Python Strings

Defining Strings in Python

- three ways:

```
>>> x = ‘Mozzarella”      # single quotes
>>>
>>> x = “Mozzarella”     # double quotes
>>>
>>> x = “””Mozzrella”””   # triple quotes
```

- use escape (\) for special characters

```
>>> print(“\a”)              # system bell
```

Character Encoding

- ASCII: each character is one byte
- (extended) ASCII: up to (256) 128
- not enough for many alphabets
- Unicode: up to 4 bytes per character
- three encoding schemes:

1. UTF−32: use 4 bytes
2. UTF−16: use 2 bytes
3. UTF−8: variable length encoding

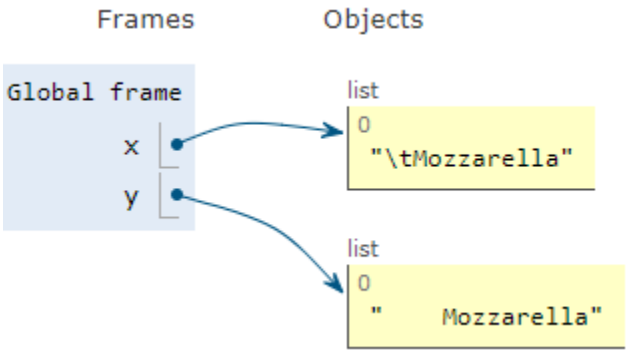
Character Escape Codes

character	description
\	newline continue
\\	backlash
\a	system bell
\b	backspace
\e	escape
\t	horizontal tab
\v	vertical tab
\”	double quotes

- raw strings: ignore escape codes

  >>> x = r‘\tMozzarella’

  >>> y = ‘\tMozzarella’



Escape Sequences

- Use "\" (backlash) for special characters:

```
>>> x = 'Mozzarella\'s"   # single quotes
Mozzarella's
>>> x = "Mozza\trella"    # tab
Mozza    rella
>>> x = "Mo\"zz\"rella"   # double quotes
Mo"zz"arella
>>> x = "D:\\bu\\python"
D:\bu\python
```

- for dir/file names, use this:

```
>>> import os
>>> x = os.path.join("D:", "bu", "python")
D:\bu\python
```

String Operations

- concatenation

```
>>>  “Mozzarella”   + “Cheddar”
‘MozzarellaCheddar’
```

- repetition

```
>>> “Mozzarella” * 3
‘MozzarellaMozzarellaMozzarella’
```

- indexing

```
>>> “Mozzarella”[0]
‘M’
```

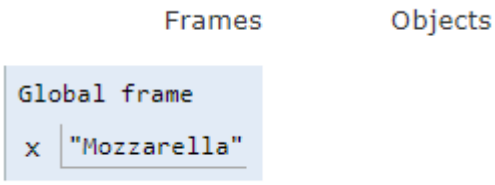
- slicng

```
>>> “Mozzarella”[0 : 4]
‘Mozz’
```

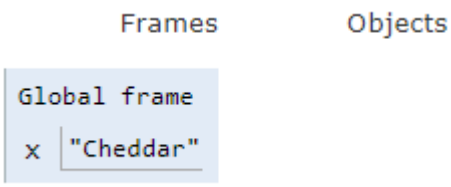
String Immutability

- once created, cannot be changed

```
>>> x = “Mozzarella”
>>> id(x)
139685060402176
```




```
>>> x =”Cheddar”
>>> id(x)
139760688281904
```



String as an Object

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x = “Mozarella”
>>> y = x.index(‘r’)
>>> y
>>> 5
```

- string is an object
- what are string attributes?

variable name	delimiter	attribute	arguments
x	.	index	(‘r’)

String as an Object

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x = “Mozzarella”       # double quotes
>>> x = ‘Mozzarella’       # single quotes
>>> x = “””Mozzarella”””  # triple quotes
>>> y = x.index(‘r’)
>>> y
>>> 5
```

- string is an object
- what are string attributes?

variable name	delimiter	attribute	arguments
x	.	index	(‘r’)

Slicing

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

>>> x = “Mozzarella”

- slice operator:

variable		start		one past end	
x	[	i	:	j	]

Indexing for String Slicing

- each element is identified by its position
- slice – elements between two positions

>>> x = “Mozzarella”

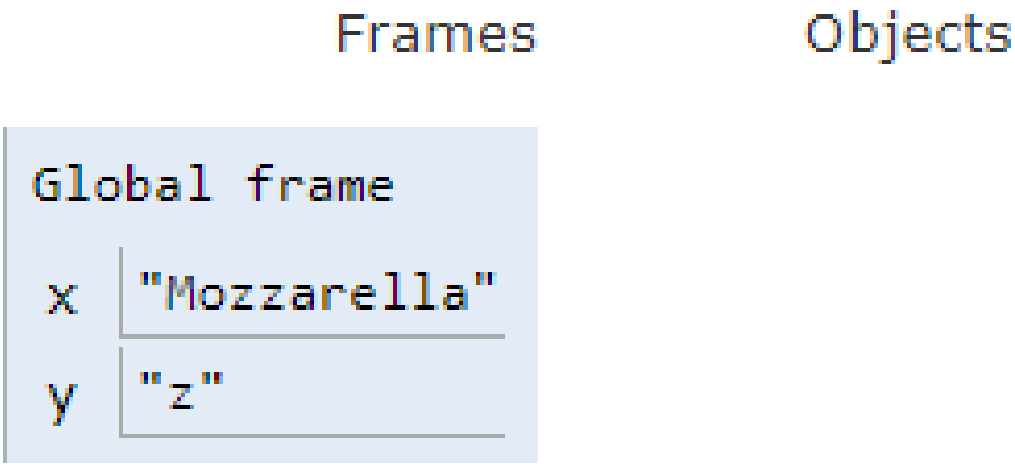
0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

- indexing starts at 0, ends at (length – 1)
- can refer to slices but positive or negative indices

Slicing with a Single Index

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

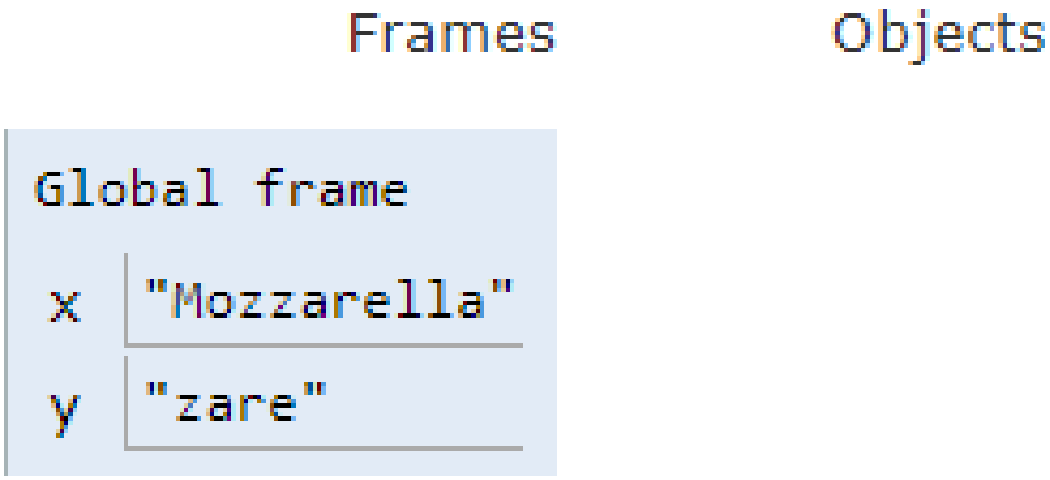
```
>>> x = “Mozzarella”
>>> y = x[2]
>>> y
>>> z
```



Slicing with Positive Indices

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

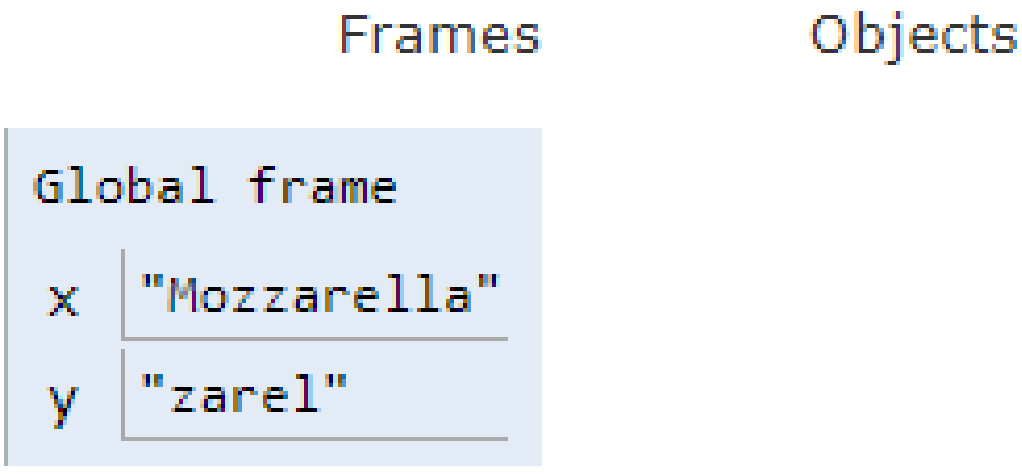
```
>>> x = “Mozzarella”
>>> y = x[3 : 7]
>>> y
>>> zare
```



