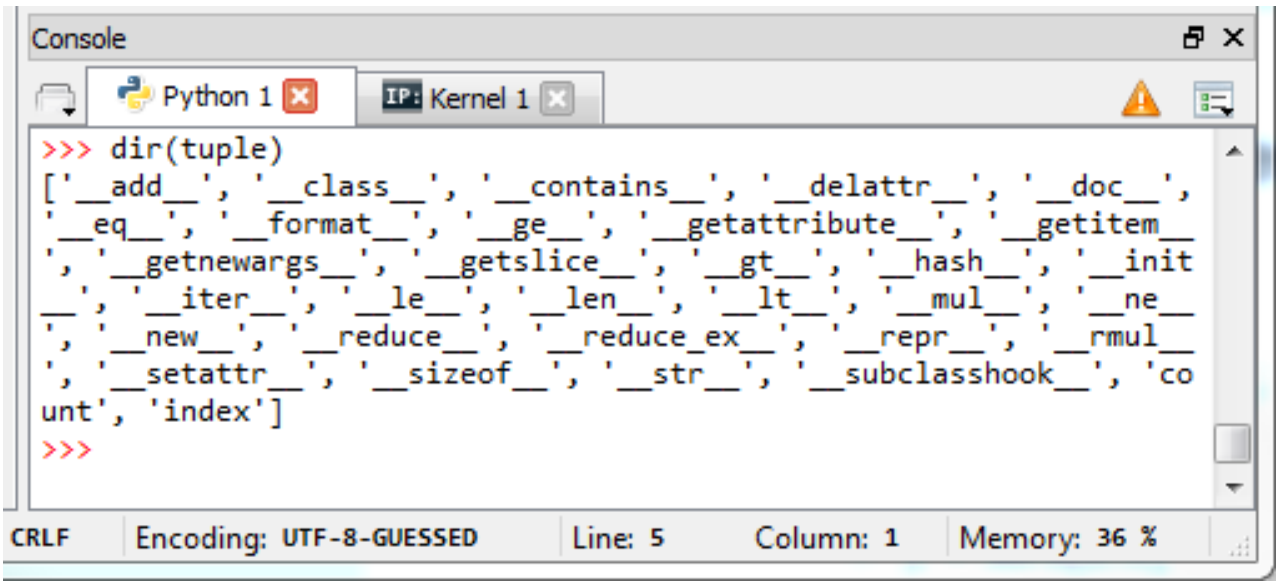
Python Tuples

Tuple Attributes

- use dir(tuple) command



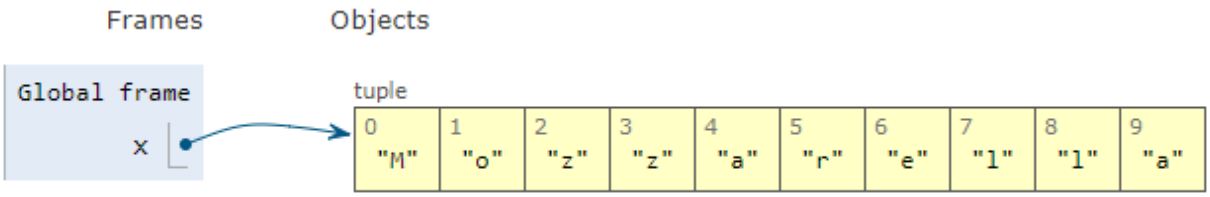
- names starting with __ refer to “local” class names/methods

Python Tuples

- ordered sequence of elements
- immutable (faster iterations)
- can contain immutable elements
- tuple data is write-protected

