

Congratulations! You passed!

TO PASS 80% or higher

Keep Learning

GRADE 100%

Practice Quiz: Strings

TOTAL POINTS 5

1. 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

```
new_string = ""
reverse_string = ""
# Traverse through each letter of the input string
for letter in input_string:
# Add any non-blank letters to the
# end of one string, and to the front
# of the other string.
if letter != " ":
new_string + letter.lower()
reverse_string = letter.lower() + reverse_string
# Compare the strings
 11
 15 * if new_string == reverse_string:
            return True
return False
                                                                                                                                                                                                                                      Run
print(is_palindrome("Never Odd or Even")) # Should be True
print(is_palindrome("abc")) # Should be False
print(is_palindrome("kayak")) # Should be True
 True
 False
 True
```

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

2. 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):
           km = miles * 1.6
result = "{} miles equals {:.1f} km".format(miles,km)
                                                                                                                                                                            Run
print(convert_distance(12)) # Should be: 12 miles equals 19.2 km
print(convert_distance(5.5)) # Should be: 5.5 miles equals 8.8 km
print(convert_distance(11)) # Should be: 11 miles equals 17.6 km
print(convert_distance(11)) # Should be: 11 miles equals 17.6 km
                                                                                                                                                                           Reset
12 miles equals 19.2 km
5.5 miles equals 8.8 km
11 miles equals 17.6 km
11 miles equals 17.6 km
```

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

hooray!

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

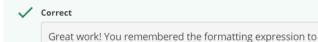
Nice job! Formatted this way, the substring preceding the character "f", which is indexed by 4, will be printed.

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

```
1  def nametag(ftrst_name, last_name):
2    return("{} {:.1s}.".format(first_name, last_name))
3
4    print(nametag("Jane", "Smith"))
5    # Should display "Jane S."
6    print(nametag("Francesco", "Rinaldi"))
7    # Should display "Francesco R."
8    print(nametag("Jean-Luc", "Grand-Pierre"))
9    # Should display "Jean-Luc G."

Jane S.
Francesco R.
Jean-Luc G.
```



limit how many characters in a string are displayed.

place and ing function raplaces the old string in a contance with the new string, but only if the

5. The replace_ending function replaces the old string in a sentence with the new string, but 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).

1 / 1 point

```
def replace_ending(sentence, old, new):
    # Check if the old string is at the end of the sentence
    if old[:] == sentence[-len(old):]:
    # Using i as the slicing index, combine the part
    # of the sentence up to the matched string at the
    # end with the new string
    is already
                i = len(old)
                 new_sentence = sentence[:-i] + new
 9
              return new_sentence
          # Return the original sentence if there is no match
          return sentence
12
13
14
15
      print(replace_ending("It's raining cats and cats", "cats", "dogs"))
# Should display "It's raining cats and dogs"
      print(replace_ending("She sells seashells by the seashore", "seashells",
        "donuts"))

# Should display "She sells seashells by the seashore"
print(replace_ending("The weather is nice in May", "may", "april"))

# Should display "The weather is nice in May"
print(replace_ending("The weather is nice in May", "May", "April"))

# Should display "The weather is nice in April"
17
19
20
                                                                                                                                                                                           Run
22
23
24
25
                                                                                                                                                                                          Reset
It's raining cats and dogs
She sells seashells by the seashore
The weather is nice in May
The weather is nice in April
```



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