

Curso > Week 1... > 2. Core... > Exercis...

Audit Access Expires 5 de ago de 2020

You lose all access to this course, including your progress, on 5 de ago de 2020.

Exercise 2

Finger Exercises due Aug 5, 2020 20:30 -03 Completo

NOTE: These exercises are ungraded.

Exercise 2 part 1

0 points possible (ungraded)

ESTIMATED TIME TO COMPLETE: 3 minutes

Note that you will have to answer all questions before you can click the Check button.

For each of the expressions below, specify its type and value. If it generates an error, select type 'NoneType' and write the word 'error' (note this is a word, not a string, no quotes) in the box for the value. While you could simply type these expressions into your IDE, we encourage you to answer them directly since this will help reinforce your understand of basic Python expressions.

Assume we've made the following assignments:

```
> str1 = 'hello'
> str2 = ','
> str3 = 'world'
```

Note: Advanced String Slicing

You've seen in lecture that you can slice a string with a call such as s[i:j], which gives you a portion of string s from index i to index j-1. However this is not the only way to slice a string! If you omit the starting index, Python will assume that you

wish to start your slice at index 0. If you omit the ending index, Python will assume you wish to end your slice at the end of the string. Check out this session with the Python shell:

```
>>> s = 'Python is Fun!'
>>> s[1:5]
'ytho'
>>> s[:5]
'Pytho'
>>> s[1:]
'ython is Fun!'
>>> s[:]
'Python is Fun!'
```

That last example is interesting! If you omit both the start and ending index, you get your original string!

There's one other cool thing you can do with string slicing. You can add a third parameter, k, like this: s[i:j:k]. This gives a slice of the string s from index i to index j-1, with step size k. Check out the following examples:

```
>>> s = 'Python is Fun!'
>>> s[1:12:2]
'yhni u'
>>> s[1:12:3]
'yoiF'
>>> s[::2]
'Pto sFn'
```

The last example is similar to the example s[:]. With s[::2], we're asking for the full string s (from index 0 through 13), with a step size of 2 - so we end up with every other character in s. Pretty cool!

Note: The Python 'in' operator

The operators in and not in test for collection membership (a 'collection' refers to a string, list, tuple or dictionary - don't worry, we will cover lists, tuples and dictionaries soon!). The expression

```
element in coll
```

evaluates to True if element is a member of the collection coll, and False otherwise.

The expression

```
element not in coll
```

evaluates to True if element is **not** a member of the collection coll, and False otherwise. Note this returns the negation of element in coll - that is, the expression element not in coll is equivalent to the expression not (element in coll). 1. str1 ✓ Answer: string string ✓ Answer: 'hello' hello 2. str1[0] ✓ Answer: string string ✓ Answer: 'h' h 3. str1[1] **✓ Answer:** string string ✓ Answer: 'e' е 4. str1[-1] ✓ Answer: string string ✓ Answer: 'o' 0 5. len(str1) int ✓ Answer: int ✓ Answer: 5 5

Enviar

You have used 1 of 9999 attempts

1 Answers are displayed within the problem

Exercise 2 part 2

0 points possible (ungraded)

ESTIMATED TIME TO COMPLETE: 7 minutes

Note that you will have to answer all questions before you can click the Check button.

For each of the expressions below, specify its type and value. If it generates an error, select type 'NoneType' and write the word 'error' (note this is a word, not a string, no quotes) in the box for the value. While you could simply type these expressions into your IDE, we encourage you to answer them directly since this will help reinforce your understand of basic Python expressions.

Assume we've made the following assignments:

```
> str1 = 'hello'
> str2 = ','
> str3 = 'world'
```

Note: Advanced String Slicing

You've seen in lecture that you can slice a string with a call such as s[i:j], which gives you a portion of string s from index i to index j-1. However this is not the only way to slice a string! If you omit the starting index, Python will assume that you wish to start your slice at index 0. If you omit the ending index, Python will assume you wish to end your slice at the end of the string. Check out this session with the Python shell:

```
>>> s = 'Python is Fun!'
>>> s[1:5]
'ytho'
>>> s[:5]
'Pytho'
>>> s[1:]
'ython is Fun!'
>>> s[:]
'Python is Fun!'
```

I nat last example is interesting! If you omit both the start and ending index, you get your original string!

