Learning Quiz 8: More Strings

Due Oct 16 at 1pm **Points** 5 **Questions** 5 **Available** until Dec 4 at 11:59pm
Time Limit None **Allowed Attempts** Unlimited

Instructions

Prior to completing this quiz, you should have read:

- Section 4.1, Strings Revisited (p. 92-98)

Please also go over Practice Problems 4.1 and 4.2 in the textbook (solutions at the end of the chapter) before attempting this quiz.

This quiz was created for learning purposes. You may attempt this quiz as many times as you would like. The highest score *prior to the deadline* will count towards the final course grade. No late submissions will be accepted.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	24 minutes	4 out of 5

Score for this attempt: **4** out of 5
Submitted Oct 15 at 2:49pm
This attempt took 24 minutes.

Question 1 — 1 / 1 pts

In Python, you can create strings using either single quotes ('---') or double quotes ("---"). Within the strings, you can insert various characters, such as alphabets, numbers, spaces, and SOME special symbols. What if we want to insert other special characters in our string, such as single quotes, double quotes, backslashes, tabs, new lines? We can use the escape character (\) to access special characters in Python.

For example:

```
a = "this is what \t tab does"
print(a)
```

Note, rather than printing \t, Python prints a tab where we see \t in the string.

Test out the following characters below in Python and match the escape sequence with the special character.

Correct!	****	backslash
Correct!	**\'**	single quote
Correct!	**\"**	double quote
Correct!	**\t**	tab
Correct!	**\n**	new line
Correct!	**\b**	backspace

Question 2

1 / 1 pts

As a shortcut, for single or double quotes, you can incorporate them by using the other type of quotes to create your string. Consider the following examples:

```
a = 'The lecture was so much "fun".'
b = "I'll be there in one hour!"
```

Notice, that by creating variable a with single quotes, I could freely use double quotes in my string. Similarly, by creating variable b with double quotes, I could freely use single quotes in my string.

I will need to use the escape character if I want to include single quotes in a string created by single quotes.

```
c = 'Ana is 5\'3" tall.'
print(c)
```

What is another way to create variable c? Select all.

- [ ] c = 'Ana is 5'3\" tall.'
- [x] c = "Ana is 5\'3\" tall." Correct!
- [x] c = 'Ana is 5\'3\" tall.' Correct!
- [x] c = "Ana is 5'3\" tall." Correct!
- [ ] c = "Ana is 5\'3" tall."

Question 3

0 / 1 pts

If you have multi-line strings, you can either use triple quotes (single or double) or "\n" as follows:

```
poem1 = '''Roses are red,
Violets are blue,
Sugar is sweet,
And so are you.'''

poem2 = 'Roses are red,\nViolets are blue,\nSugar is sweet,\nAnd so are yo
u.'
```

Another special character in Python is the \ at the end of a line, which forces Python to IGNORE the new line in your code.

```
print("line 1 \
and line 2 \
will print in the same line"
```

Consider the following lines of code:

```
seuss = '''
One fish
Two fish
Red fish
Blue fish'''
```

Which of the following will create the exact same **seuss** variable?

ou Answered

- [x] `seuss = 'One fish\nTwo fish\nRed fish\nBlue fish'`

- [ ]
```
seuss = "
One fish
Two fish
Red fish
Blue fish"
```

ou Answered

- [x]
```
seuss = """
One fish
Two fish
Red fish
Blue fish
"""
```

Correct!

- [x] `seuss = '\nOne fish\nTwo fish\nRed fish\nBlue fish'`

Question 4

1 / 1 pts

Make sure you are familiar with the string methods listed in Table 4.1 (page 97).

The string method s.replace(old, new) will replace every instance of the old string with a new string. However, since strings are immutable, mystring will not automatically update. For example:

```
mystring = 'I brought a red T-shirt with me to the game. The fans hated my r
ed shirt.'
mystring.replace("red", "yellow")
print(mystring)      ## This still says "red"
mystring = mystring.replace("red", "yellow")
print(mystring)      ## This now says "yellow"
```

Consider the following lines of code.

```
funtimes = 'Programming in Python is fun! I like to write functions.'
funtimes = funtimes.replace('fun', 'easy')
print(funtimes)
```

What prints?

- [ ] Programming in Python is easy! I like to write functions.

Correct!

- [x] Programming in Python is easy! I like to write easyctions.
- [ ] Programming in Python is fun! I like to write functions.
- [ ] Programming in Python is ! I like to write ctions.
- [ ] Programming in Python is ! I like to write functions.

Question 5

1 / 1 pts

Make sure you are familiar with the string methods listed in Table 4.1 (page 97).

The string method s.translate(table) will convert letters in your string based on a given table. Consider the following translation table tocryp which translates 'a' to 'x', 'b' to 'y', and 'c' to 'z'. In our term 'banana' both 'a' and 'b' will convert, but 'n' will remain the same. Again, since strings are immutable, we need to manually replace mystr to update it with the translation:

```
mystr = 'banana'
tocryp = mystr.maketrans('abc','xyz')   ## Note, 'abc' and 'xyz' must be the
same length
mystr = mystr.translate(tocryp)     ## mystr will not automatically update u
nless we replace it
print(mystr)
```

Consider the following lines of code.

```
newstr = 'I like to program in Python'
newcryp = newstr.maketrans('abcdefghijklmnop','!@#$%^&*()_QWERT')
print(newstr.translate(newcryp))
```

What prints?

- [ ] I q(_% tr trr&r!w (e Pyt*re
- [ ] I l(_% to pro&r!m (n pyt*on
- [ ] I l(_% to pro&r!m (n Pyt*on
- [ ] I Q(_% tR TrR&r!W (E Tyt*RE
- [x] I Q(_% tR TrR&r!W (E Pyt*RE

Correct!

Quiz Score: **4** out of 5