camelCase vs kebab-case

Experiment 2, Experimentation & Evaluation 2024

Rossinelli Luca, Elisei Giovanni

Abstract

The following experiment aims to measure which style of function and variable naming between camel case and kebab case is best when it comes to the subject of coding. The performance of the two styles has been measured in the speed at which a certain name is recognized as correct when written in either of the two styles. The experiment has been conducted through a purposely developed webapp that registered the timing and some personal information about each subject that took part in the experiment. Then, all the data collected from subjects was used to produce a fairly accurate esteem of which of the two naming styles would produce a higher quality code in terms of readability.

1. Introduction

When it comes to writing code, naming your functions and variables in a comprehensible way is good for a long list of reasons, likely starting with easing code readability both for the one who developed the code and those who will have to review it or reimplement parts of it if and when the code is shared. Regarding this topic, there is an age-old debate ongoing when it comes to which style should be used conventionally to write the names of functions and variables. Among the long lists of candidates, this experiment aims to compare the performance of the two most used styles: camel case and kebab case. When speaking of performance, this experiment will cover the speed at which a certain name is recognized as correct when written in either of the two styles.

Hypotheses:

Write down your (falsifiable!) hypotheses here. Each hypothesis must include **independent** and your **dependent** variables. You must write down your hypotheses **before** you do your experiment!

- Kebab case will have a lower average time spent than camel case because it can be easier to recognize it since it has physical divisions among words
  (IV: Provided style) (DV: Time spent on each input)

- Camel case will have in general more mistakes compared to kebab case
  (IV: Provided style) (DV: Mistakes made for each input)

- The correct to mistake ratio will be higher for camel case, because of the absence of physical separation between the various words
  (IV: Provided style) (DV: Mistakes made for each input)

- More mistakes will be made when it comes to kebab case rather than camel case for more knowledgeable programmers

(IV: Provided style) (DV: Mistakes made for each input)

- Expertise level could not be as influential in the time spent as it could seem, since this is simply recognizing names
  (IV: Provided style) (DV: Time spent on each input)

2. Method

2.1 Variables

Explicitly identify the independent variable(s) (i.e., what you as the experimenter manipulate):

Independent variable	**Levels**
Style	Camel Case Kebab Case
Expertise	Never Coded Beginner Intermediate Expert
Length of the input	2 words 3 words 4 words

Dependent variable	**Measurement Scale**
Time spend on each input	Seconds / milliseconds
Mistakes	Integer

Control variable	**Fixed Value**
Input text	The provided text for each user is always the same, it is only shuffled the order in which the various choices are displayed in the multiple choice.

Blocking variable	**Levels**
Expertise	Never Coded Beginner Intermediate Expert

2.2 Design

Type of Study (check one):

☐ **Observational Study**	☐ **Quasi-Experiment**	x **Experiment**

Number of Factors (check one):

☐ **Single-Factor Design**	x **Multi-Factor Design**	☐ Other

Between vs. Within (check one):

☐ **Between Group Design** (independent measures)	x **Within Subject Design** (repeated measures)	☐ Other

(1) What we conducted was an experiment, since we manipulated the variable (naming style) and measured its effect on the subject to which the experiment was proposed. Also, considering the factors that were taken into consideration (expertise and naming style) , this can be considered to be a multi-factor experiment.

(2)

Never Coded	Camel case Kebab case
Beginner	Camel case Kebab case
Intermediate	Camel case Kebab case
Expert	Camel case Kebab case

(3) The design of the experiment is within-subject, since all of the selected participants were presented with the same set of names to provide a naming for. Therefore, although the order of the questions was randomized to avoid repetition, the actual name did not change and all of the subjects were proposed the same experiment.

2.3 Participants

We asked mostly our friends in the classroom to take part in the experiment. So the vast majority of the participants are male, with an age between 20 and 25 years old and already intermediate/ expert when it comes to coding. Some of the participants that we took could be considered outliers, as we chose a few friends from other faculties.
All the participants were recruited by asking (kindly) whoever was in class or study room to take part of the experiment.

We (Luca and Giovanni) dirtied our hands, and were part of the control group, while the rest of the recruits were the experimental group.

2.4 Apparatus and Materials

The props we used for this experiment were:

- Our laptop, as the webapp was run locally, and as it is easier to convince people to do something when asking in person.
- React (version 19.0.0) and Python (version 3.12).
- PyCharm to write and run the plotter.
- To measure the time we measured everything in the frontend using intervals.