There's one other cool thing you can do with string slicing. You can add a third parameter, k, like this: s[i:j:k]. This gives a slice of the string s from index i to index j-1, with step size k. Check out the following examples:

```
>>> s = 'Python is Fun!'
>>> s[1:12:2]
'yhni u'
>>> s[1:12:3]
'yoiF'
>>> s[::2]
'Pto sFn'
```

The last example is similar to the example s[:]. With s[::2], we're asking for the full string s (from index 0 through 13), with a step size of 2 - so we end up with every other character in s. Pretty cool!

Note: The Python 'in' operator

The operators in and not in test for collection membership (a 'collection' refers to a string, list, tuple or dictionary - don't worry, we will cover lists, tuples and dictionaries soon!). The expression

element in coll

evaluates to True if element is a member of the collection coll, and False otherwise.

The expression

element not in coll

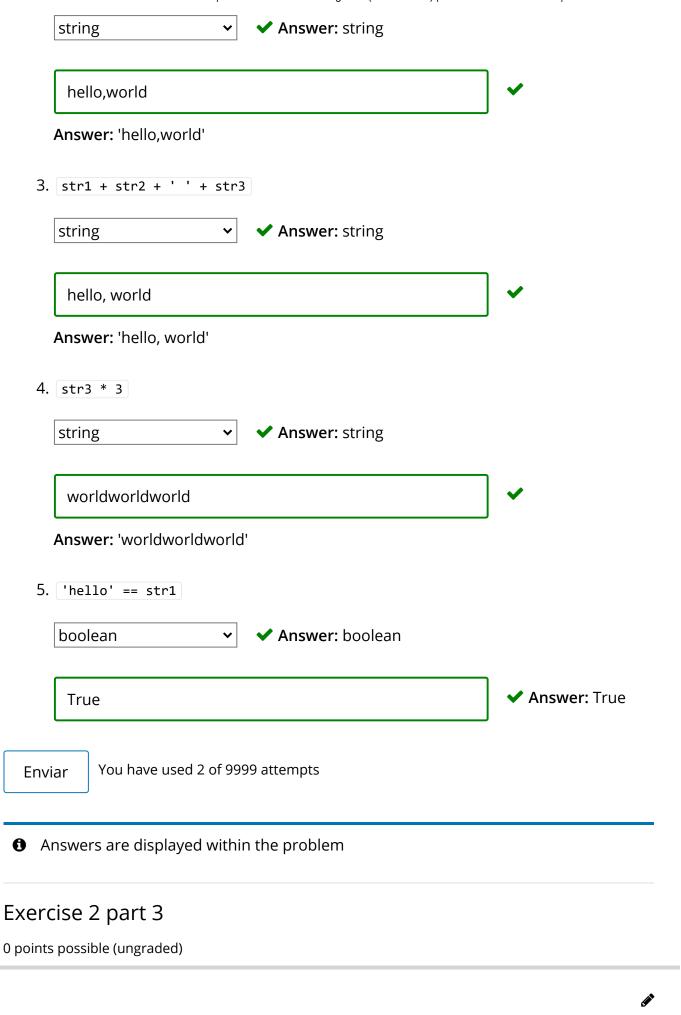
evaluates to True if element is **not** a member of the collection coll, and False otherwise.

Note this returns the negation of element in coll - that is, the expression element not in coll is equivalent to the expression not (element in coll).

1. str1[len(str1)]

NoneType	~	✓ Answer: NoneType	
error			✓ Answer: error

2. str1 + str2 + str3



ESTIMATED TIME TO COMPLETE: 7 minutes Note that you will have to answer all questions before you can click the Check button.

For each of the expressions below, specify its type and value. If it generates an error, select type 'NoneType' and write the word 'error' (note this is a word, not a string, no quotes) in the box for the value. While you could simply type these expressions into your IDE, we encourage you to answer them directly since this will help reinforce your understand of basic Python expressions.

