enter name of ** Location 7 **: Salinas
    -> PM-2.5 [ug/m3, 24-hr avg]: -1
    -> PM-10 [ug/m3, 24-hr avg]: -1
    -> NO2 [ppb, 1-hr avg]: 55
    -> SO2 [ppb, 1-hr avg]: -1
    -> CO [ppm, 8-hr avg]: 3.0
    -> O3 [ppb, 8-hr avg]: 56
    -> O3 [ppb, 1-hr avg]: -1
```

```
          [-1, -1, 55, -1, 3.0, 56, -1]

          [-1, -1, 52, -1, 34, 54, -1]

['Cupertino', 'Davis', 'Woodland', 'Vacaville', 'Santa Cruz', 'Watsonville', 'Salinas']

[205, 189, 68, 151, 21, 50, 54]

[12.3, 130.0, 20.0, 10.0, 5.0, 12.0]
```

### Advice and Hints

Warning: You must implement the functions as specified. Functions should not print anything, unless it is specified.

# Final Code

In your final code, you will extend the prior code to match the full behavior of Project 1's Final Code. In particular:

- You will create any missing printed output by writing new functions and adding appropriate calls to these in `main()`. The functions `get_per_pollutant_index()` and `get_concentration()` still should not print anything.
- You should implement a function to produce the "per-pollutant concentration" statements and document its behavior using a docstring that follows our class docstring conventions (there is a TODO for this in the template).
- You should implement a function to produce the "Summary Report" and document its behavior using a docstring that follows our class docstring conventions (there is a TODO for this in the template).
- All printed output should be formatted as shown in the prior project; see Sample 7, Sample 9 and Sample 10.
- You should remove extraneous printed output to match the provided samples.
- The error behavior should be as described in the prior project; see Sample 5, Sample 6.

There is no demo for your final code.

### Advice and Hints

Warning: Do not ignore the TODOs requesting that you write functions to produce the sample output.

**Sample 9**

```
=== Air Quality Index (AQI) Calculator ===

Select the number of locations for the report: 1


Enter name of ** Location 1 **: Newport Beach

    -> PM-2.5 [ug/m3, 24-hr avg]: -1
```

```
    -> PM-10 [ug/m3, 24-hr avg]: 40.1
       PM-10 concentration 40 yields 37 index
    -> NO2 [ppb, 1-hr avg]: 25
       NO2 concentration 25 yields 24 index
    -> SO2 [ppb, 1-hr avg]: -1
    -> CO [ppm, 8-hr avg]: 12
       CO concentration 12.0 yields 143 index
    -> O3 [ppb, 8-hr avg]: 302
    -> O3 [ppb, 1-hr avg]: 126
       O3 concentration 126 yields 102 index
    AQI for Newport Beach is 143
    Condition: Unhealthy for Sensitive Groups


Summary Report
    Location with max AQI is Newport Beach (143)
    Location with min AQI is Newport Beach (143)
    Avg PM-2.5 concentration reading: No Data
```

**Sample 10**

```
=== Air Quality Index (AQI) Calculator ===
Select the number of locations for the report: 7


Enter name of ** Location 1 **: Cupertino
    -> PM-2.5 [ug/m3, 24-hr avg]: 12.3
       PM-2.5 concentration 12.3 yields 51 index
    -> PM-10 [ug/m3, 24-hr avg]: 0.1
       PM-10 concentration 0 yields 0 index
    -> NO2 [ppb, 1-hr avg]: 0.5
       NO2 concentration 0 yields 0 index
    -> SO2 [ppb, 1-hr avg]: 11
       SO2 concentration 11 yields 16 index
    -> CO [ppm, 8-hr avg]: 12
       CO concentration 12.0 yields 143 index
```

```
    -> O3 [ppb, 8-hr avg]: 110
    -> O3 [ppb, 1-hr avg]: 170
       O3 concentration 110 yields 205 index
    AQI for Cupertino is 205
    Condition: Very Unhealthy

Enter name of ** Location 2 **: Davis
    -> PM-2.5 [ug/m3, 24-hr avg]: 130
       PM-2.5 concentration 130.0 yields 189 index
    -> PM-10 [ug/m3, 24-hr avg]: 25.33
       PM-10 concentration 25 yields 23 index
    -> NO2 [ppb, 1-hr avg]: 5.4
       NO2 concentration 5 yields 5 index
    -> SO2 [ppb, 1-hr avg]: 0
       SO2 concentration 0 yields 0 index
    -> CO [ppm, 8-hr avg]: 1.4
       CO concentration 1.4 yields 16 index
    -> O3 [ppb, 8-hr avg]: 69
    -> O3 [ppb, 1-hr avg]: 100
       O3 concentration 69 yields 97 index
    AQI for Davis is 189
    Condition: Unhealthy