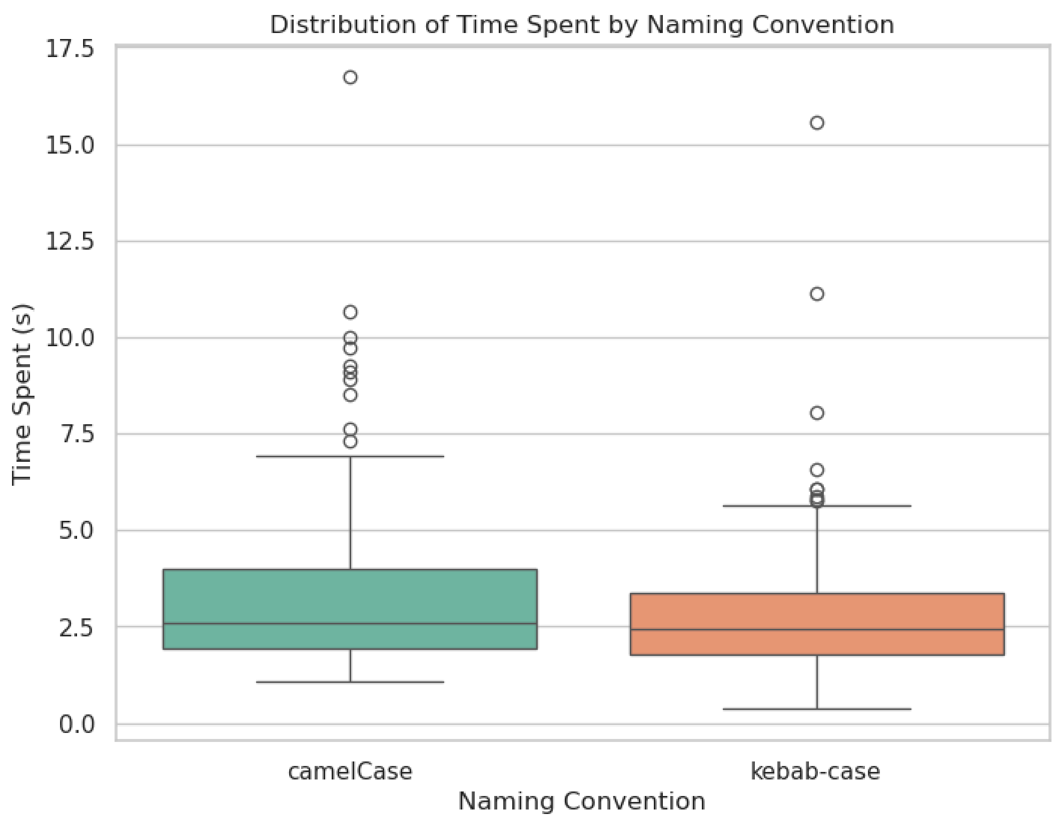
2.5 Procedure

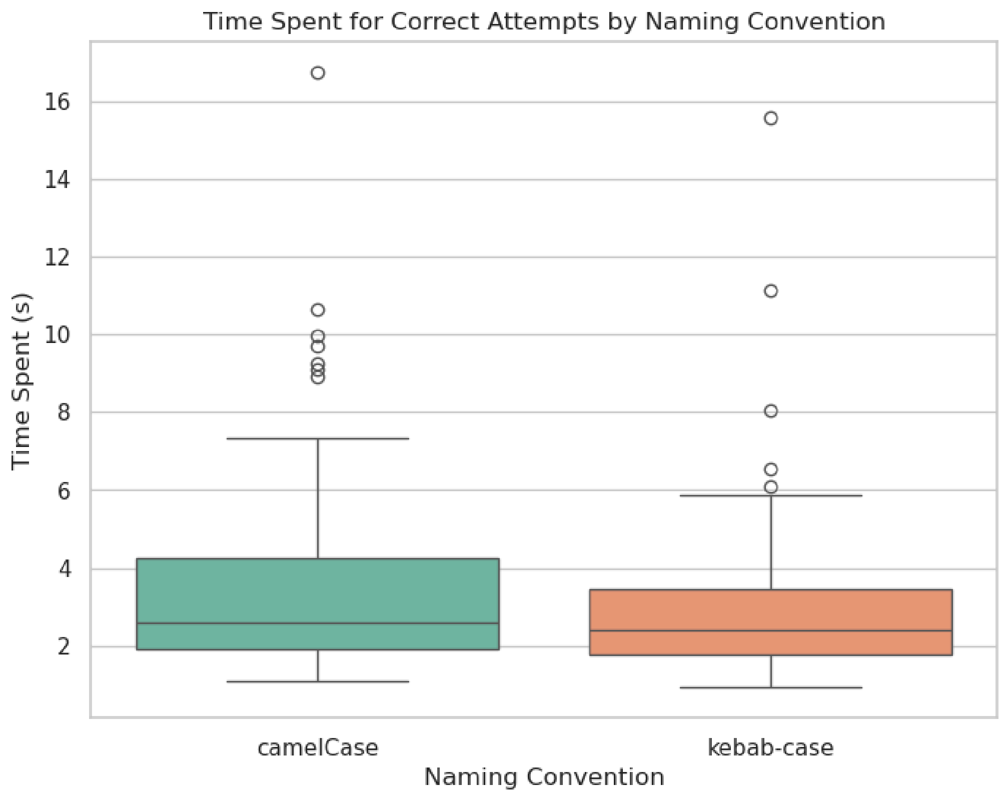
To carry out the experimentTo carry out the experiment we selected 16 people belonging to various groups of coding expertise. This ultimately meant a wilder and more uniform distribution in our results.
The participant was handed the laptop (because of how the webapp was developed, the laptops used were always the same: Luca's or Giovanni's) and was presented with the main instruction screen, which instructed them of all they needed to know on how the experiment would take place. Finally, when the participant finished reading the instructions, they began the experiment by clicking "Start" at the bottom of the instructions page.
The experiment itself was conducted by presenting a "preparation screen" to the participant, which contained a brief instruction on what to do (i.e. "read the name carefully and remember it"), the name to remember (i.e. "back down") and a "Ready" button which they had to click once they memorized the name.
Once the button was clicked, the measurement part of the experiment began: the subject was presented with 3 mistakenly spelled options and 1 correct one. The user had to choose the correct one as soon as they recognized it. Once one of the options was selected, the time was taken by calculating the endTime - startTime interval and whether the selected option was correct or not was registered. The experiment continued in this fashion until completion. At the end, the subject was shown their results and their answers were saved to a .csv file that was ultimately moved into the correct folder by hand by the developer (this approach was chosen because of simplicity of use, repeatability and the lack of necessity for a database). Finally, by clicking a button, the internal variables of the webapp were resetted and a new participant was subjected to the experiment.