	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	

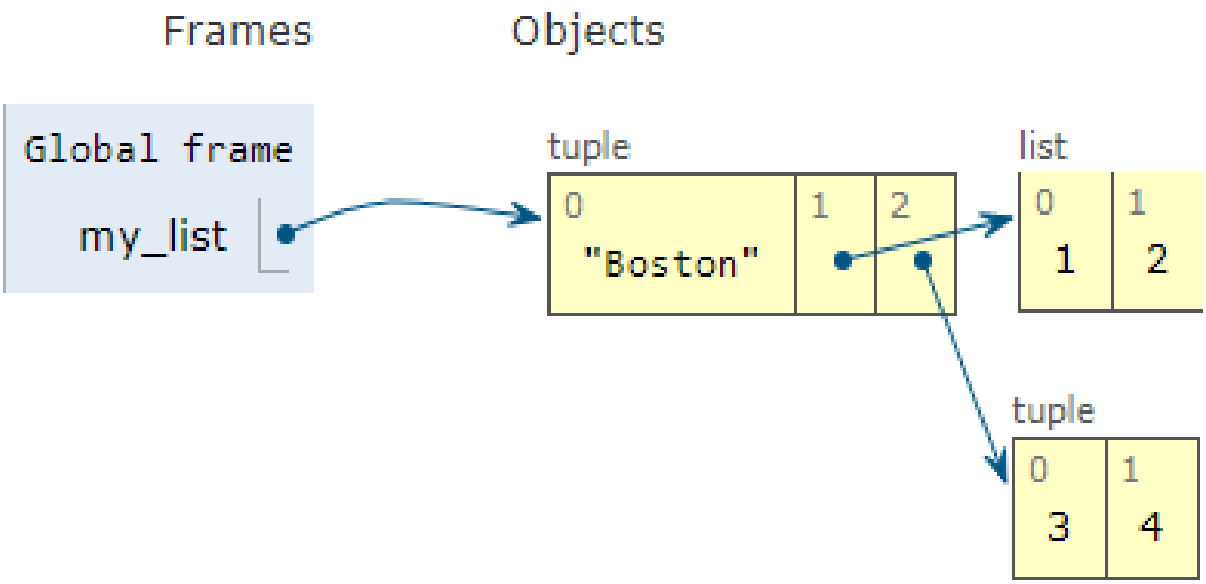
```
>>> x = (‘M’,‘o’,‘z’,‘z’,‘a’,‘r’,‘e’,‘l’,‘l’,a’)
```



Tuple with Mixed Types

- can contain mixed data types
- by contrasts, lists typically contain homogeneous data

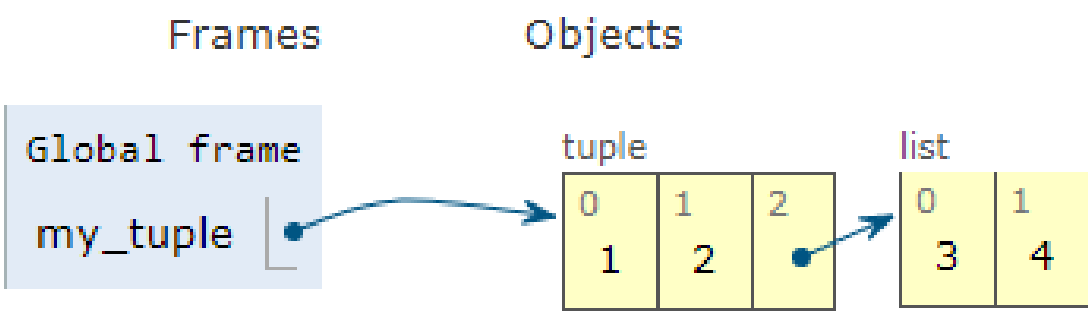
```
>>> my_tuple = (“Boston”, [1, 2], (4, 5))
```