Assume we've made the following assignments:

```
> str1 = 'hello'
> str2 = ','
> str3 = 'world'
```

Note: Advanced String Slicing

You've seen in lecture that you can slice a string with a call such as s[i:j], which gives you a portion of string s from index i to index j-1. However this is not the only way to slice a string! If you omit the starting index, Python will assume that you wish to start your slice at index 0. If you omit the ending index, Python will assume you wish to end your slice at the end of the string. Check out this session with the Python shell:

```
>>> s = 'Python is Fun!'
>>> s[1:5]
'ytho'
>>> s[:5]
'Pytho'
>>> s[1:]
'ython is Fun!'
>>> s[:]
'Python is Fun!'
```

That last example is interesting! If you omit both the start and ending index, you get your original string!

There's one other cool thing you can do with string slicing. You can add a third parameter, k, like this: s[i:j:k]. This gives a slice of the string s from index i to index j-1, with step size k. Check out the following examples:

```
>>> s = 'Python is Fun!'
>>> s[1:12:2]
'yhni u'
>>> s[1:12:3]
'yoiF'
>>> s[::2]
'Pto sFn'
```

The last example is similar to the example s[:]. With s[::2], we're asking for the full string s (from index 0 through 13), with a step size of 2 - so we end up with every other character in s. Pretty cool!

Note: The Python 'in' operator The operators in and not in test for collection membership (a 'collection' refers to a string, list, tuple or dictionary - don't worry, we will cover lists, tuples and dictionaries soon!). The expression element in coll evaluates to True if element is a member of the collection coll, and False otherwise. The expression element not in coll evaluates to True if element is **not** a member of the collection coll, and False otherwise. Note this returns the negation of element in coll - that is, the expression element not in coll is equivalent to the expression not (element in coll). 1. 'HELLO' == str1 ✓ Answer: boolean boolean Answer: False False 2. 'a' in str3

✓ Answer: boolean

boolean

Answer: False False 3. str4 = str1 + str3'low' in str4 boolean ✓ Answer: boolean ✓ Answer: True True 4. str3[1:3] ✓ Answer: string string ✓ Answer: 'or' or 5. str3[:3] Answer: string string Answer: 'wor' wor You have used 2 of 9999 attempts Enviar **1** Answers are displayed within the problem

Exercise 2 part 4

0 points possible (ungraded)

ESTIMATED TIME TO COMPLETE: 7 minutes

Note that you will have to answer all questions before you can click the Check button.

For each of the expressions below specify its type and value. If it generates an error select type 'NoneType' and write the word 'error' (note this is a word, not a string, no quotes) in the box for the value. While you could simply type these expressions into yo IDE, we encourage you to answer them directly since this will help reinforce your understand of basic Python expressions.

Assume we've made the following assignments:

```
> str1 = 'hello'
> str2 = ','
> str3 = 'world'
> str4 = str1 + str3
```

Note: Advanced String Slicing

You've seen in lecture that you can slice a string with a call such as <code>s[i:j]</code> , which gives you a portion of string s from index i to index j-1. However this is not the only way to slice a string! If you omit the starting index, Python will assume that you wish to start your slice at index 0. If you omit the ending index, Python will assume you wish to end your slice at the end of the string. Check out this session with the Python shell:

```
>>> s = 'Python is Fun!'
>>> s[1:5]
'ytho'
>>> s[:5]
'Pytho'
>>> s[1:]
'ython is Fun!'
>>> s[:]
'Python is Fun!'
```

That last example is interesting! If you omit both the start and ending index, you get your original string!

There's one other cool thing you can do with string slicing. You can add a third parameter, k, like this: s[i:j:k]. This gives a slice of the string s from index i to index j-1, with step size k. Check out the following examples:

