## Congratulations! You passed!

Grade received 100% To pass 80% or higher

Go to next item

The is\_palindrome function checks if a string is a palindrome. A palindrome is a string that can be equally read
from left to right or right to left, omitting blank spaces, and ignoring capitalization. Examples of palindromes are
words like kayak and radar, and phrases like "Never Odd or Even". Fill in the blanks in this function to return True
if the passed string is a palindrome, False if not.

1/1 point

Correct
 Woohoo! You're quickly becoming the Python string expert!

Using the format method, fill in the gaps in the convert\_distance function so that it returns the phrase "X miles
equals Y km", with Y having only 1 decimal place. For example, convert\_distance(12) should return "12 miles
equals 19.2 km".

1/1 point

```
1 def convert_distance(miles):
2 km = miles * 1.6
3 result = *() miles equals {:.1f} km*.format(miles, km)
4 return result
5
6 print(convert_distance(12)) # Should be: 12 miles equals 19.2 km
7 print(convert_distance(12)) # Should be: 5.5 miles equals 8.8 km
8 print(convert_distance(11)) # Should be: 11 miles equals 17.6 km
Reset

12 miles equals 19.2 km
Neset
```

Congrats! You're getting the hang of formatting strings, hooray!

3. If we have a string variable named Weather = "Rainfall", which of the following will print the substring or all characters before the "f"?

1/1 point

- print(Weather[:4])
- O print(Weather[4:])
- O print(Weather[1:4])
- O print(Weather[:"f"])

Correc

Nice job! Formatted this way, the substring preceding the character "f", which is indexed by 4, will be printed.

4. Fill in the gaps in the nametag function so that it uses the format method to return first\_name and the first initial of last\_name followed by a period. For example, nametag("Jane", "Smith") should return "Jane S."

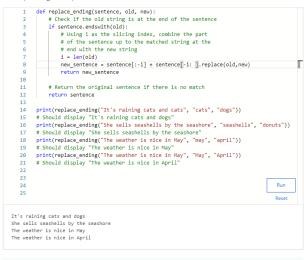
1/1 point

Correct

Great work! You remembered the formatting expression to limit how many characters in a string are displayed.

1/1 point

a. In ereptace\_ending function reptaces the old string in a sentence with the new string, out only if the sentence ends with the old string. If there is more than one occurrence of the old string in the sentence, only the one at the end is replaced, not all of them. For example, replace\_ending("abcabc", "abc", "xyz") should return abcxyz, not xyzxyz or xyzabc. The string comparison is case-sensitive, so replace\_ending("abcabc", "ABC", "xyz") should return abcabc (no changes made).



Outstanding! Look at all of the things that you can do with these string commands!