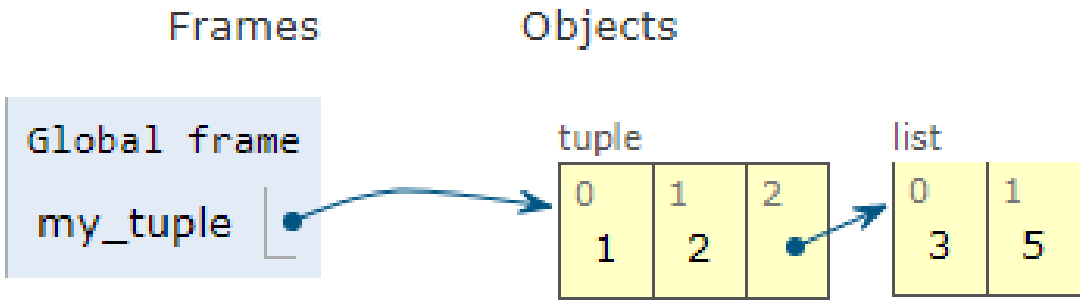
Tuple Immutability

- cannot be changed (immutable)
- nested mutable elements can be changed

```
>>> my_tuple = (1, 2, [3, 4])
```




```
>>> my_tuple[2][1] = 5
```



Tuple Packing/Unpacking

- packing: create tuple from a sequence

```
>>> choices = “Mozzarrella”, “Brie”
>>> choices
(“Mozzarrella”, “Brie”)
```

- unpacking: distribute tuple elements across variables

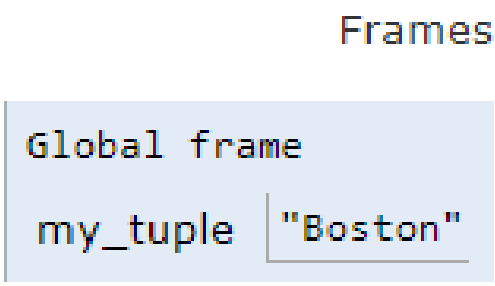
```
>>> choices = “Mozzarrella”, “Brie”
>>> choice_1, choice_2 = choices
>>> choice_2
“Brie”
```

- similar unpacking for lists

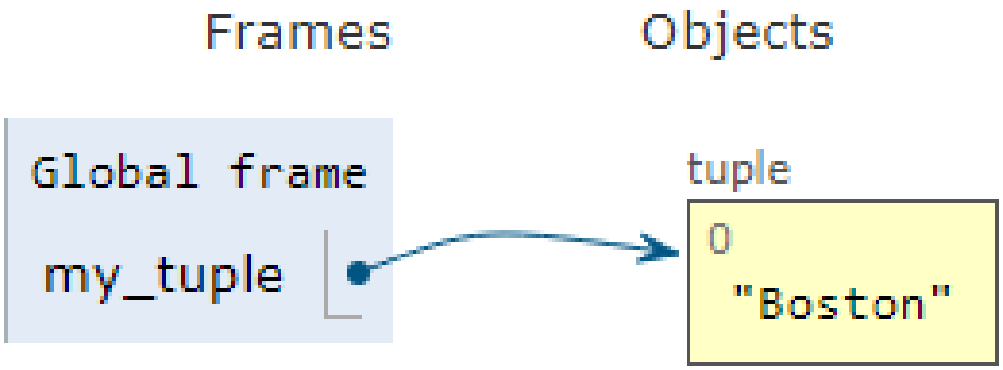
One Element Tuple

- tuple constructor is ambiguous
- use comma to indicate a tuple

```
>>> my_tuple = (“Boston”)      # string!
```




```
>>> my_tuple = (“Boston”, )    # tuple!
```