Slicing w. Negative Indices

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

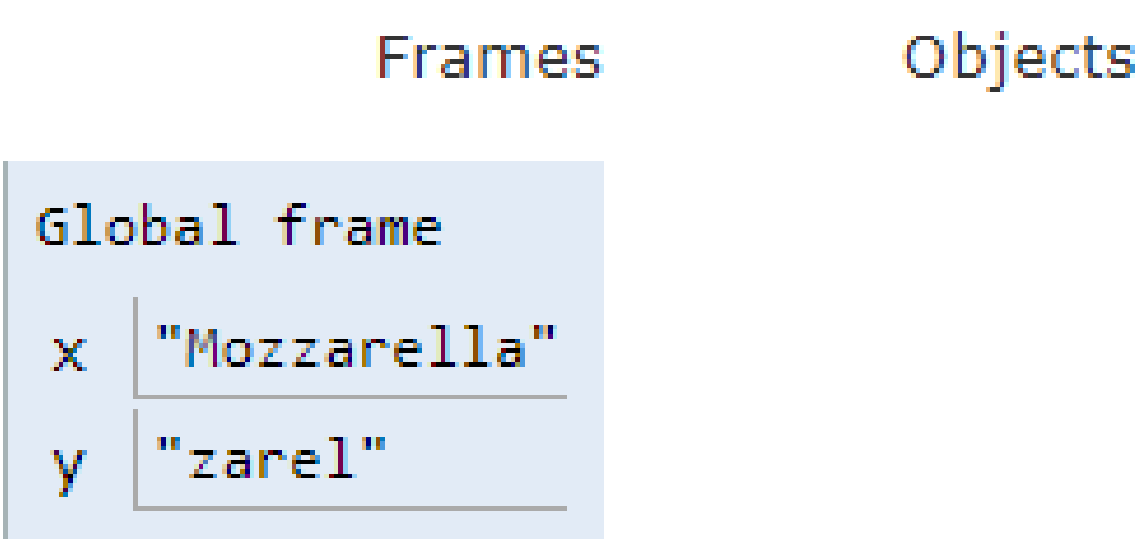
```
>>> x = “Mozzarella”
>>> y = x[−7 : −2]
>>> y
>>> zarel
```



Slicing with Positive and Negative Indices

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x = “Mozzarella”
>>> y = x[3 : -2]
>>> y
>>> zarel
```

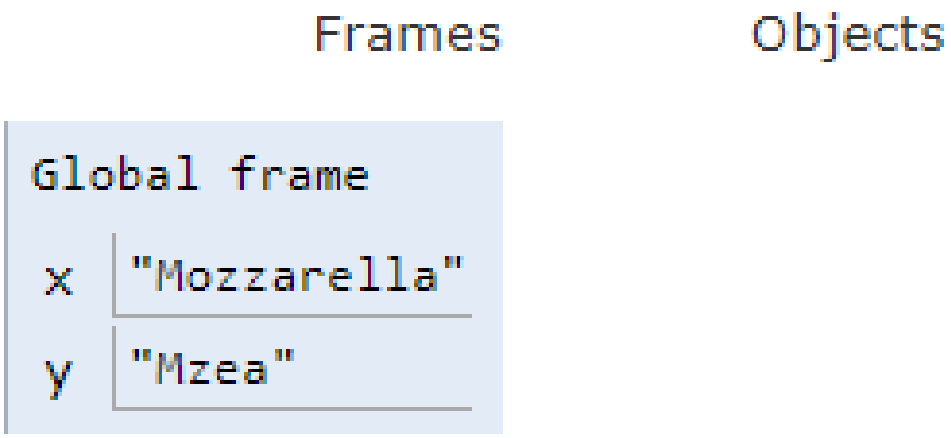


Extended Slicing

- [start: finish: count]

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

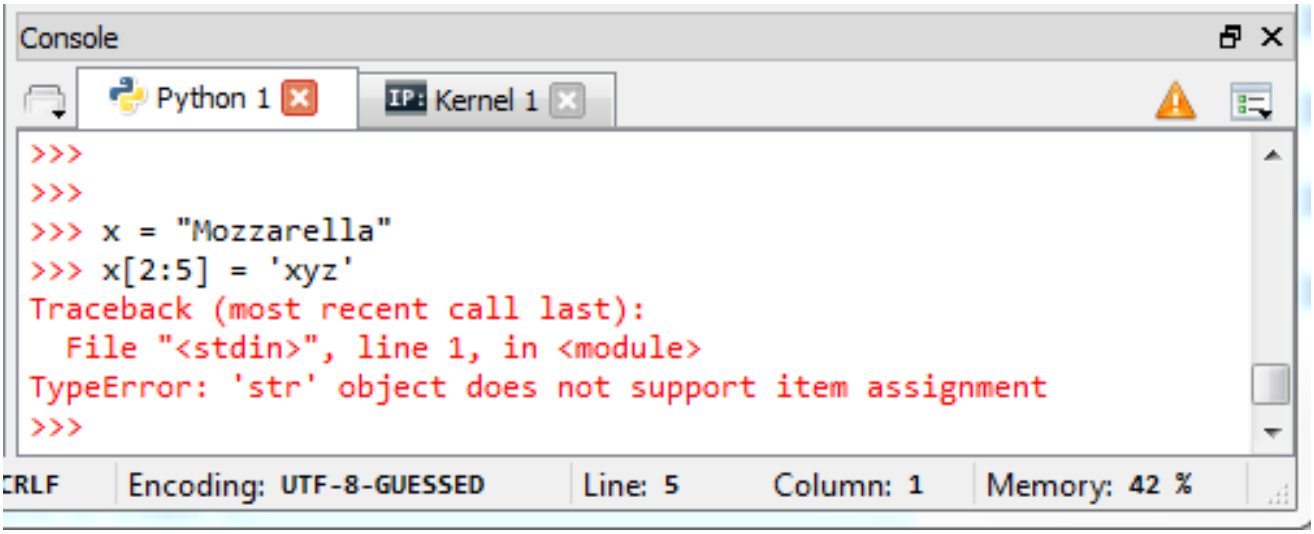
```
>>> # extract every third letter
>>> x = “Mozzarella”
>>> y = x[0 : 10 : 3]
>>> y
>>> Mzea
```



Slice Replacing

- strings are immutable
- cannot change substrings without creating new strings!!!

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

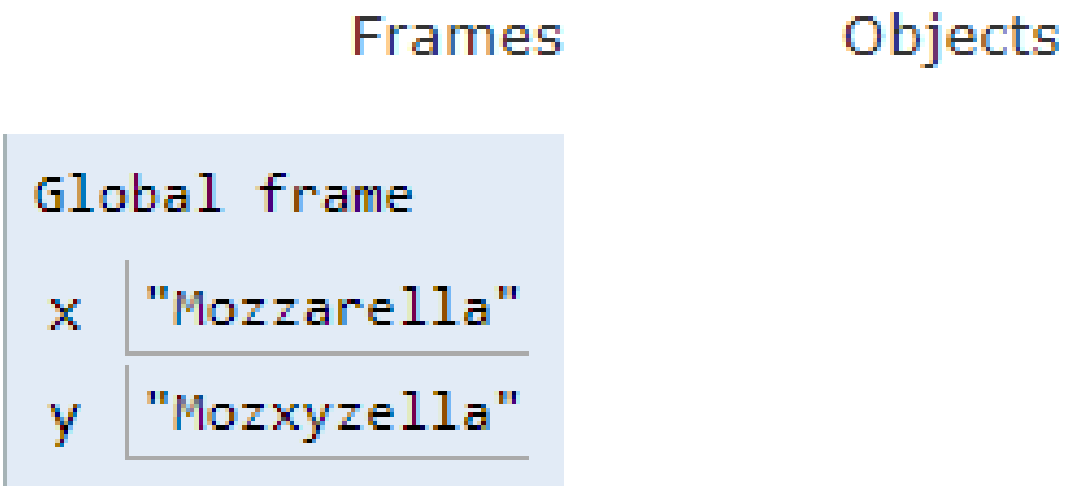


Substring Replacement

- can create a new string

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

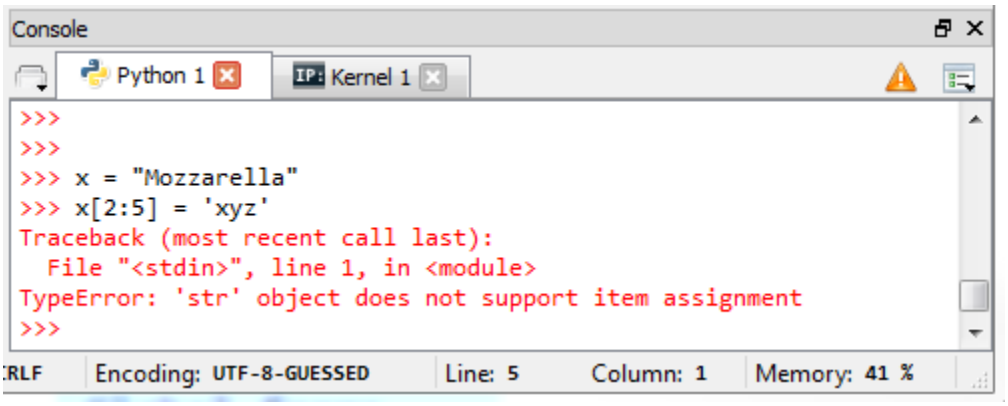
```
>>> x = “Mozzarella”
>>> y = x.replace(“zar”, “xyz”)
>>> y
>>> Mozxyzella
```



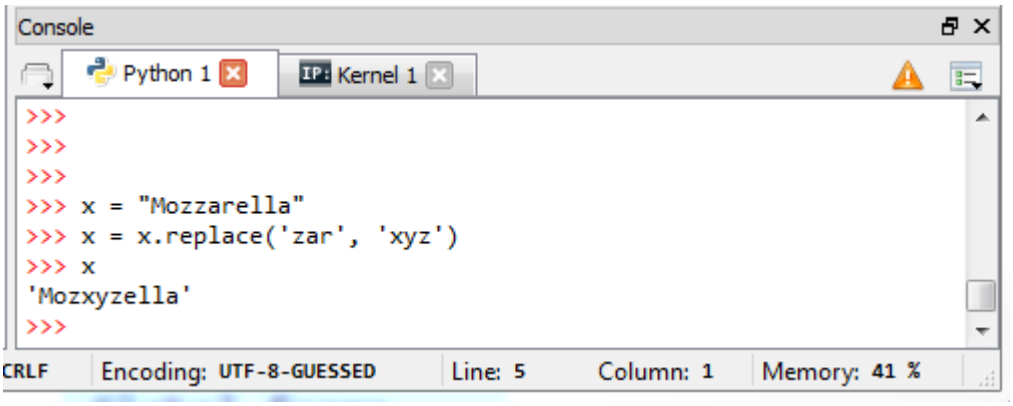
Valid vs. Invalid Slicing

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

- why is this illegal?



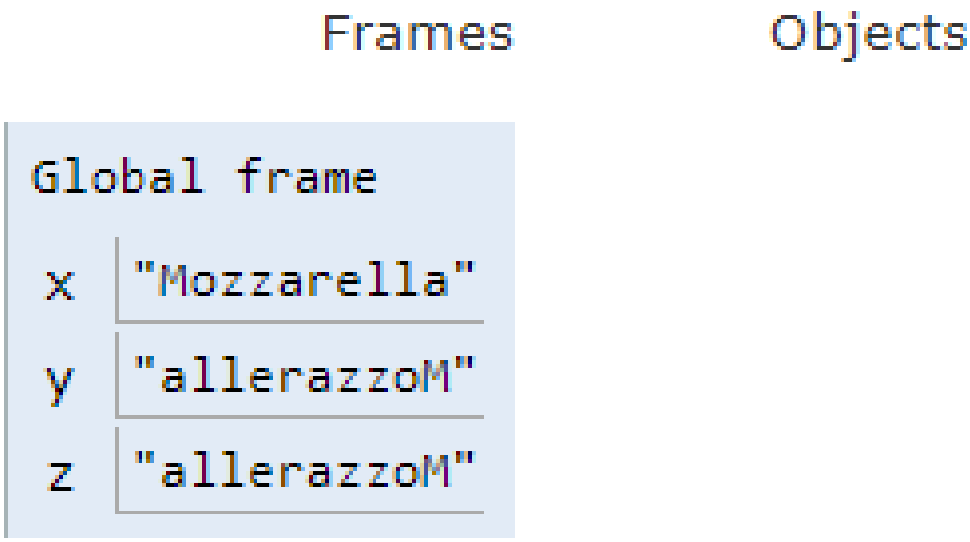
- why is this legal?



String Reversal

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x = “Mozzarella”
>>> y = x[: : −1]                    # idiomatic
>>> z = “”.join(reversed(x))    # slower
```



