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`

   string ✔ **Answer:** string

   hello ✔ **Answer:** 'hello'

2. `str1[0]`

   string ✔ **Answer:** string

   h ✔ **Answer:** 'h'

3. `str1[1]`

   string ✔ **Answer:** string

   e ✔ **Answer:** 'e'

4. `str1[-1]`

   string ✔ **Answer:** string

   o ✔ **Answer:** 'o'

5. `len(str1)`

   int ✔ **Answer:** int

   5 ✔ **Answer:** 5

Enviar You have used 1 of 9999 attempts

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!'
```

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[len(str1)]`

   NoneType ✔ **Answer:** NoneType

   error ✔ **Answer:** error

2. `str1 + str2 + str3`

string ✔ **Answer:** string

hello,world ✔

Answer: 'hello,world'

3. `str1 + str2 + ' ' + str3`

string ✔ **Answer:** string

hello, world ✔

Answer: 'hello, world'

4. `str3 * 3`

string ✔ **Answer:** string

worldworldworld ✔

Answer: 'worldworldworld'

5. `'hello' == str1`

boolean ✔ **Answer:** boolean

True ✔ **Answer:** True

Enviar You have used 2 of 9999 attempts

Answers are displayed within the problem

Exercise 2 part 3

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!'
```

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`

   boolean ✔ **Answer:** boolean

   False ✔ **Answer:** False

2. `'a' in str3`

   boolean ✔ **Answer:** boolean

False

✔ **Answer:** False

3.

```
str4 = str1 + str3
'low' in str4
```

boolean

✔ **Answer:** boolean

True

✔ **Answer:** True

4. `str3[1:3]`

string

✔ **Answer:** string

or

✔ **Answer:** 'or'

5. `str3[:3]`

string

✔ **Answer:** string

wor

✔ **Answer:** 'wor'

Enviar

You have used 2 of 9999 attempts

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 `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. `str3[:-1]`

   string ✔ **Answer:** string

   worl ✔ **Answer:** 'worl'

2. `str1[1:]`

   string ✔ **Answer:** string

   ello ✔ **Answer:** 'ello'

3. `str4[1:9]`

   string ✔ **Answer:** string

   elloworl ✔

   Answer: 'elloworl'

4. `str4[1:9:2]`

string ✔ **Answer:** string

elwr ✔ **Answer:** 'elwr'

5. `str4[::-1]`

string ✔ **Answer:** string

dlrowolleh ✔

Answer: 'dlrowolleh'

Enviar You have used 2 of 9999 attempts

Answers are displayed within the problem

Exercise 2

Ocultar discussão

Topic: Lecture 2 / Exercise 2

Show all posts por atividade recente

Good information about slicing
https://www.w3schools.com/python/ref_func_slice.asp#:~:text=The%20slice()%20function%20ret... 2

Helpful for understanding string slicing
More thorough explanation for anyone struggling: https://www.digitalocean.com/community/tut... 3

Exercice 2 part.4.4
Can someone explain me why str4[1:9:2] = 'elwr'. Thanks 8

Question 5 Part 4 (Spoiler)
Could someone explain why the answer would be 'dlrowolleh' 5

why is str1[-1] = o?
[-1] not covered in video lecture. 12

Starting point for [: : i]
As I understand it, this means to start at the very beginning , and slice till the end with a step i. In ... 4

? Can't click on submit in Exercise 2 part 4 2

Exercise 2 part4 9
Please who can explain question 1,4 and 5

? Cannot submit Part 1 and Part 2 2
I have completed Part 1 and Part 2 exercises, but the Submit button is greyed out, and I am not a...

Hiding the ball on negative indices 3
Why is there no write up in here about the effect of negative indices in general? I find it very disho...

? Ex2 part3 number3 5
where there the word 'low' in string 4?????

Exercise2_Part1 Question No:4 2

Asking things which were not taught. Frustrated! 12
Asking things that were not taught. Frustrated!

What is collections?