Hash-Table

Building a Metaphorical Hash Table

Irving F. Sanchez

Lewis University

Course Number: *SP25-CPSC-34000-002*

Professor: David Nowak

Building a Metaphorical Hash Table

Before diving into the technical journey, it’s important to share the intention behind how I approach coding projects such as this one. I write code as if I’m teaching it to someone who has little to no experience with programming—not because I assume they can’t understand it, but because I believe most confusion in math, science, and computer science stems from unnecessary overcomplication. My goal each time I write code is to make concepts like hash tables approachable, relatable, and even a little fun ‘maybe’. Every line I write is an opportunity to build a bridge—not just between variables and functions, but between learners and ideas. I want my code to do more than just earn a grade. I want it to be something that could help someone, a high school student exploring engineering, a curious adult learning to code for the first time—feel like this world is accessible to them too. This has always been a part of my mission apparatus, ever since I learned about my complications in a learning environment.

The goal of this project was to implement a hash table from scratch using Python, emphasizing the use of the built-in function and collision handling through chaining. Rather than constructing the solution in a purely technical manner, I approached the design with the intent of making it accessible to learners who may not have a background in computer science. Inspired by my own challenges in traditional learning environments, I chose to center the project around a metaphor—the “Brain Filing System”—where each key is a thought and each value a memory, stored in one of several mental drawers. This was also inspired by the movie Inside Out which revolves around a young girl and how her thoughts are managed within her growing mind.

One of the first insights I encountered during development was the behavior of Python’s function. Initially, I assumed the drawer (index) for each thought would be consistent across program runs. However, I observed that the same input was being stored in different drawers depending on when the program was executed. This led to research and discovery of Python’s hash randomization, a security feature that intentionally changes hash values across sessions (Heimes, 2013). While confusing at first, understanding this behavior helped me appreciate the variability of hash functions in real-world environments and how indexing must account for such fluctuations.

To meet the assignment’s requirements for collision handling, I implemented chaining by using a list at each drawer. This allowed multiple key-value pairs to coexist in the same drawer if their hash values collided. To demonstrate this functionality, I ran a test using a brain initialized with only two drawers—significantly increasing the likelihood of collisions. The resulting output showed that distinct thoughts were properly stored together within a single drawer, confirming that my implementation managed collisions as intended.

Throughout this process, I remained focused on clarity and accessibility, adding detailed comments, labeled code blocks, and human-readable method names like and . This approach not only reinforced my own understanding but created a resource that others could learn from. In the end, this project became more than just an implementation of a data structure—it became a reflection of how metaphor, empathy, and storytelling can make technical concepts approachable and meaningful. Let the record show that I always struggle with all of my assignments—if I could explain how many thoughts run in my head on a daily basis, we’d need the millennium falcon to catch up to my thoughts.

# References

Dojo, C. (2017, June 25). *www.youtube.com*. Retrieved from https://www.youtube.com/@KindsonTheGenius: https://www.youtube.com/watch?v=KT8UbevKq6s

GeeksForGeeks. (2025, March 04). *www.geeksforgeeks.org*. Retrieved from https://www.geeksforgeeks.org/separate-chaining-collision-handling-technique-in-hashing/

Heimes, C. (2013, October 06). *peps.python.org*. Retrieved from https://peps.python.org/pep-0456/