Slicing: Quiz

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

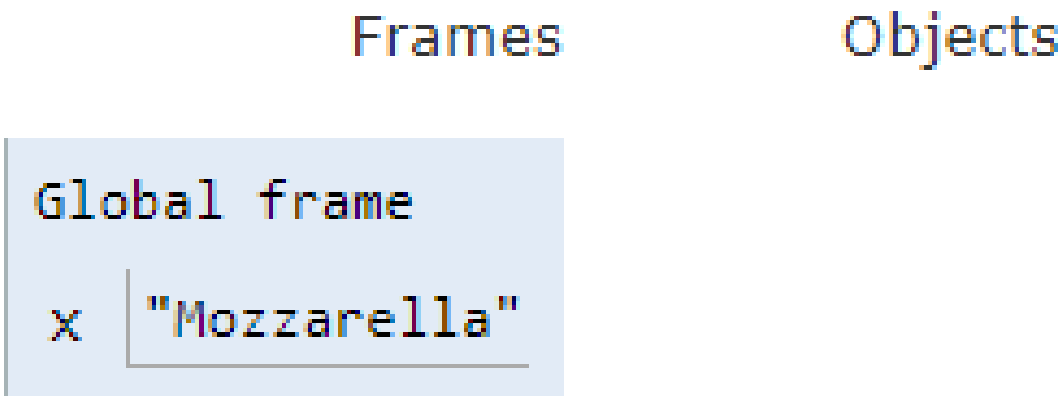
- what is the result of these operations?

```
>>> x =  “Mozzarella”
>>> y = x[−1]
>>> z = x[2 : 6]
>>> w = x[−7 : −3]
>>> v = x[ : ]
```

Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x =  "Mozzarella"
```

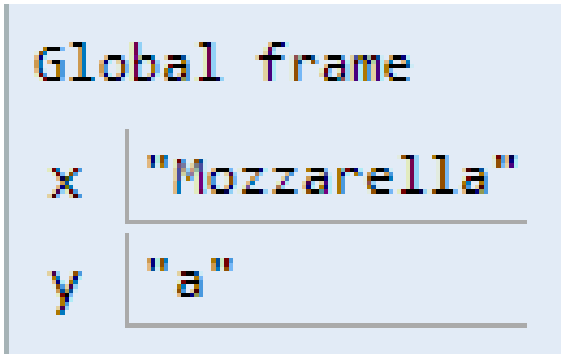


Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x =  "Mozzarella"
>>> y = x[−1]
```

Frames Objects

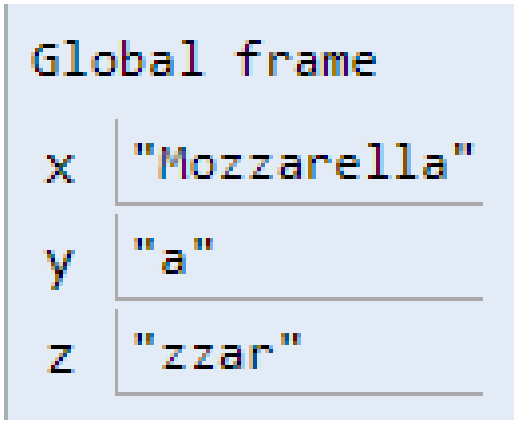


Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x =  "Mozzarella"
>>> y = x[−1]
>>> z = x[2 : 6]
```

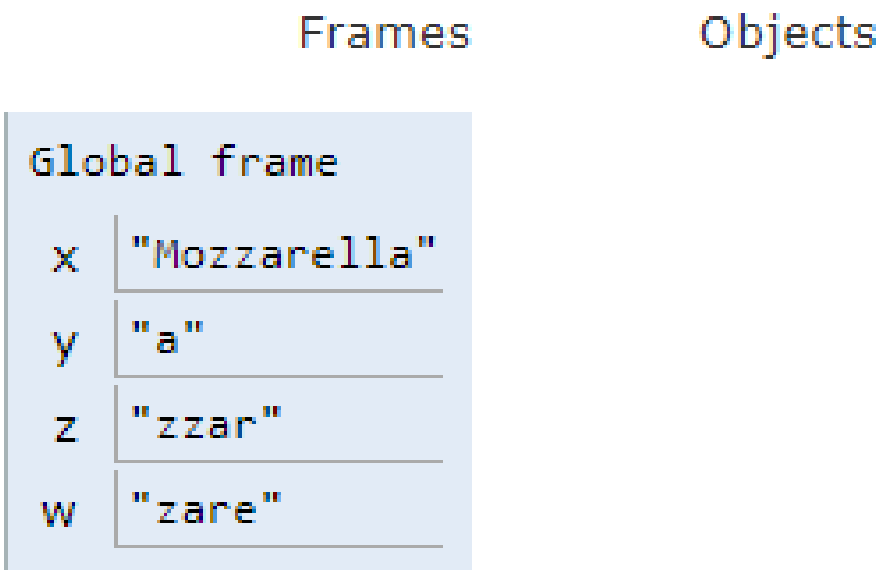
Frames Objects



Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x =  "Mozzarella"
>>> y = x[−1]
>>> z = x[2 : 6]
>>> w = x[−7 : −3]
```

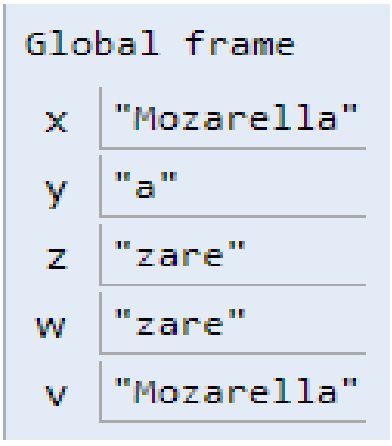


Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x =  "Mozzarella"
>>> y = x[ −1]
>>> z = x[2 : 6]
>>> w = x[ −7 : −3]
>>> v = x[ : ]
```

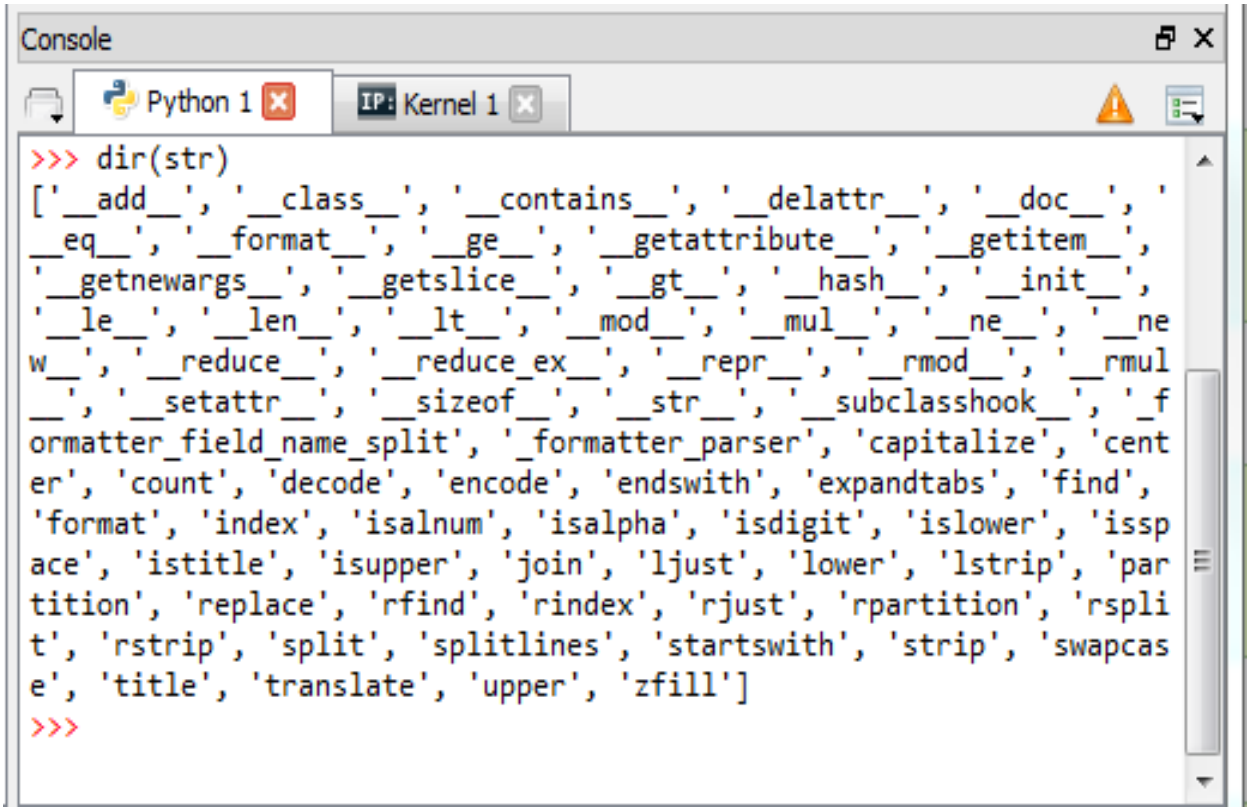
Frames Objects



- example of a shallow copy

String Attributes

- use dir(str) command



- names starting with __ refer to “local” class names/methods

String Comparisons

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

