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

```
def is_palindrome(input_string):
    # We'll create two strings, to compare them
    new_string = ""
    reverse_string = ""
    # Traverse through each letter of the input string
    for x in input_string.lower():
        # Add any non-blank letters to the
        # end of one string, and to the front
        # of the other string.
        if x.strip():
            new_string = new_string + x
            reverse_string = x + reverse_string
    # Compare the strings
    if new_string == reverse_string:
        return True
    return False

print(is_palindrome("Never Odd or Even")) # Should be True
print(is_palindrome("abc")) # Should be False
print(is_palindrome("kayak")) # Should be True
```

Run
Reset

```
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

```
def convert_distance(miles):
    km = miles * 1.6
    result = "{} miles equals {:.1f} km".format(miles, km)
    return result

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
```

Run
Reset

```
12 miles equals 19.2 km
5.5 miles equals 8.8 km
11 miles equals 17.6 km
```

Correct

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])
- ( ) 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

```
def nametag(first_name, last_name):
    return("{} {:.1s}.".format(first_name, last_name))

print(nametag("Jane", "Smith"))
# Should display "Jane S."
print(nametag("Francesco", "Rinaldi"))
# Should display "Francesco R."
print(nametag("Jean-Luc", "Grand-Pierre"))
# Should display "Jean-Luc G."
```

Run

Reset

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

Correct

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

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 sentence.endswith(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
        sentence_lst = sentence.split()
        new_sentence = ' '.join(sentence_lst[:-1] + [new])
        return new_sentence

    # Return the original sentence if there is no match
    return sentence

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"


```

Run

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
```

Correct

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