1. The most classical example will be “eat” and “ate”. While hash_function_1() return as 314 for both values, but hash_function_2(“eat”) = 643 and hash_function_2(“ate”) = 632. Because hash_function_1() is just the sum of the three letters from the ASCII code while hash_function_2() is sum of the letters but multiplying the position by a different value.

    a. Hash_function_1()
        i. eat: e = 101, a = 97, and t = 116. 101 + 97 + 116 = 314
        ii. ate: a = 97, t = 116, e = 101. 97 + 116 + 101 = 314
    b. Hash_function_2()
        i. eat: e = 101 * 1 = 101, a = 97 * 2 = 194, and t = 116 * 3 = 348. 101 + 197 + 348 = 643
        ii. ate: a = 97, t = 116 * 2 = 232, and e = 101 * 3 = 303. 97 + 232 + 303 = 632
2. hash_function_1() has more likelihood to have sane key yet different values which might have collision while hash_function_2() is less likelihood to have a collision between key values.
3. No, there is no difference between hash_function_1() and hash_function_2() because both functions (empty_bucket() and table_load()) does not use the hash_function to calculate the hashkey value. It only uses the self.capacity, and self.size of the bucket which doesn’t related to the hash_function values.
4. A prime number’s factor is 1 and itself. Which means the hash table will be distributed evenly compare to even number. Because the calculation of adding into the hash table is the remainder of hash_function for a certain key / table size, which if the table size is a prime number, the remainder will be between 1 to the prime number value – 1. Which every bucket will be filled while even number, sometimes, will be filled. Therefore, the difference between prime number and even number of table size that the prime number will have less (no) empty bucket while even number has some (at least 1) empty bucket.