```
>>> x = “Mozzarella”
>>> y = x.upper()
>>> condition_1 = (x[4] == y[9])
>>> condition_2 = (x[4] == y[9].lower())
>>> condition_3 = (x[0 : 2] <= y[0 : 2])
>>> condition_4 = (x[0 : 2] > y[0 : 2])
```

Frames Objects

Global frame

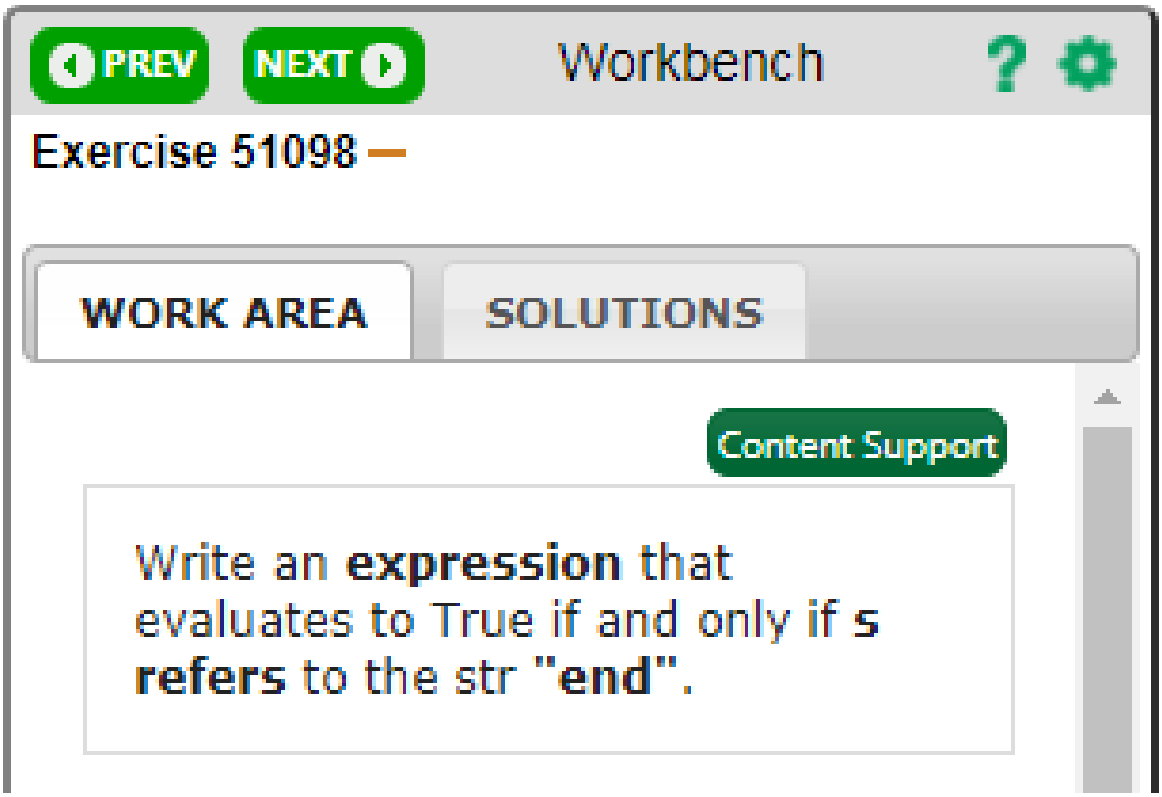
x	"Mozzarella"
y	"MOZZARELLA"
condition_1	False