3. Results

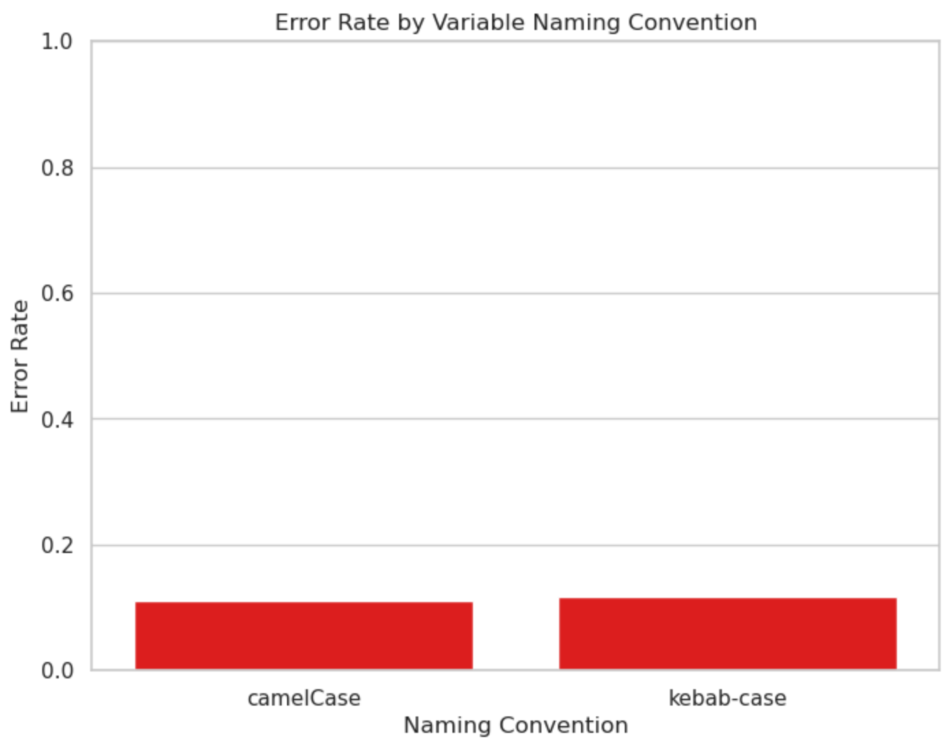
3.1 Visual Overview

As an overview, we decided to first show the boxplot of how much time was needed for each of the two naming conventions, both for only correct answers or for any type of answer.