Enter name of ** Location 3 **: Woodland
    -> PM-2.5 [ug/m3, 24-hr avg]: 20
       PM-2.5 concentration 20.0 yields 68 index
    -> PM-10 [ug/m3, 24-hr avg]: 10
       PM-10 concentration 10 yields 9 index
    -> NO2 [ppb, 1-hr avg]: 40
       NO2 concentration 40 yields 38 index
    -> SO2 [ppb, 1-hr avg]: 15
       SO2 concentration 15 yields 21 index
    -> CO [ppm, 8-hr avg]: 1.4
```

```
       CO concentration 1.4 yields 16 index
    -> O3 [ppb, 8-hr avg]: 10
    -> O3 [ppb, 1-hr avg]: 123
       O3 concentration 10 yields 9 index
    AQI for Woodland is 68
    Condition: Moderate

Enter name of ** Location 4 **: Vacaville
    -> PM-2.5 [ug/m3, 24-hr avg]: 10
       PM-2.5 concentration 10.0 yields 42 index
    -> PM-10 [ug/m3, 24-hr avg]: 2
       PM-10 concentration 2 yields 2 index
    -> NO2 [ppb, 1-hr avg]: 10
       NO2 concentration 10 yields 9 index
    -> SO2 [ppb, 1-hr avg]: 5
       SO2 concentration 5 yields 7 index
    -> CO [ppm, 8-hr avg]: 2.4
       CO concentration 2.4 yields 27 index
    -> O3 [ppb, 8-hr avg]: 8
    -> O3 [ppb, 1-hr avg]: 165
       O3 concentration 165 yields 151 index
    AQI for Vacaville is 151
    Condition: Unhealthy

Enter name of ** Location 5 **: Santa Cruz
    -> PM-2.5 [ug/m3, 24-hr avg]: 5
       PM-2.5 concentration 5.0 yields 21 index
    -> PM-10 [ug/m3, 24-hr avg]: 12.5
       PM-10 concentration 12 yields 11 index
    -> NO2 [ppb, 1-hr avg]: 20.8
       NO2 concentration 20 yields 19 index
    -> SO2 [ppb, 1-hr avg]: 4
       SO2 concentration 4 yields 6 index
```

```
    -> CO [ppm, 8-hr avg]: 1.1
       CO concentration 1.1 yields 12 index
    -> O3 [ppb, 8-hr avg]: -1
    -> O3 [ppb, 1-hr avg]: -1
    AQI for Santa Cruz is 21
    Condition: Good

Enter name of ** Location 6 **: Watsonville
    -> PM-2.5 [ug/m3, 24-hr avg]: 12
       PM-2.5 concentration 12.0 yields 50 index
    -> PM-10 [ug/m3, 24-hr avg]: -1
    -> NO2 [ppb, 1-hr avg]: -1
    -> SO2 [ppb, 1-hr avg]: -1
    -> CO [ppm, 8-hr avg]: -1
    -> O3 [ppb, 8-hr avg]: -1
    -> O3 [ppb, 1-hr avg]: -1
    AQI for Watsonville is 50
    Condition: Good

Enter name of ** Location 7 **: Salinas
    -> PM-2.5 [ug/m3, 24-hr avg]: -1
    -> PM-10 [ug/m3, 24-hr avg]: -1
    -> NO2 [ppb, 1-hr avg]: 55
       NO2 concentration 55 yields 52 index
    -> SO2 [ppb, 1-hr avg]: -1
    -> CO [ppm, 8-hr avg]: 3.0
       CO concentration 3.0 yields 34 index
    -> O3 [ppb, 8-hr avg]: 56
    -> O3 [ppb, 1-hr avg]: -1
       O3 concentration 56 yields 54 index
    AQI for Salinas is 54
    Condition: Moderate
```