condition_2	True
condition_3	False
condition_4	True

Review Problems

Interview Problem

- what is negative index?

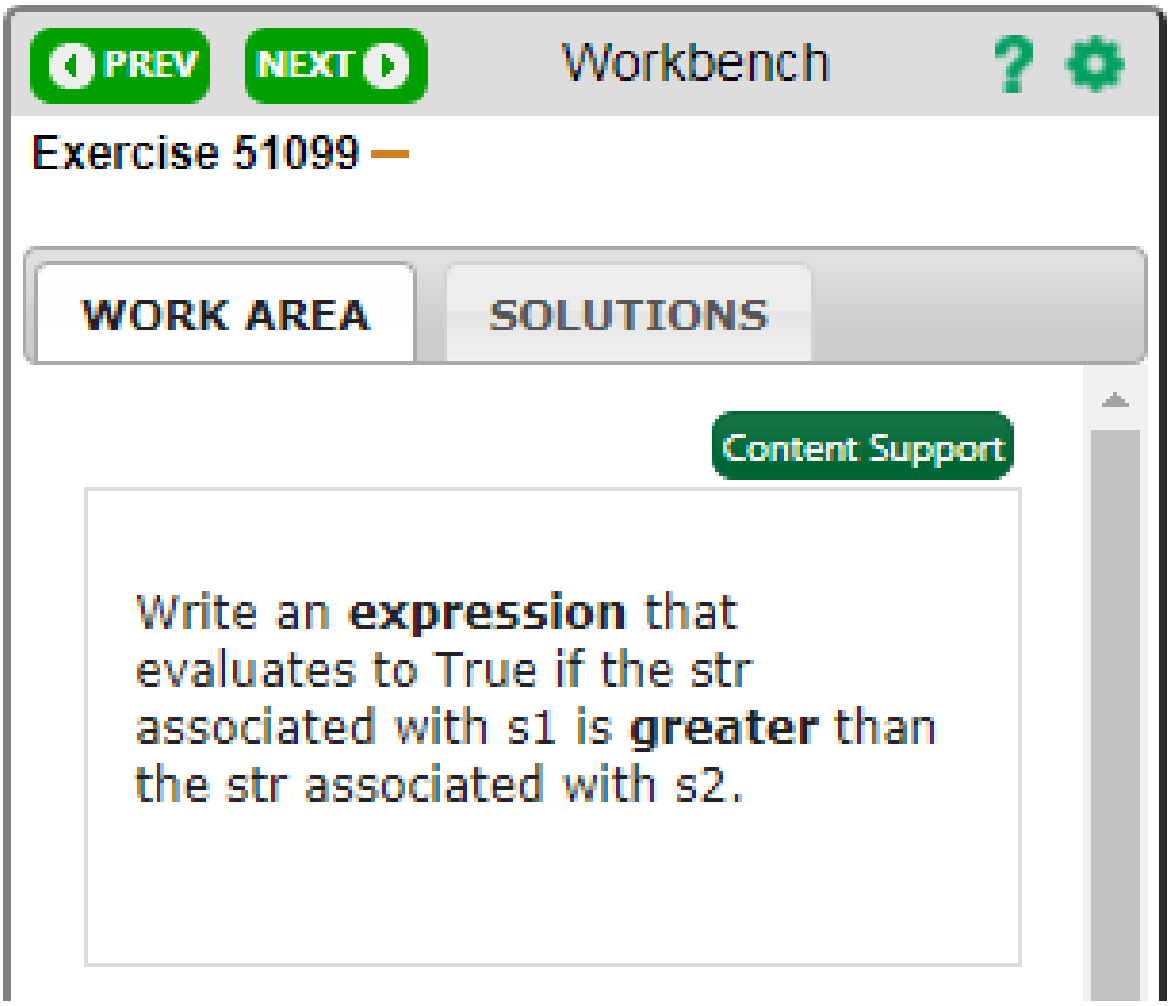
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

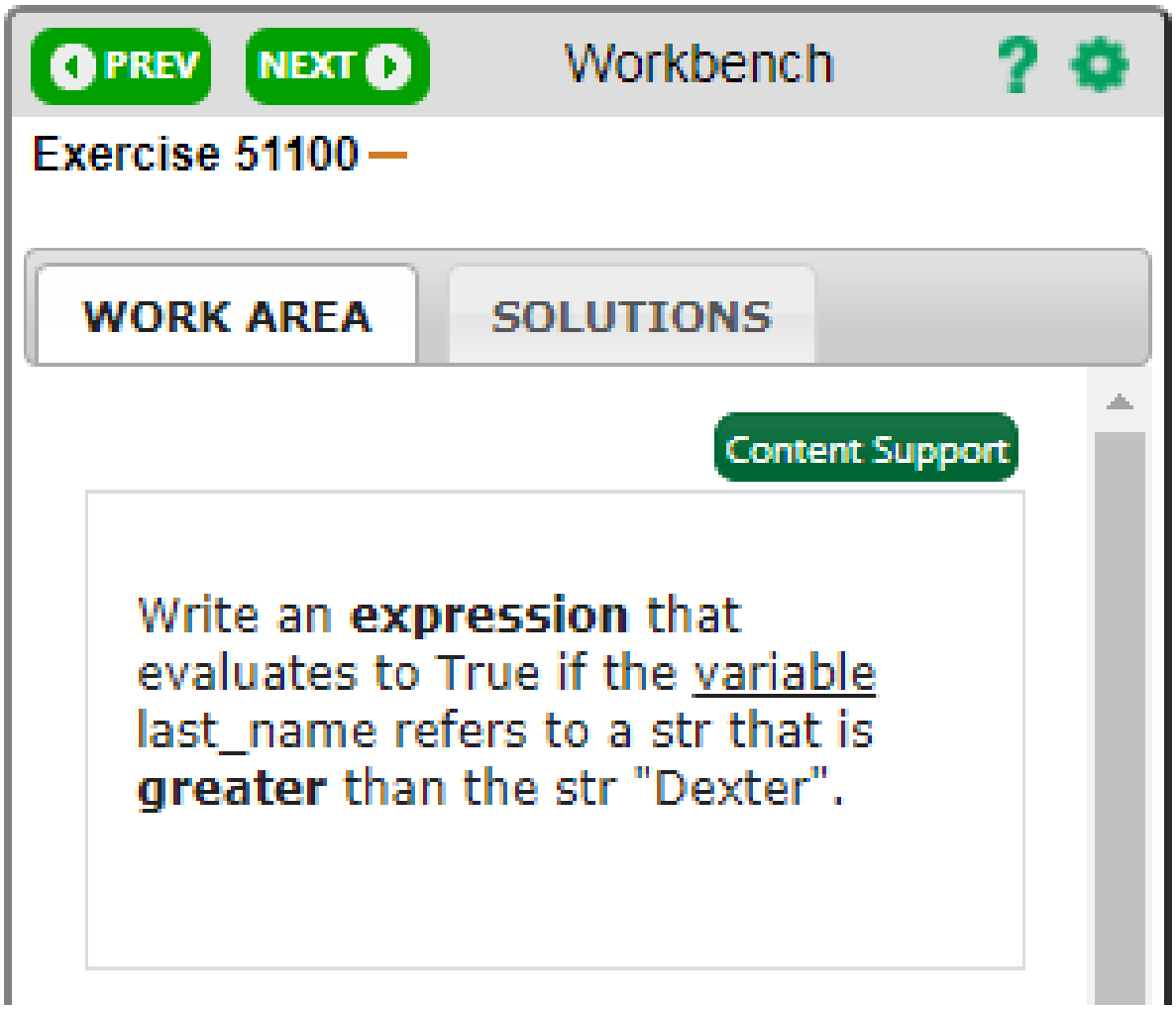
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