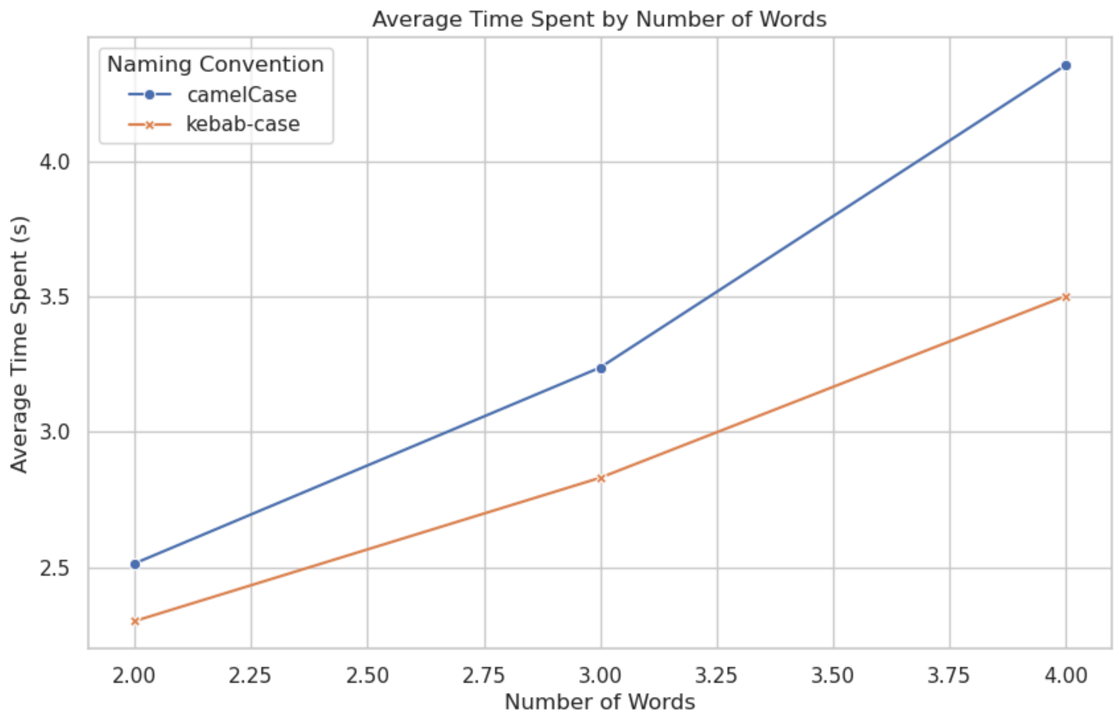




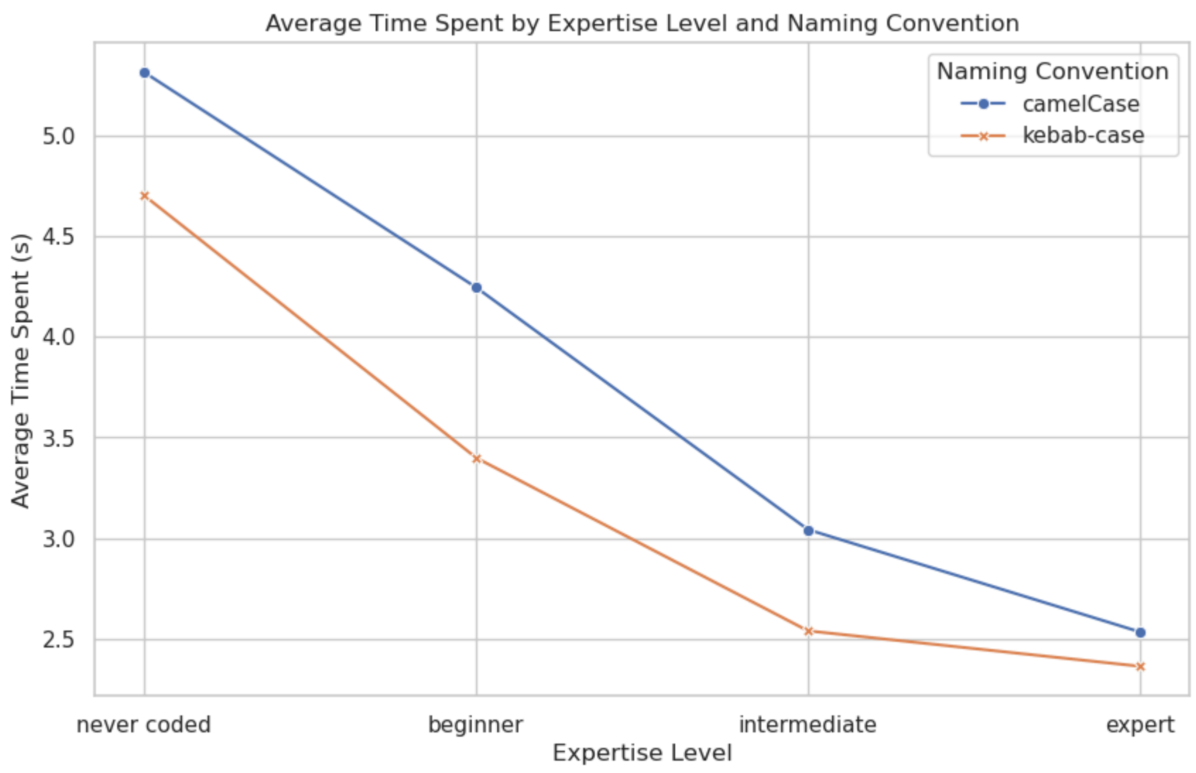
Additionally we plotted the rate of error depending on the naming convention:



Then how the naming conventions scale with the number of words:



And lastly how the expertise impacts on the time:



3.2 Descriptive Statistics

To explain this statistic, for example with the time per type, it means:

minimum	the smallest observed time, for that naming convention,
first quartile	25% of recorded times are below this number.
median	the middle value, which separates the lower and upper halves of the data.
third quartile	75% of times are below this number.
maximum	the largest observed time.

