1. Zyra the code mage has hidden a mysterious cipher in reversed messages. You must help Zyra uncover the secrets of the digital realm. Create a function called *reverse_string* that takes the variable *word* (a string) and returns the word in reversed order.
   For this problem, you must use iteration (a loop) not slicing.

   Examples:

   - reverse_string("programming") → "gnimmargorp"
   - reverse_string("python") → "nohtyp"
   - reverse_string("hello") → "olleh"

2. The **normal human body temperature** is 98.6F in Fahrenheit and 37C in Celsuis. Create a function that determines if the *temp* is considered a fever(anove normal body temperature) or not. *temp* will be measured in Fahrenheit and Celsuis.
   Notice: The F or C will always be the last character in the string.

   Examples:

   - is_fever("99F") → True,
   - is_fever("37C") → False,
   - is_fever("98F") → False,

3. The **boiling point** of water is 212F in Fahrenheit and 100C in Celsuis. Create a function that determines if the *temp* is considered boiling or not. *temp* will be measured in Fahrenheit and Celsuis.
   Notice: The F or C will always be the last character in the string.

   Examples:

   - is_boiling("212F") → True,
   - is_boiling("100C") → True,
   - is_boiling("0F") → False,

4. The **hamming distance** is the number of characters that differ between two strings.
   To illustrate,
   str1 = "abcbba"
   str2 = "abcbda"
   The hamming distance is 1 since the only difference is the 5^{th} character.
   That is, "b" in str1 vs. "d" in str2.

   Your task: create a function named hamming distance that takes two strings as arguments, and returns the hamming distance between the two strings.

   Examples:

   - hamming_distance("abcde", "bcdef") → 5, since all 5 letters are different.
   - hamming_distance("abcdef", "abcdef") → 0, since all 6 letters are the same.
   - hamming_distance("strong", "strung") → 1, since there is only 1 character that is different.

5. An **Isogram** is a word that has no duplicate letters. Create a function that takes a string and returns either True or False depending on whether or not it is an "isogram". You may assume words will only have lower case letters.

   Examples:

- is_isogram("algorithm") → True
- is_isogram("password") → False (multiple of s)
- is_isogram("consecutive") → False (multiple of c)
- is_isogram("python") → True

6. A fruit juice company tags their fruit juices by concatenating the first **three letters** of the words in a flavor's name, with its capacity. Create a function that creates product IDs for different fruit juices. Notice that the first input is a string and the second is an integer.

   Examples:

   - get_drink_ID("apple", 500) → "app500"
   - get_drink_ID("pineapple", 45) → "pin45"
   - get_drink_ID("watermelon", 750) → "wat750"

7. Professor Dumbledore seeks to decipher powerful encoded spells in the Hogwarts Library, their secrets revealed by the first letter of each word. Create a function called *first_letters* that takes the variable *sentence* (a string) and returns a string made up of the first letters of each word in the sentence.

   Examples:

   - first_letters("wingardium leviosa makes objects float") → "wlmof"
   - first_letters("expecto patronum repels dementors") → "eprd"
   - first_letters("the magic is within you") → "tmiwy"

8. Severus Snape seeks to harness powerful spells in the Hogwarts Library, you must encode them by using the last letter of each word. Create a function called *last_letters* that takes the variable *sentence* (a string) and returns a string made up of the last letters of each word in the sentence.

   Examples:

   - last_letters("wingardium leviosa makes objects float") → "masst"
   - last_letters("expecto patronum repels dementors") → "omss"
   - last_letters("the magic is within you") → "ecsnu"

9. Write a function called *flip_flop* that takes a string as an argument and returns a new word made up of the second half of the word first combined with the first half of the word second.

   Examples:

   - *flip_flop*("abcd") → "cdab" (that is, "cd" then "ab" ... even length)
   - *flip_flop*("grapes") → "pesgra" (that is, "pes" then "gra" ... even length)
   - *flip_flop*("abcde")→ "decab" (that is, "de" then "c" then "ab" ... odd length)
   - *flip_flop*("cranberries")→ "rriesecranb" (that is, "rries" then "e" then "cranb" ... odd length)