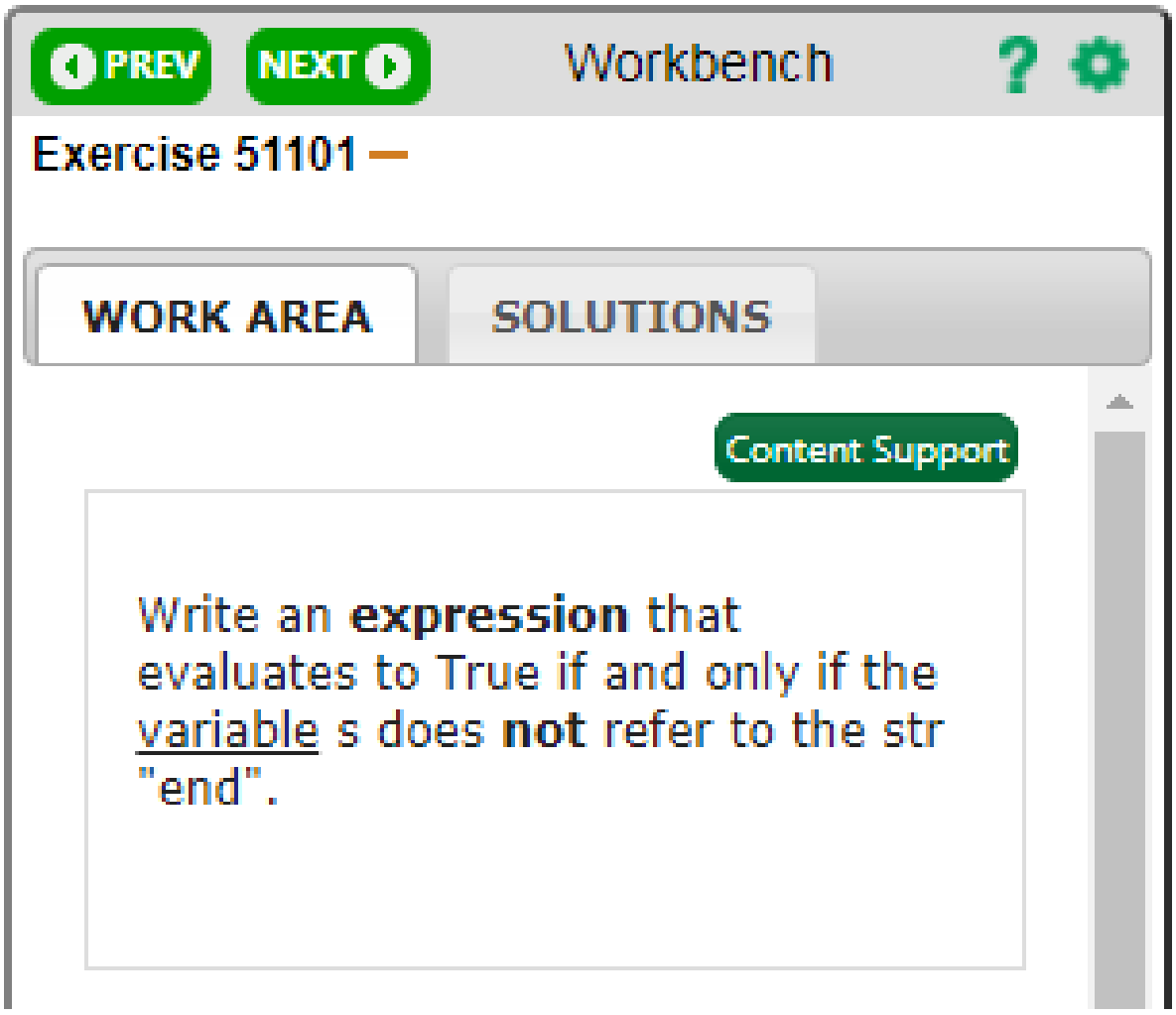
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

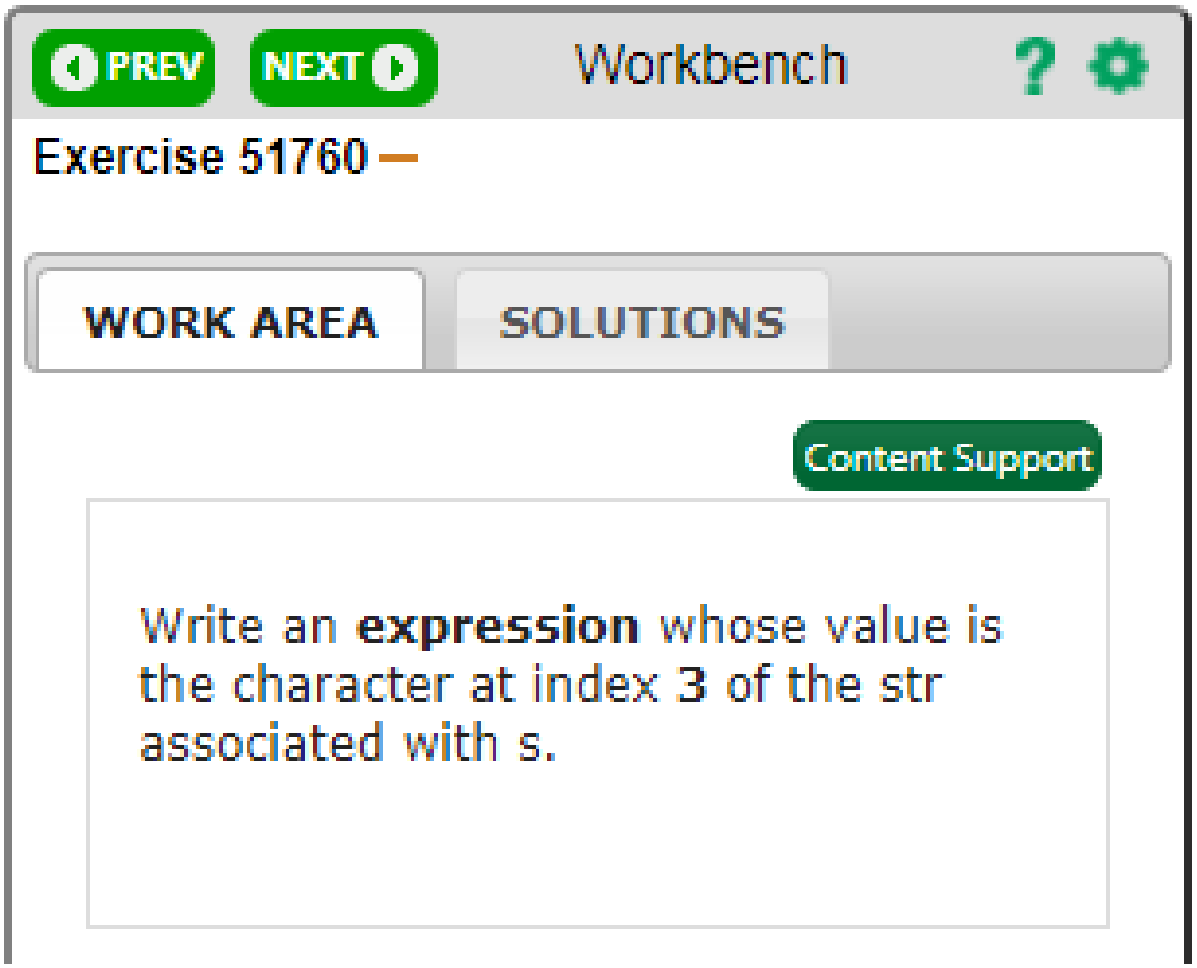
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

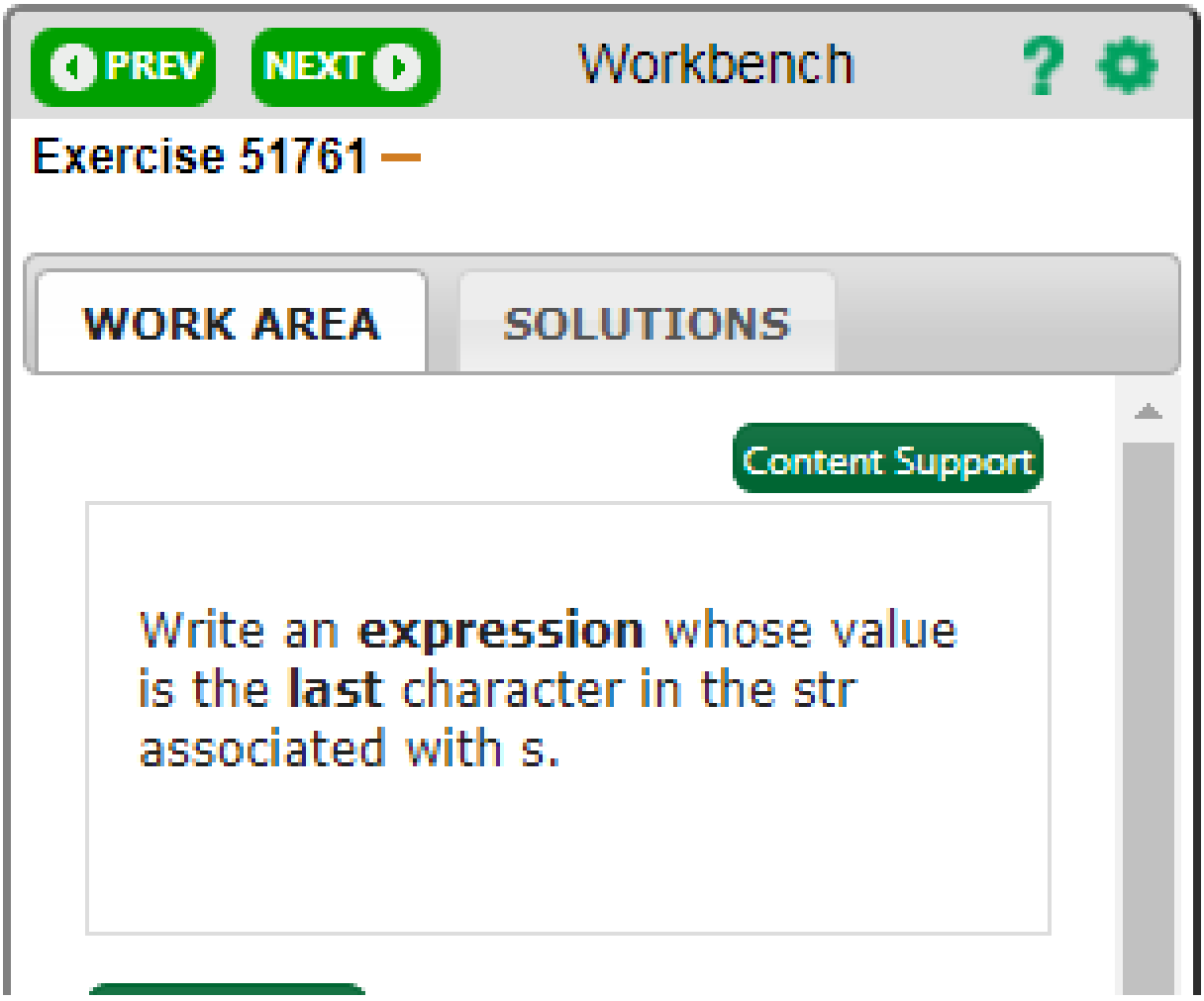
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

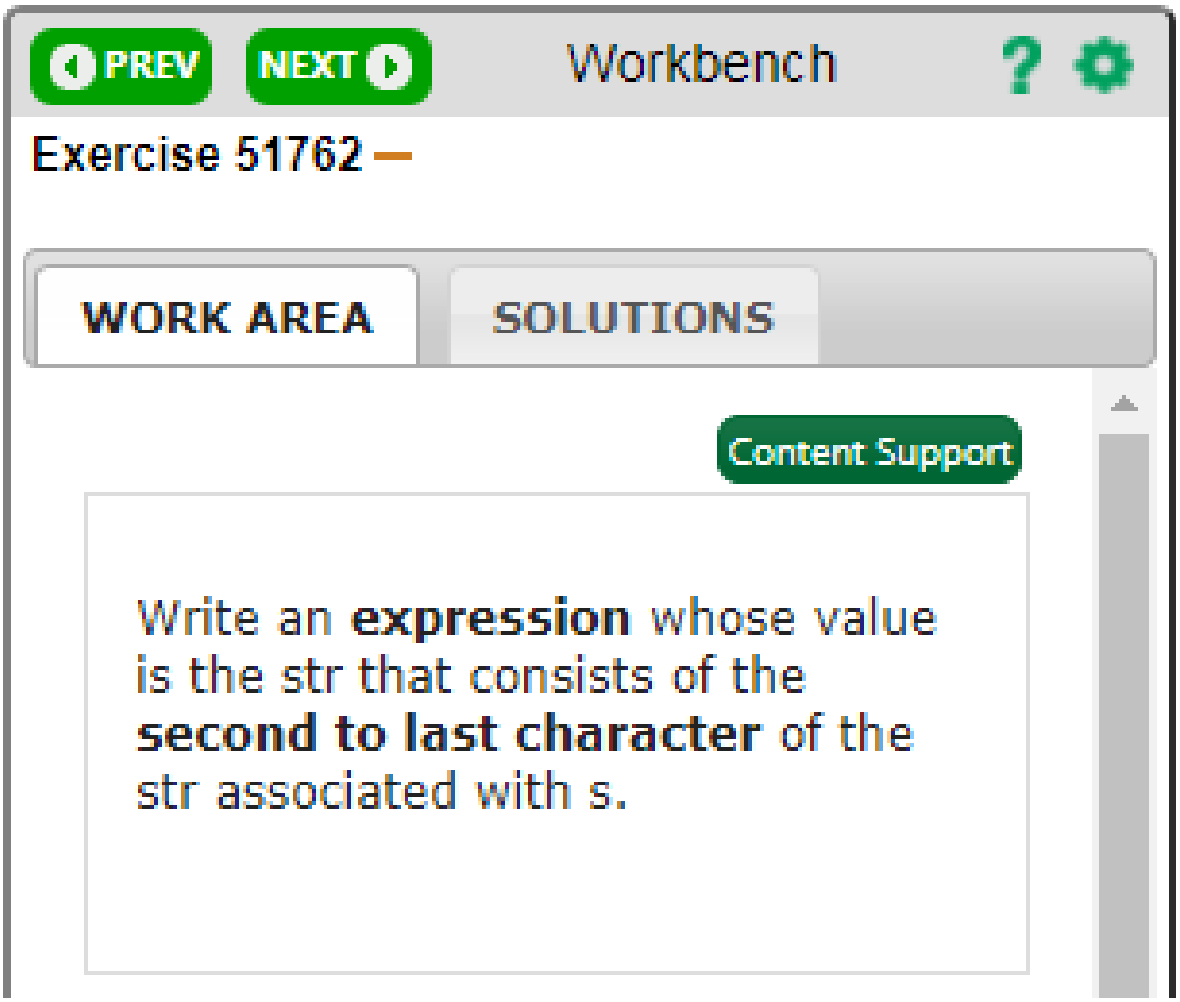