The data we have for it is:

```
Overall Descriptive Statistics:
             min   25%   50%   75%    max  mean   std
Type
camelCase   1.08  1.92  2.58  4.02  16.73  3.37  2.28
kebab-case  0.36  1.77  2.45  3.36  15.59  2.88  1.84
```

While with the Error Rates by Naming Convention it's straightforward as:

- camelCase Error Rate: ~10%
- kebab-case Error Rate: ~10%

Similarly the rest of the graphs are pretty straightforward, as they show how the time changes in function to the length of the sentence or to the expertise level.

3.3 Inferential Statistics

Cohen's d

To explain shortly, Cohen's d measures the effect size, or how large the difference between the two groups is in terms of standard deviations.

The formula for it is:

$d = (x1 - x2) / S$
$S = \sqrt{}[(s1\text{^}2 + s2\text{^}2) / 2]$

$x1 = 3.37\ (camelCase)$
$x2 = 2.88\ (kebab - case)$
$s1 = 2.28\ (camelCase\ standard\ deviation)$
$s2 = 1.84\ (kebab - case\ standard\ deviation)$

Therefore we get
$S = 2.07$
$d = 3.37 - 2.88 / 2.07 = 0.237$

This represents a small effect size, suggesting a modest difference in readability between camelCase and kebab-case.

Now lets calculate the confidence, the confidence interval provides a range within which the true mean difference is likely to fall.

Calculating the it, we get:
$Mean\ Difference = 3.37 - 2.88 = 0.49$
$SE(Standard\ error) = \sqrt{}[\ (2.28\text{^}2) / 144 + (1.84\text{^}2) / 144\ ] = 0.285$

Therefore we get $tValue = 1.97$

$Margin\ of\ Error = 1.97 \cdot 0.285 = 0.561$
$Confidence\ Interval = 0.49 \pm 0.561$

And we get:

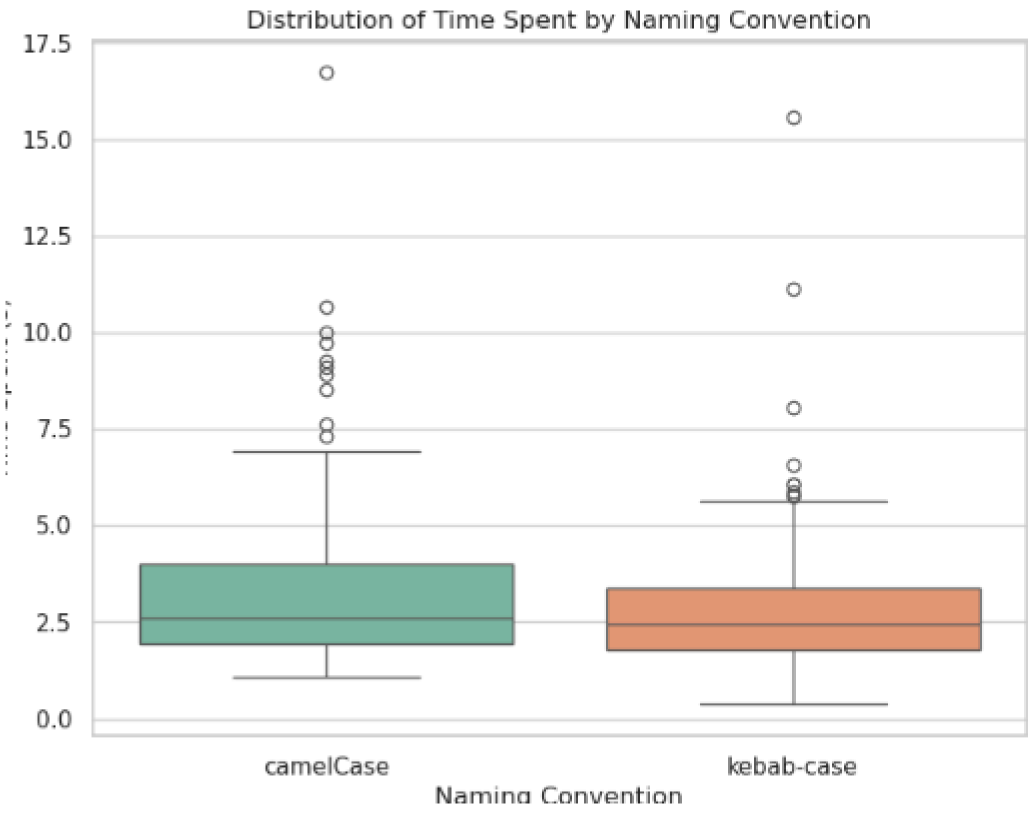
$$Confidence\ Interval = (0.49 - 0.561,\ 0.49 + 0.561) = 0.49 \pm 0.561$$

The 95% confidence interval is $(0.011, 0.973)$.
This means the true difference in readability times between camelCase and kebab-case is likely between 0.011 and 0.973, favoring kebab-case.