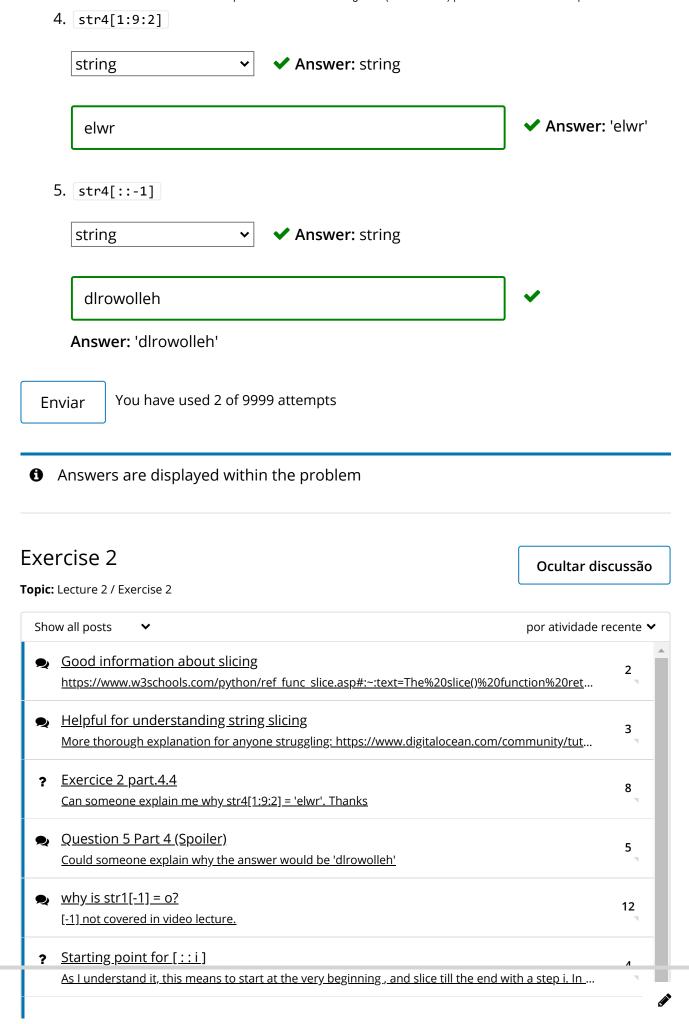
```
>>> s = 'Python is Fun!'
>>> s[1:12:2]
'yhni u'
>>> s[1:12:3]
'yoiF'
>>> s[::2]
'Pto sFn'
```

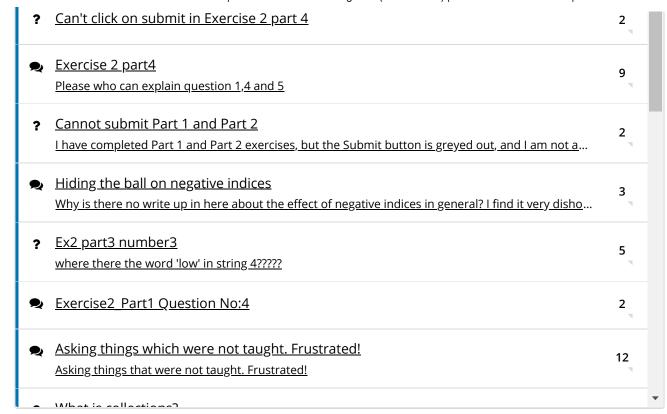
The last example is similar to the example s[:]. With s[::2], we're asking for the full string s (from index 0 through 13), with a sten size of 2 - so we end up with every other character in s. Pretty cool!

Note: The Python 'in' operator The operators in and not in test for collection membership (a 'collection' refers to a string, list, tuple or dictionary - don't worry, we will cover lists, tuples and dictionaries soon!). The expression element in coll evaluates to True if element is a member of the collection coll, and False otherwise. The expression element not in coll evaluates to True if element is **not** a member of the collection coll, and False otherwise. Note this returns the negation of element in coll - that is, the expression element not in coll is equivalent to the expression not (element in coll). 1. str3[:-1] string ✓ Answer: string Answer: 'worl' worl 2. str1[1:] **✓ Answer:** string string ✓ Answer: 'ello' ello 3. str4[1:9] string **✓ Answer:** string elloworl

https://courses.edx.org/courses/courses/course-v1:MITx+6.00.1x+2T2020/courseware/fc8f42302c644118adfcfa720f9f403e/35f82f6c3ecb4e9e913dc279... 11/13

Answer: 'elloworl'





© All Rights Reserved