```
Summary Report
    Location with max AQI is Cupertino (205)
    Location with min AQI is Santa Cruz (21)
    Avg PM-2.5 concentration reading: 31.5
```

# Grading Rubric

**Checkpoints [10%]**

The checkpoint demo is worth 10 points (all or nothing).

**Programming Design and Style [35%]**

In addition to being correct, your program should be easy to understand and well documented. For details, see the rubric below.

**Correctness [55%]**

Your program will be tested numerous times, using different inputs, to be sure that it meets the specification. You will not get full credit for this unless your output matches the sample output exactly for every case, including capitalization and spacing. For details, see the rubric below.

## Detailed Rubric

### Correctness: functionality and specifications (50 points -- 5 points each)

| | |
|---|---|
| Feature 1: | Program makes effective use of the `MONITORS` and `BREAKPOINTS` variables from the template, and these have not been modified. In particular, the program does not hardcode any of these values in the program logic. |
| Feature 2: | The behavior of the `get_concentration()` function matches its specification. |
| Feature 3: | The `get_concentration()` function is implemented in a way that does not duplicate code. If there are lines of code that have been repeatedly copy-pasted, consider re-writing the function's logic. |
| Feature 4: | The behavior of the `get_per_pollutant_index()` function matches its specification. |

| | |
|---|---|
| Feature 5: | The `get_per_pollutant_index()` function is implemented in a way that does not duplicate code. If there are lines of code that have been repeatedly copy-pasted, consider re-writing the function's logic. |
| Feature 6: | When the user inputs an invalid number for the number of locations, the program produces an appropriate error message and exit code (Ex: Samples 5 and 6). |
| Feature 7: | The per-pollutant concentration is printed by a sensible function that is correctly documented. It should be used where the template had originally noted a TODO including `write a new function`. |
| Feature 8: | The Summary Report is printed using a sensible function that is correctly documented. It should be used where the template had originally noted a TODO including `write a new function`. |
| Feature 9: | The average PM-2.5 concentration is handled correctly in the Summary Report, even in the scenario where no PM-2.5 readings were recorded (Ex: Samples 9 and 10). |
| Feature 10: | Functions call one another. There is no function that has been written that is not used. |

## Correctness: spacing, newlines, output decorations, etc. (5 points)

All or nothing: all spacing, newlines, text decorations (`'->'`, etc.) and printed output match the sample output. You can check yours by using an online diff tool to compare your program output with the text in Sample 10. This will be automatically graded -- if there is any difference, no matter the magnitude, this will not be marked correct.

## Programming Design and Style (30 points)

### Docstring (5 points)

There should be a docstring at the top of your submitted file with the following information:

1 pt. Your name (first and last)

1 pt. The course (CS 115)

1 pt. The assignment (e.g., Project 1)

2 pts. A brief description of what the program does

### Documentation (6 points)

Not counting the docstring, your program should contain at least three comments explaining aspects of your code that are potentially tricky for a person reading it to understand. You

should assume that the person understands what Python syntax means but may not understand *why* you are doing what you are doing.

6 pts. You have at least 3 useful comments (2 points each)

**Variables (5 points)**

5 pts. Variables have helpful names that indicate what kind of information they contain.

**Algorithm (4 points)**

2 pts. Your algorithm is straightforward and easy to follow.

2 pts. Your algorithm is reasonably efficient, with no wasted computation or unused variables.

**Dead code or misleading comments (5 points)**

3 pts. Your final program should not contain dead code. Dead code is any old code that has been commented-out or otherwise been made permanently inactive, i.e., for the purposes of development or testing.

2 pts. Your program should not have irrelevant or misleading comments. This includes `#TODO` comments that do not indicate things that still need to be done; if they been done then they should no longer be marked "TODO". This also includes comments you have added to the code or inherited from the template that do not describe the code any longer.

**Program structure (5 points)**

All or nothing: your code should define a main function and then call that function, just like our programs do in the lab. Other than library imports, the docstring, and the final call to `main()`, you should not have any stray code outside a function definition.

**Catchall**

For students using language features that were not covered in class, up to 5 points may be taken off if the principles of programming style are not adhered to when using these features. If you have any questions about what this means, then *ask*.

# Submission

You should submit your final code on Moodle by the deadline. I strongly encourage you to take precautions to make and manage backups while you work on your project, in case something goes wrong either while working or with your submission to Moodle.

Name the file you submit to Moodle `*yourlastname*P3.py`, substituting your actual last name (in lowercase) for *yourlastname*.

# Late Policies

Project late policies are outlined in the course policies page.

# Collaboration Policy

Programming projects must be your own work, and academic misconduct is taken very seriously. You may discuss ideas and approaches with other students and the course staff, but you should work out all details and write up all solutions on your own. The following actions will be penalized as academic dishonesty:

- Copying part or all of another student's assignment
- Copying old or published solutions
- Looking at another student's code or discussing it in great detail. You will be penalized if your program matches another student's program too closely.
- Showing your code or describing your code in great detail to anyone other than the course staff or tutor.

Project collaboration policies are described in the course policies page.

CS115 Fall 2017

Link to original work: http://www.cs.sonoma.edu/cs115/F17/proj/p3/cs115_p3.html