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

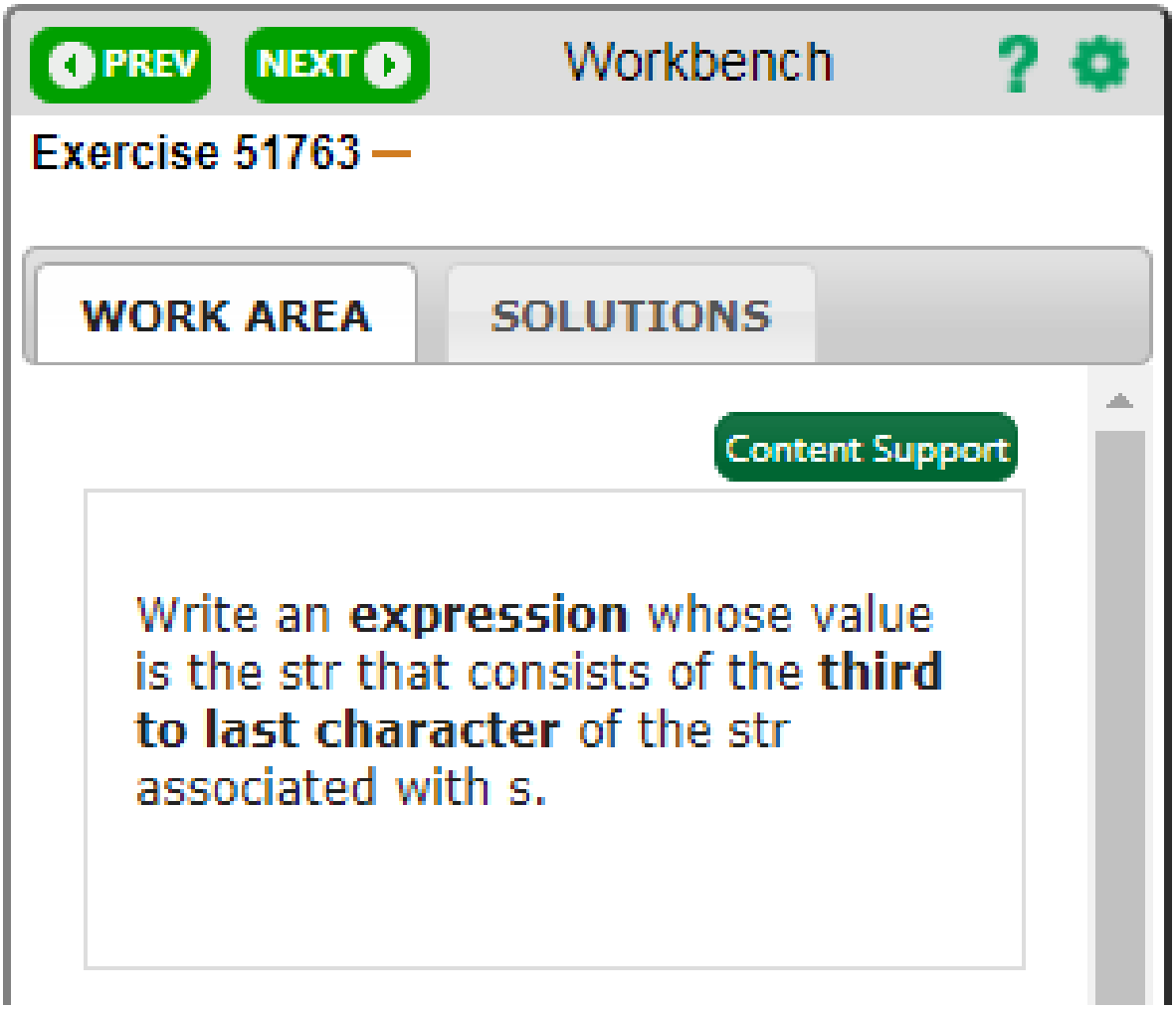
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

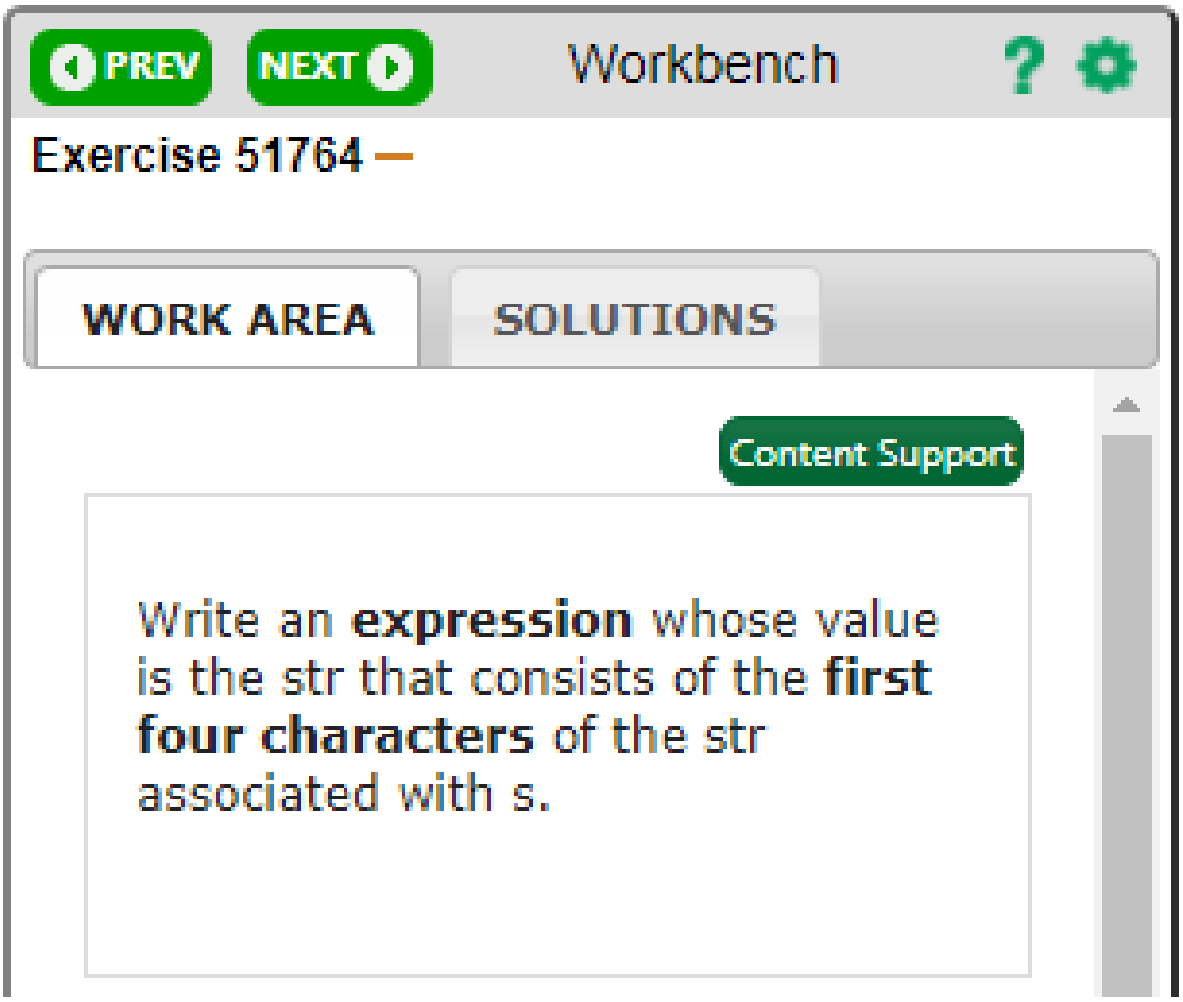
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

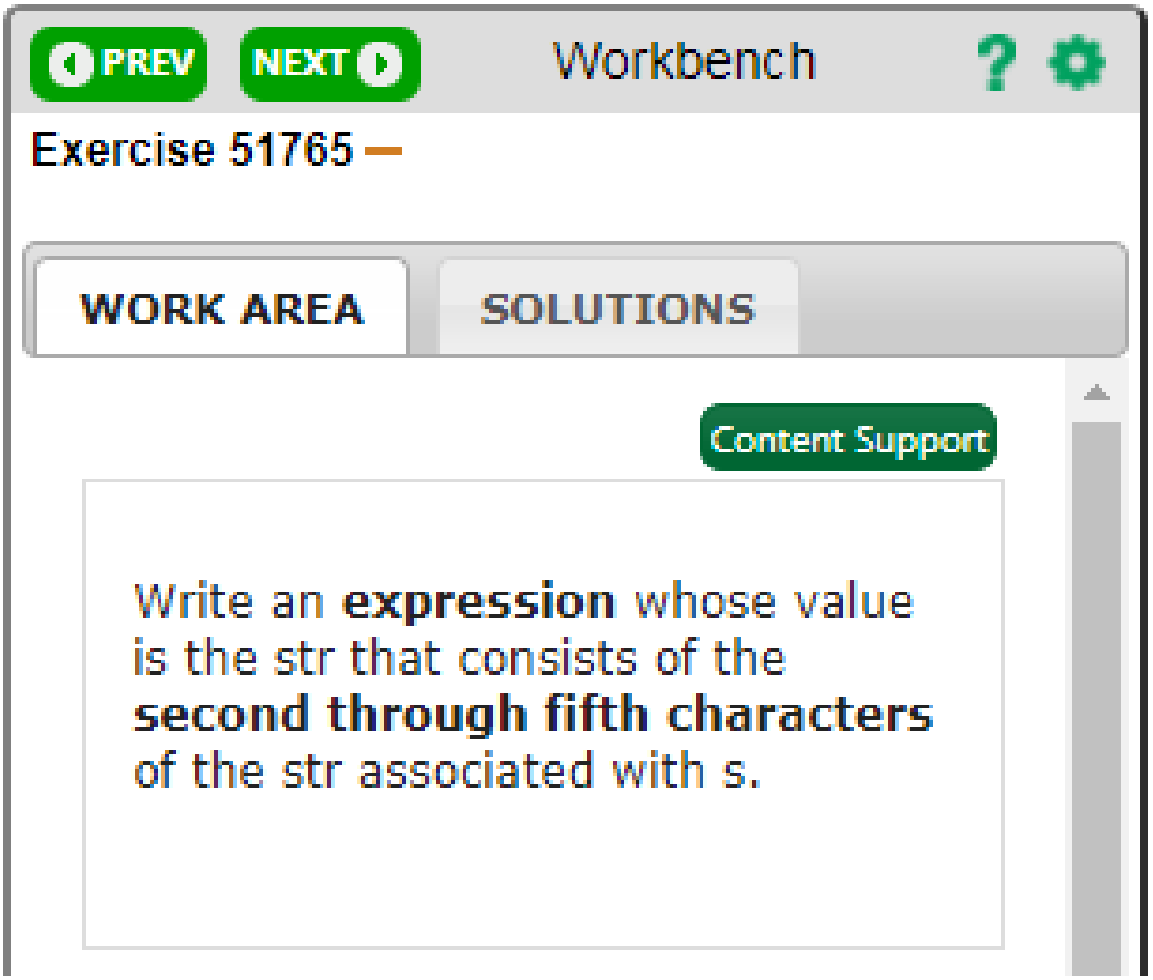
