**Standard Deviation**

**Implementation notes**

Given the object-oriented approach for the design of the project, standard\_deviation module was developed to implement required functionalities for standard deviation function in the calculator. Implementation encapsulated functionalities using StandardDeviation class. Desired properties of the standard deviation function such as mean, population standard deviation, and sample standard deviation are integrated as attributes of the class. Class variable, attributes, and helper functions: \_\_mean, \_population\_standard\_deviation, and \_sample\_standard\_deviation are kept private to avoid privacy leak. Class variable and attributes can be accessed and modified through their respective accessor and mutator. The class constructor will initialize

* \_values which represents the set of values, using the arguments given during invocation.
* \_n\_values which represents the number of values in the set, by incrementing a counter while looping through the set of values.
* \_mean which represents the mean of the set, by utilizing the helper function \_\_mean.
* \_psd which represents the population standard deviation, using the helper function \_population\_standard\_deviation.
* \_ssd represents the sample standard deviation for the set, by utilizing the helper function \_sample\_standard\_deviation

Aside from the mutators and accessors, add\_values function has been implemented to allow new values to be appended to the already existing set. Invocation of this function will trigger an update of other attributes as they’re dependent on the \_values attribute (same logic is used in implementation of the set\_values function).

**Pseudocodes**

**Algorithm** mean(S, n)

**Input** set S of n values

**Output** mean of the set

total 🡨 0

**for** **each** value **in** S **do**

total 🡨 total **+** value

**return** total**/**n

**Algorithm** population\_standard\_deviation(S, n)

**Input** set **S** of **n** values

total 🡨 0

mean 🡨 mean(S, n)

**for each** value **in** S **do**

total 🡨 total **+**

**return** sqrt(total/n)