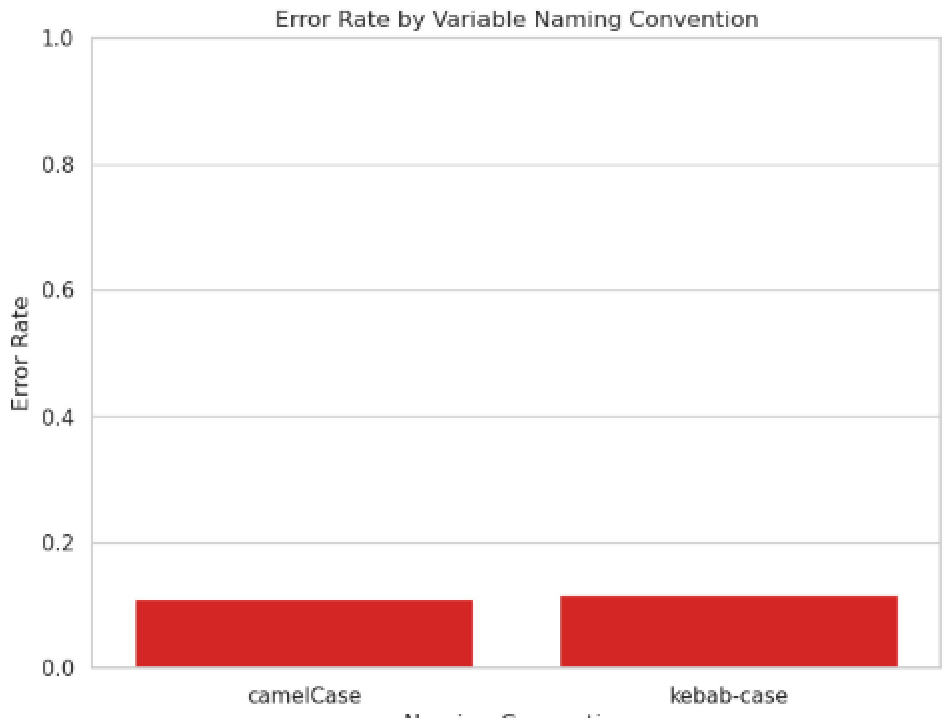
4. Discussion

4.1 Compare Hypothesis to Results

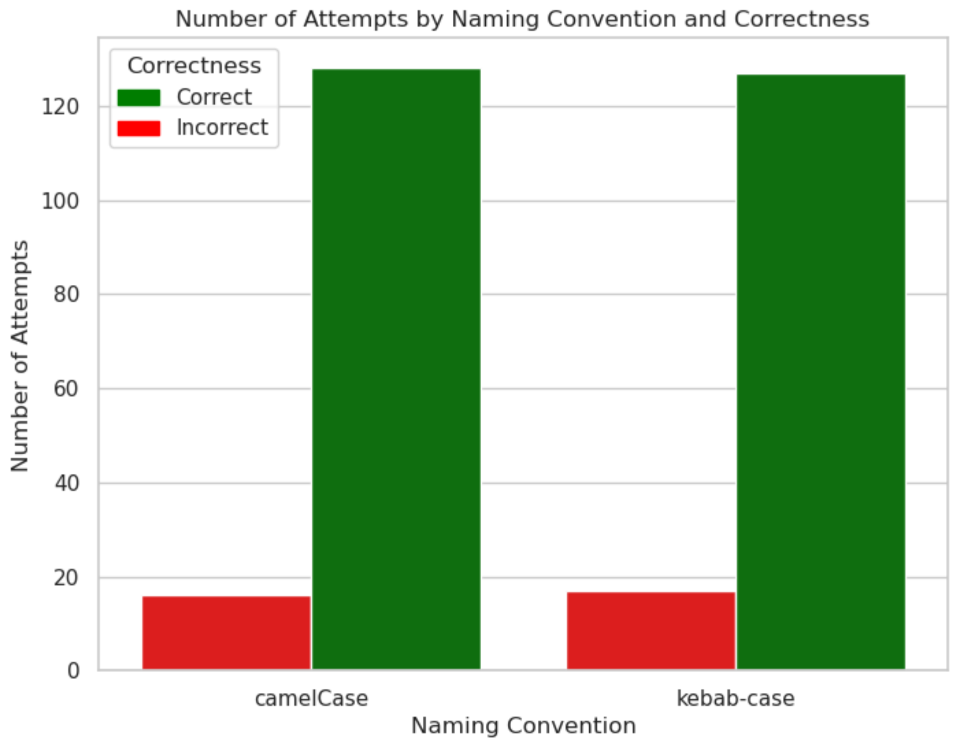
(1) **Kebab case will have a lower average time spent than camel case because it can be easier to recognize since it has physical divisions among words.** This was revealed to be true, since the candles here shown are a clear display of this. Although this is true, it cannot be determined with 100% certainty that it was due to the fact that each word had a physical separator. This considered, it can be definitely derived that this aspect played an important role in the time spent for each proposed name, since it was confirmed by some of the test subjects as being one of the reasons for why they chose kebab case on average while coding.



