University of Tsukuba
Experiment Design in Computer Sciences

ASSIGNMENT 2:
For and While Loop Comparison

Sato Mario Tsukassa – 202220694

1. Introduction

When studying a programming language, it is natural to learn the iterators *for loop* and *while loop*. However, it is quite difficult for beginners to understand which one is better in terms of speed for processing numbers. Naturally, some programmers prefer *for loop* over *while loop*, and vice-versa. But it does not mean necessarily that one is better in terms of efficiency than another. In this assignment, it is proposed the comparison of *for* and *while loop* to test which one is faster for a specific task designed in this experiment.

2. Design

In this report, the proposed hypothesis is that the **processing speed of counting numbers from 1 to 1,000,000 (1 million) by using *for loop* is faster than the *while loop* to do the same task**.

2.1. Variables

Control Factors	Response Variable
For loop iterator	Processing time of counting the numbers from 1 to 1,000,000 (in seconds)
While loop iterator	Processing time of counting the numbers from 1 to 1,000,000 (in seconds)

As the control factors, it is defined 2 types of iterators, which are *for loop* and *while loop*. Furthermore, as the response variable, there is the processing time of counting the numbers from 1 to 1,000,000 (1 million) in seconds.

2.2. Environment

Upon the environment for the test, the **Jupyter Lab** notebook was used as the computer interaction platform. Also, it is worthwhile to mention that the condition of the operating system is the same to run both iterators. In other words, all the software will be closed before the start of the running experiment. To measure the processing time, the ***timeit*** function is used. It counts the execution time, in seconds, of a specific function when used. Moreover, Python 3.9.10 was used as the main language in this experiment.

2.3. Experiment Parameters

For this comparison of two types of iterators, it was defined some important parameters:

- Desired significance: $\alpha = 0.05$
- Minimum desired power: $(1 - \beta) = 0.85$

Since the experimental question is about the difference in processing time between two iterators, the mean processing time of these can be compared. The hypothesis can be written as follows:

$$\boldsymbol{Null\ Hypothesis\ (H_0)}: \ \boldsymbol{\mu_{for} \geq \mu_{while}}$$

(mean processing time of *for loop* is higher or equal to the mean processing time of *while loop* to count integer from 1 to 1 million)

$$\boldsymbol{Alternative\ Hypothesis\ (H_1)}: \ \boldsymbol{\mu_{for} < \mu_{while}}$$

(mean processing time of *for loop is* lower than the mean processing time of *while loop* to count integer from 1 to 1 million)

Next, to find the right sample size, the pilot study with 10 sample sizes was conducted to obtain the standard deviation for each iteration. The values of the standard deviation of *for loop* and *while loop* is 0.00536 and 0.00547 respectively. Thus, the reasonable value is *sd* = 0.006. With this, it was possible to calculate the standardized effect size.

- Standardized effect size (Cohen's d): 0.20 (medium)

The small, standardized effect size ($d^* = 0.20$) was defined based on Cohen's d values [2]. Now, the delta can be found using this equation.

$$d^* = \delta^*/\sigma = 0.20$$

$$\delta = \sigma * 0.20 = 0.006 * 0.20 = 0.0012$$

Thus, applying the power t-test for sample size, will present the sample as follows:

```
> power.t.test(delta = (0.006*0.2), sd = 0.006, sig.level = 0.05,
+              power = 0.85,
+              type = "two.sample",
+              alternative = "one.sided")

     Two-sample t test power calculation

              n = 360.1433
          delta = 0.0012
             sd = 0.006
      sig.level = 0.05
          power = 0.85
    alternative = one.sided

NOTE: n is number in *each* group
```

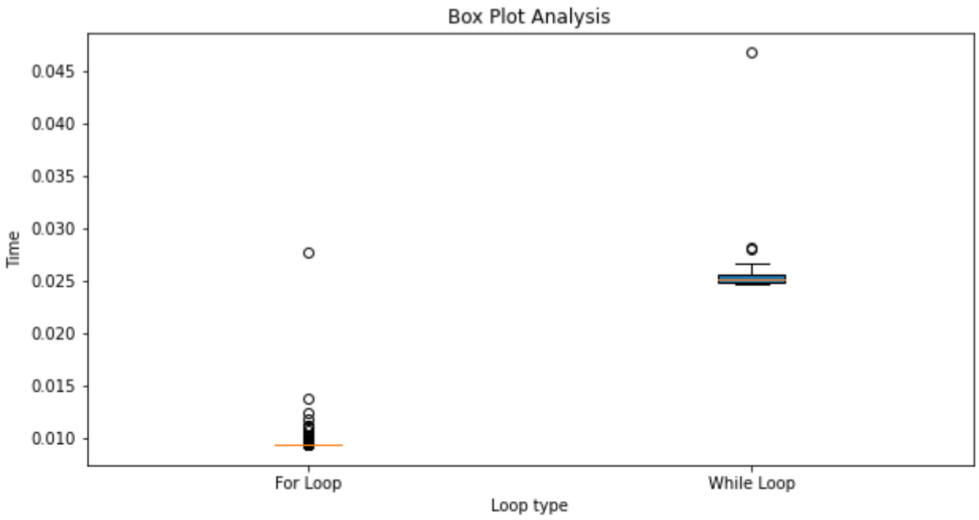
Moreover, the ideal sample size for this experiment is 360, based on power analysis for sample size.