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

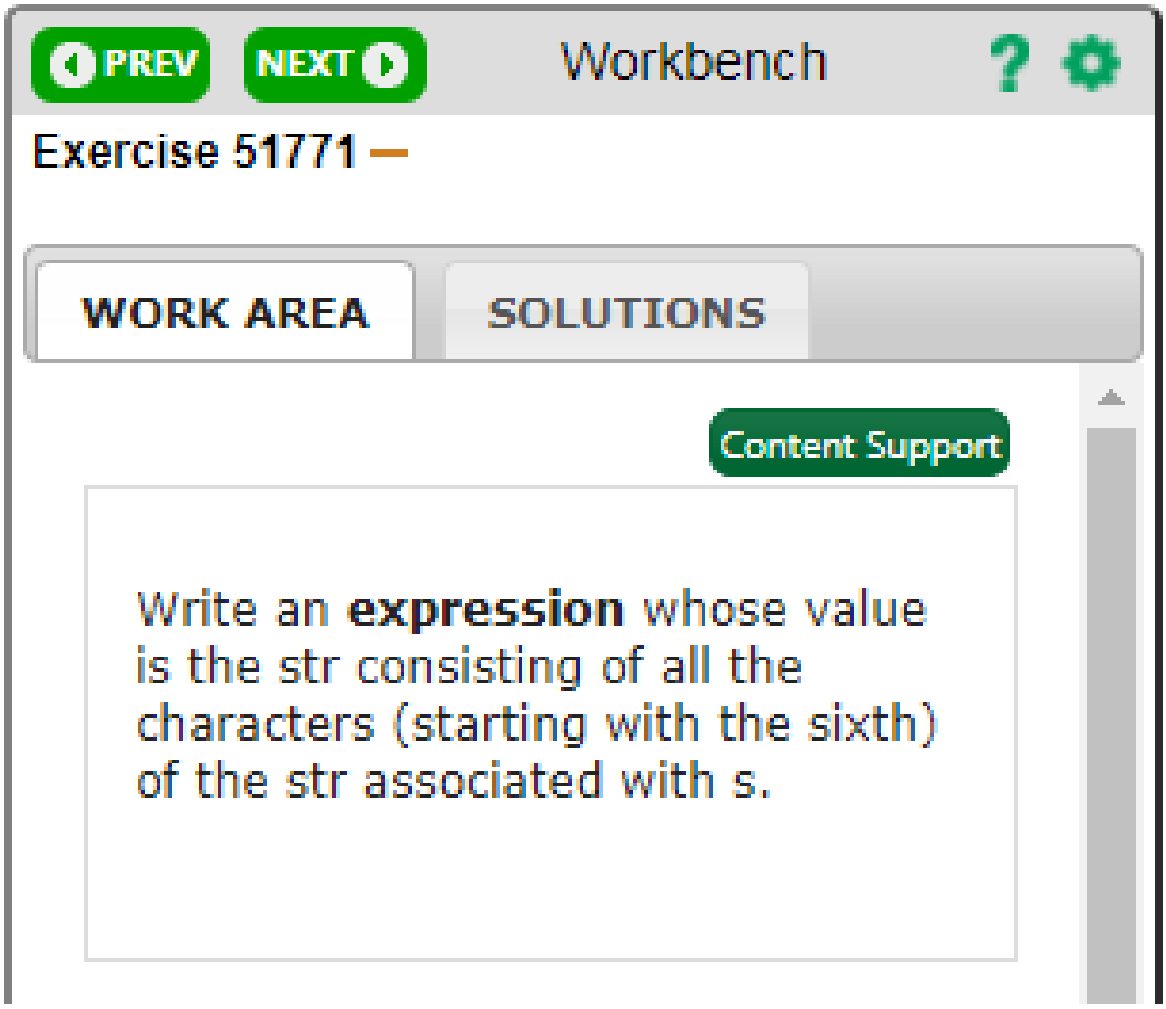
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

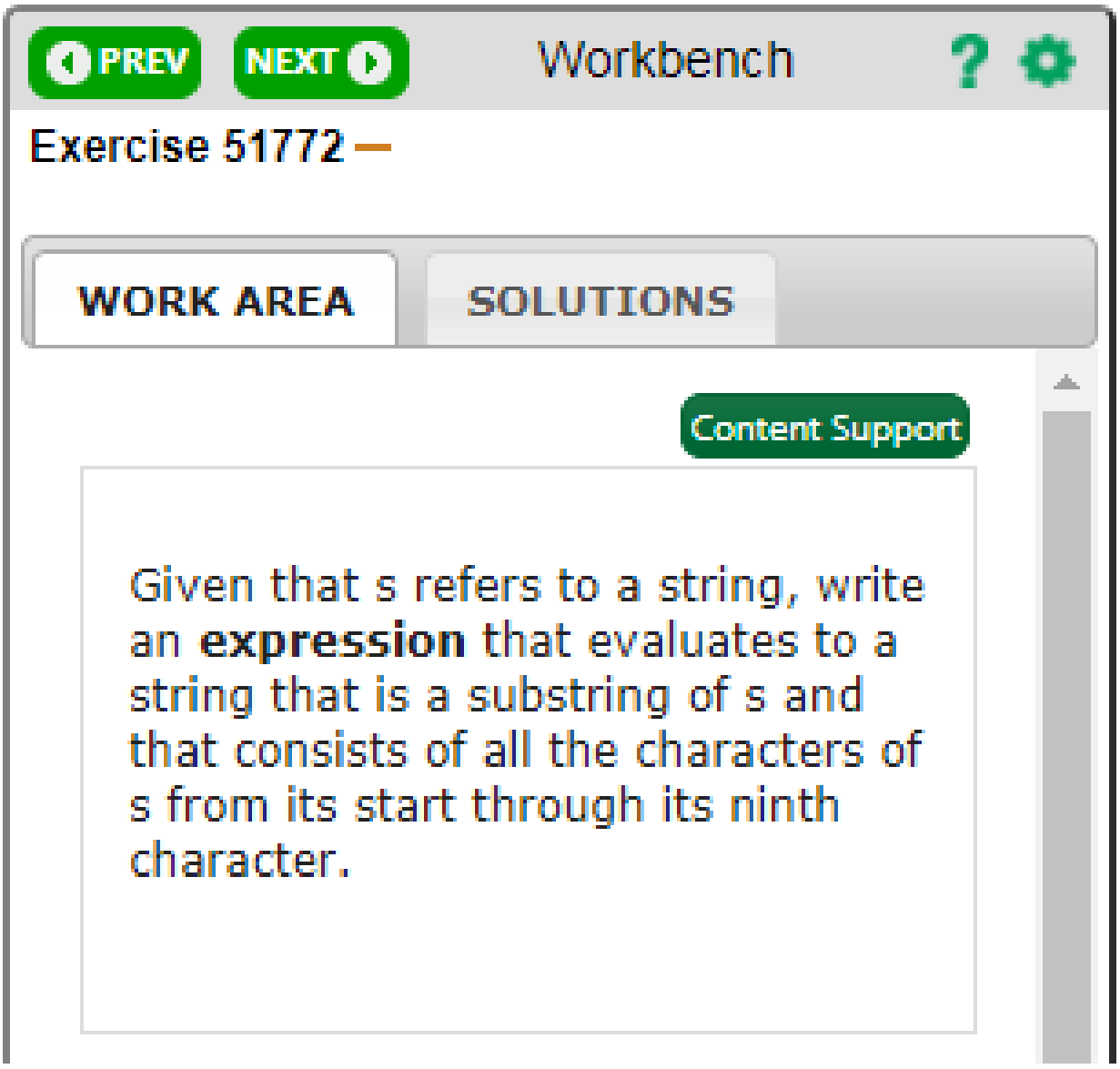
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

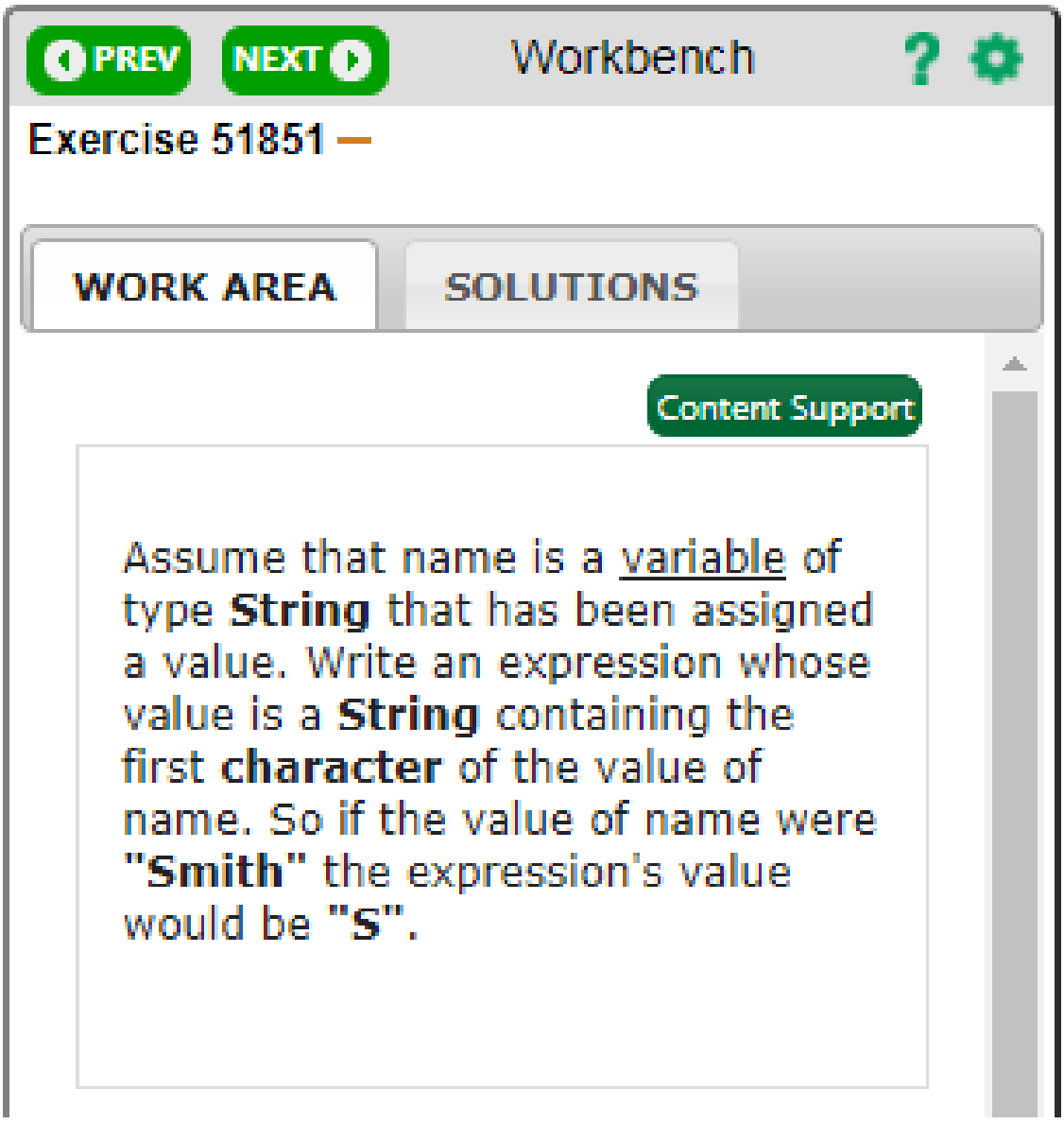
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank