```python
import math
import random


def mean(values):
    """ Arithmetic mean """
    return float(sum(values)) / len(values)


def variance(values):
    """ Unbiased variance """
    sample_mean = mean(values)
    sum_of_squares = 0
    for sample in values:
        sum_of_squares += (sample - sample_mean) ** 2
    return sum_of_squares / (len(values) - 1)


def adaptive_confidence_interval(values, max_iterations=1000, alpha=0.05, trials=5,
variance_threshold=0.5):
    """ Compute confidence interval using as few iterations as possible """

    try_iterations = 10

    while True:
        intervals = [confidence_interval(values, try_iterations, alpha) for _ in range(trials)]
        band_variance = variance([upper_bound - lower_bound for lower_bound, upper_bound in intervals])

        print(try_iterations, band_variance)

        if band_variance < variance_threshold or try_iterations > max_iterations:
            return intervals[random.randint(0, trials - 1)], try_iterations

        try_iterations *= 2


def confidence_interval(values, iterations, alpha):
    """ Compute confidence interval of mean """

    subsample_means = []
    for _ in range(iterations):
        subsample_indices = [random.randint(0, len(values) - 1) for _ in range(len(values))]
        subsample_values = [values[idx] for idx in subsample_indices]
        subsample_means.append(mean(subsample_values))
    subsample_means.sort()

    lower_index = int(math.floor(iterations * (1 - alpha / 2)))
    upper_index = int(math.floor(iterations * alpha / 2))

    pivot = lambda idx: (2 * mean(values) - subsample_means[idx])

    return pivot(lower_index), pivot(upper_index)


def main(sample_size):
    random.seed(12345)
    values = [random.randint(0, 1000) for _ in range(sample_size)]
    print(mean(values), adaptive_confidence_interval(values))


if __name__ == '__main__':
    N = 100000
    main(N)
```

Python profiling cheat sheet*

Timing code

```
$ time -p python script.py
$ time -p python -c 'test_code()'

$ perf stat python script.py

$ python -m timeit -v -n NUM_LOOPS -r NUM_REPEATS \
    -s 'import foo; setup_code()' 'test_code()'
```

Profiling CPU usage

cProfile

```
$ python -m cProfile -s cumulative script.py

$ python -m cProfile -o script.cProfile script.py
$ pyprof2calltree -i script.cProfile -o cachegrind.out
$ kcachegrind cachegrind.out &
```

line_profiler†

```
$ kernprof.py -l -o script.lprof script.py
$ python -m line_profiler script.lprof
```

statprof.py

```
import statprof
statprof.start()
try:
    do_something()
finally:
    statprof.stop()
statprof.display()
```

Profiling memory usage

memory_profiler†

```
$ python -m memory_profiler script.py
```

Massif

```
$ valgrind --tool=massif python script.py
$ massif-visualizer &
```

*Materials available at http://zaytsev.net/confidence-2.zip

†Don't forget to add the `@profile` decorator