2.4. Data Collection

In this topic, the necessary data for the test is collected. Thus, the code is run 360 times to generate samples for *for* and *while* loops. With this, it was possible to collect the data as follows:

Iteration type	Average time	Standard Deviation	Min. time	Max. time
For loop	0.009511	0.00103	0.009362	0.027730
While loop	0.025296	0.00122	0.024754	0.046745

Also, the following graph is a boxplot comparing two processing times:

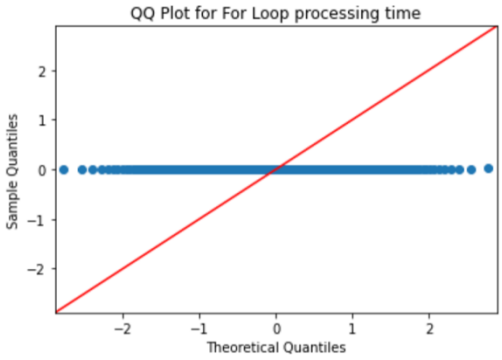


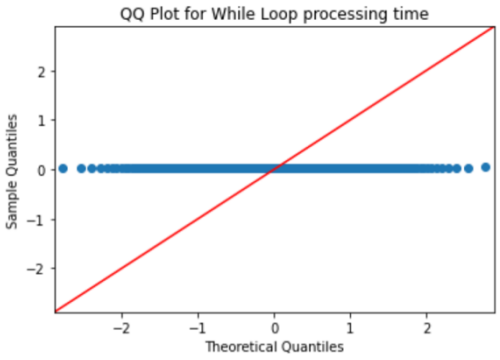
There are some outliers for both iterators. But, in this report, these points will be considered as a part of the analysis, since the real-time variation (in seconds) is very little. For example, in practice, the difference between the *While Loop* outlier of 0.046 and the mean value of 0.025 is 0.021, which means 2% of 1 second.

3. Testing and evaluation

Now, it is important to verify the necessary assumptions to conduct the Null Hypothesis Significance Testing.

The assumption of Independence was considered in the experiment design. The observations of samples are independent of each other. As for the assumption of normality, it will be tested with the *Q-Q Plot* [3] and the *Shapiro-Wilk* test [4].





As it is possible to note, the *Q-Q Plot* indicates that both iterators do not follow the normal distri bution. Thus, it was applied the *Shapiro-Wilk* test. For the *for loop*, the obtained p-value is 7.93e-38. For the *while loop*, the p-value is 1.33e-35. It means that **the assumption of normality is vio lated.**

Therefore, the *Wilcoxon rank sum test* [5] is conducted with the parameters set previously. Applying this test, it was possible to get p-value= 2.67e-118, which is much less than significance level $\alpha = 0.05$. In conclusion, with a sample size of 360, a significance level of 0.05, a power of 0.85, and an effect size of 0.20, **there is sufficient evidence to reject the null hypothesis**.

4. Conclusion

As discussed in the previous chapter, the null hypothesis is rejected with a significance level of 0.05. It means that the data indicates that *for loop* is faster than the *while loop* to count the numbers from 1 to 1 million.

There are some possible reasons for this. First, the *for loop* iterates with a known number of iterations. This leads to efficient iterations because it knows the initial, middle, and final conditions from the beginning. On the other hand, the *while loop* does not iterate with known numbers. Instead, it checks the condition every time it iterates. According to a test run by Benjamin Foreback from l3harris, the iteration through an array using the *while loop* took 26 times more than using a *for loop* to do the same task [6]. Although this information needs to be studied, many sources recommend the *for loop* to do a specific task than the *while loop* because of its efficiency. Finally, in assignment 1, it was also studied about this comparison. It is worthwhile to note that even though

the study in that assignment was more superficial, the *for loop* was faster to count the numbers than the *while loop* as well, in general.

5. References

[1] Aranha, Claus. 2022. Experiment Design in Computer Sciences. University of Tsukuba.

[2] GrabNGoInfo Amy. 2019. *https://pub.towardsai.net*. <*https://pub.towardsai.net/power-analysis-for-sample-size-using-python-33da7d5f570d*>

[3] Zach. 2020. *https://www.statology.org*. <*https://www.statology.org/q-q-plot-python/*>

[4] geetansh044. 2022. *https://www.geeksforgeeks.org*. <*https://www.geeksforgeeks.org/how-to-perform-a-shapiro-wilk-test-in-python/*>

[5] Czarniak, Elizabeth. 2022. *https://nathancarter.github.io/* <*https://nathancarter.github.io/how2data/site/how-to-do-a-wilcoxon-rank-sum-test-in-python-using-scipy/*>

[6] Foreback, Benjamin. 2016. *https://www.l3harrisgeospatial.com* <*https://www.l3harrisgeospatial.com/Learn/Blogs/Blog-Details/ArtMID/10198/ArticleID/15332/What-Type-of-Loop-Should-I-Use*>