Strings Cheat Sheet

\texttt{word[pos]}
- Return the character at position \texttt{pos}.
- Positions are counted starting at 0 for the first character and going up, or starting at -1 for the last character and going down.

\texttt{word[start:end], word[start:end:step]}
- Return the string of characters obtained by starting with the character at \texttt{start}, “counting” by \texttt{step}, and going to, but not including, the character at \texttt{end}; or, if starting at \texttt{start} and counting by \texttt{step} does not eventually lead to \texttt{end}, return the empty string.
- If \texttt{step} and the final colon are omitted (first form), then count by 1.
- If \texttt{start} is omitted: start with the first character, if \texttt{step} is positive; or start with the last character, if \texttt{step} is negative.
- If \texttt{end} is omitted: include the last character, if \texttt{step} is positive; or include the first character, if \texttt{step} is negative.

\texttt{word.index(sub), word.index(sub, pos)}
\texttt{word.find(sub), word.find(sub, pos)}
- If \texttt{pos} is not present (first form), return the smallest index in \texttt{word} where substring \texttt{sub} is found.
- If \texttt{pos} is present (second form), return the smallest index greater than or equal to \texttt{pos} where substring \texttt{sub} is found in \texttt{word}.
- If \texttt{sub} is not found: \texttt{index} produces an error, whereas \texttt{find} returns -1.

\texttt{word.replace(old, new)}
- Return a copy of \texttt{word} with all occurrences of substring \texttt{old} replaced by string \texttt{new}.

\texttt{len(word)}
- Return the length of \texttt{word}.

\texttt{word1 + word2}
- Return the string of characters made by concatenating a copy of \texttt{word1} and a copy of \texttt{word2} to form one string.

\texttt{word.lower()}
- Return a new string produced from \texttt{word} by converting all alphabet characters to lower case.

\texttt{word.upper()}
- Return a new string produced from \texttt{word} by converting all alphabet characters to upper case.
word1 < word2, word1 <= word2, word1 > word2, word1 >= word2
  • Compare characters in word1 and word2, starting from position 0; return True or False based on the relationship between the Unicode values of the characters at the first position where word1 and word2 differ.

word1 == word2, word1 != word2
  • Compare words for equality (==) or inequality (!=); return True or False.

word1 in word2, word1 not in word2
  • Check if word1 appears in word2 as a substring: the first expression returns True if it does, and False if it does not; the second expression returns False if it does, and True if it does not.

for c in word:
    suite

Repeatedly execute suite for each character in word; before each iteration, assign c the next character in word.