[Jahangirnagar University](https://www.coursera.org/programs/jahangirnagar-university-na-qyc9d)

[Skip to Main Content](https://www.coursera.org/learn/python-crash-course/quiz/sDmOq/practice-quiz-strings/attempt?redirectToCover=true#main)

* 

MD. TAZEL HOSSAN

[Crash Course on Python](https://www.coursera.org/learn/python-crash-course/home/welcome)

[Week 4](https://www.coursera.org/learn/python-crash-course/home/week/4)

Practice Quiz: Strings

[Prev](https://www.coursera.org/learn/python-crash-course/supplement/JbXSA/formatting-strings-cheat-sheet)

[Next](https://www.coursera.org/learn/python-crash-course/lecture/DoC7j/what-is-a-list)

**PRACTICE QUIZ • 25 MIN**

**Practice Quiz: Strings**

**Submit your assignment**

Try again

**Receive grade**

**TO PASS**80% or higher

**Grade**

100%

View Feedback

We keep your highest score

Practice Quiz: Strings

Practice Quiz • 25 min

**Congratulations! You passed!**

**TO PASS**80% or higher

Keep Learning

**GRADE**

100%

**Practice Quiz: Strings**

**TOTAL POINTS 5**

1.Question 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**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

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 i in input\_string:

        # Add any non-blank letters to the

        # end of one string, and to the front

        # of the other string.

        if i!=" ":

            new\_string += i.lower()

            reverse\_string = i.lower()+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





RunReset

**Correct**

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

2.Question 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

2

3

4

5

6

7

8

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





RunReset

**Correct**

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

hooray!

3.Question 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.Question 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

2

3

4

5

6

7

8

9

def nametag(first\_name, last\_name):

    return("{} {}.".format(first\_name,last\_name[0]))

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."





RunReset

**Correct**

Great work! You remembered the formatting expression to

limit how many characters in a string are displayed.

5.Question 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**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

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

        i = sentence.rfind(old)

        new\_sentence = sentence[:i]+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"





RunReset

**Correct**

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

these string commands!