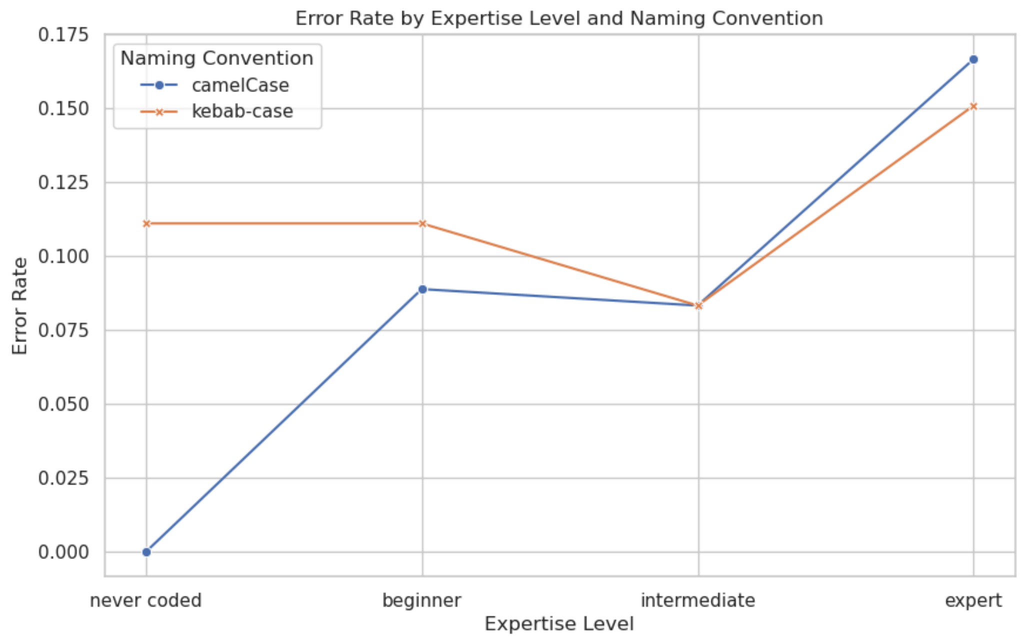
(2) **Camel case will have in general more mistakes compared to kebab case.** This was revealed to be false. The test subjects scored a higher percentage of correct inputs for camel case than kebab case as shown in the graph here below. This is likely due to the fact that, being more difficult to read, the subjects spent more time checking the accuracy of camel case rather than kebab. This ultimately resulted in a lower percentage of mistakes on kebab case.



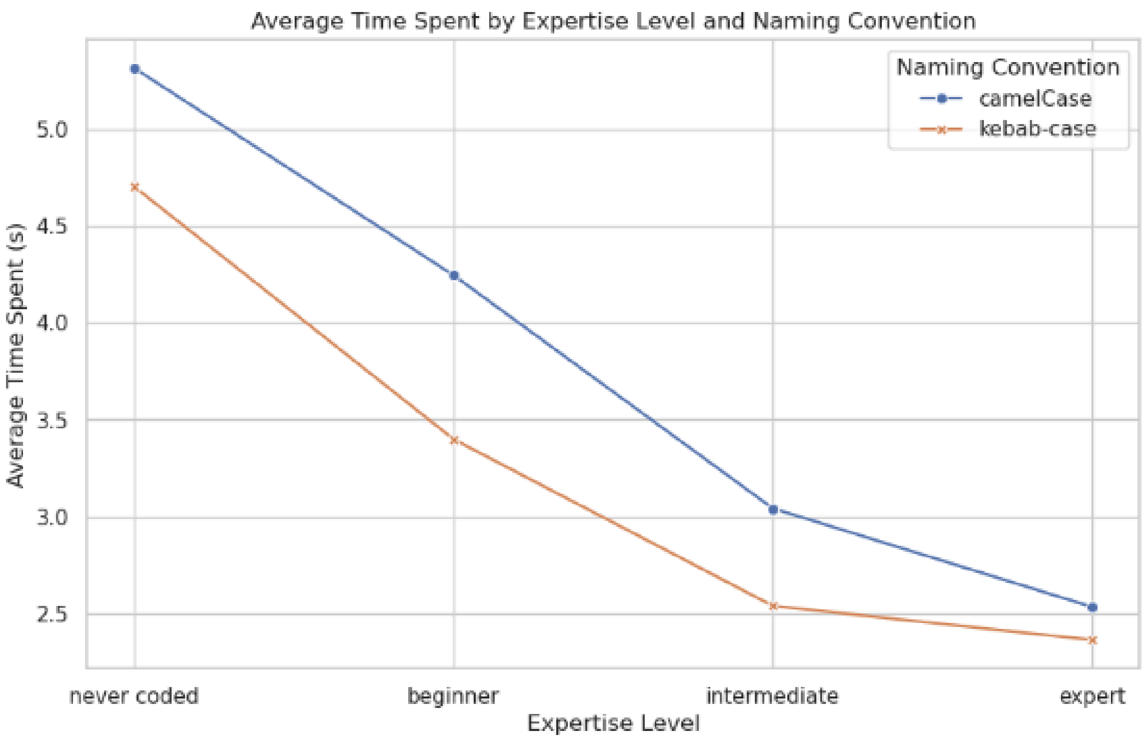
(3) **The correct to mistake ratio will be higher for camel case, because of the absence of physical separation between the various words.** Like the previous hypothesis, this was revealed to be false. And again, although this couldn't be proven with 100% certainty, it is likely due to the fact that, being harder to read, subjects spent more time checking camel case for correctness, resulting in a more accurate choice.



(4) **More mistakes will be made when it comes to kebab case rather than camel case for more knowledgeable programmers.** This surprisingly was proven to be wrong. Although it is not clear to why this is true, we can speculate that the more expert subjects were led by a false sense of security given the previous experience. This meant a lower time spent on each name and likely resulted in a higher percentage of mistakes.



(5) **Expertise level could be less influential in the time spent than it could seem, since this is simply recognizing names.** When it comes to the speed at which the test subjects completed the experiment, it would seem that with the level of expertise, the confidence of the subject increased, ultimately resulting in a lower time spent on each proposed name. But, as seen for the percentage of mistakes, this also seemed to result in a higher percentage of mistakes



4.2 Limitations and Threats to Validity

Two main threats were spotted when it comes to the validity of the obtained results in our experiment:

- Experiment length
- Number of subjects

In the first instance, it was recognised that the number of names that were presented (namely 18: 6 camel case, 6 kebab case). The limited number of names makes it so that the "luck factor", meaning that there could be the possibility that the chosen names were a particular case so that it was harder to read in a particular style, is non-zero. This could be avoided by increasing the number of test cases, although, through the experience of other groups, we did find that though there is a lower bound for validity, there is also an upper bound, in the sense that being the test subjects human, providing 60 questions for each case introduced a new factor in the equation: subject tiredness. The subject experienced mental fatigue and this increased the percentage of errors in both cases. Given this externally acquired knowledge, we estimated that in another iteration of the experiment, probably around 30 names for each of the styles could be a valid amount to test for efficiency.

For the second issue, the problem is merely the amount of participants that were chosen to run the experiment. Although the obtained results after running the experiment through 16 subjects can already be considered satisfying, it is undeniable that more test subjects would have produced a more accurate range of results that better reflected the actual efficiency of the two naming styles. An already more acceptable number of participants would have been around 25 to 30 and, although this number is based on almost nothing, we thought this is one of the cases where "the more the merrier" is true.

4.3 Conclusions

Drawing the conclusion, two main aspects of the experiment can be extrapolated from the experience:

- Camel case, in terms of code readability, turns out to be the worst of the two as shown by the graph that relates correct and mistakes. It is harder to read and therefore makes for less understandable code on a first read. Kebab case, on the other hand, seems to be faster to read, in the sense that the general meaning of the words is understood faster.

- Also, as seen from the graph relating the mistakes made between the two styles, it can be derived that, although a difference can be seen in the mistakes made in either of the two, the difference is minimal and the two styles can be considered equal under that aspect.

Therefore, if one is aiming strictly for easily readable code, kebab case seems to be the better option compared to camel case, since it is faster to read and makes for better overall code readability.

Appendix

Reproduction Package (or: Raw Data)

The collected data has been produced by 16 different people, it’s stored in the “result” folder. The data is then used by plotter.ipynb to create different visualisations.
When running the experiment, once it is completed, the result is saved on the computer. It is then manually moved to the results folder.
To run the experiment, open the folder in the terminal, run the initialization **npm install** and **npm start,** the experiment should open on its own.