Tuple as 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 = (‘M’,‘o’,‘z’,‘z’,‘a’,‘r’,‘e’,‘l’,‘l’,a’)
>>> x.index(‘r’)
>>> 5
```

- tuple is an object
- what are tuple attributes?

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

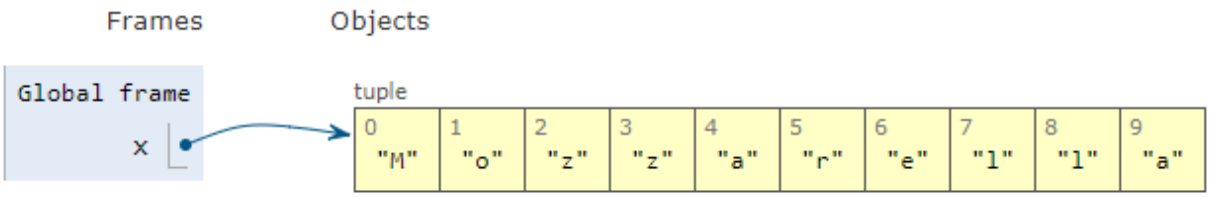
Tuple Indexing

- each tuple element has its position
- position identified by an index value

	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 = (‘M’,‘o’,‘z’,‘z’,‘a’,‘r’,‘e’,‘l’,‘l’,a’)
```

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

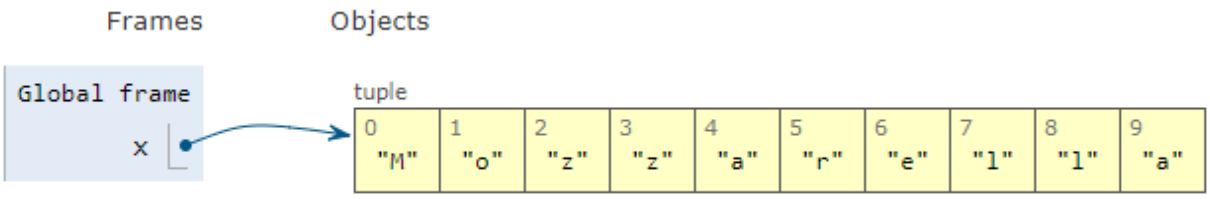


Tuple Slicing

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

```
>>> x = ('M','o','z','z','a','r','e','l','l','a')
```

	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	

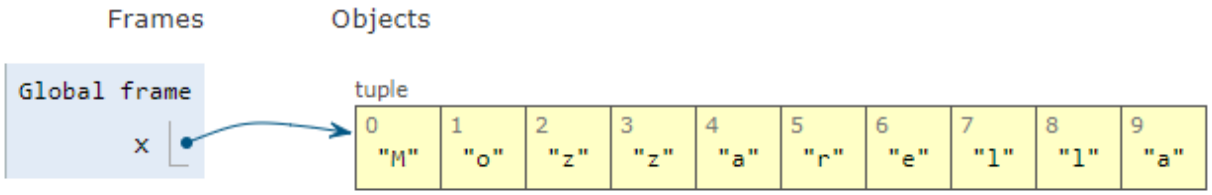


- indexing starts at 0, ends at (length – 1)
- can refer to slices but positive or negative indices

Tuple 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 = (‘M’,‘o’,‘z’,‘z’,‘a’,‘r’,‘e’,‘l’,‘l’,a’)



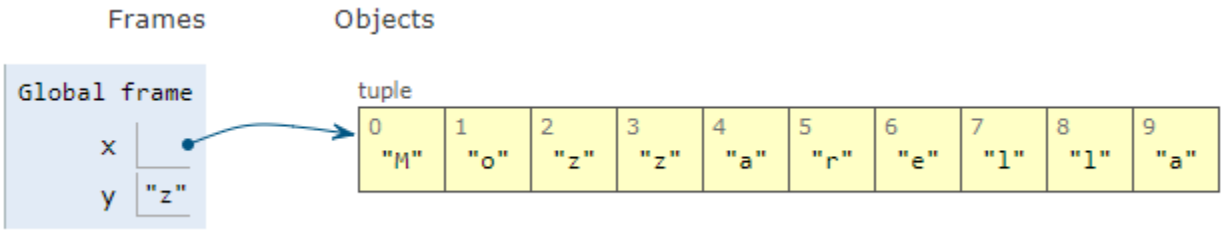
- slice operator:

variable		start		one past end	
x	(	i	:	j	)

Slicing with 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	

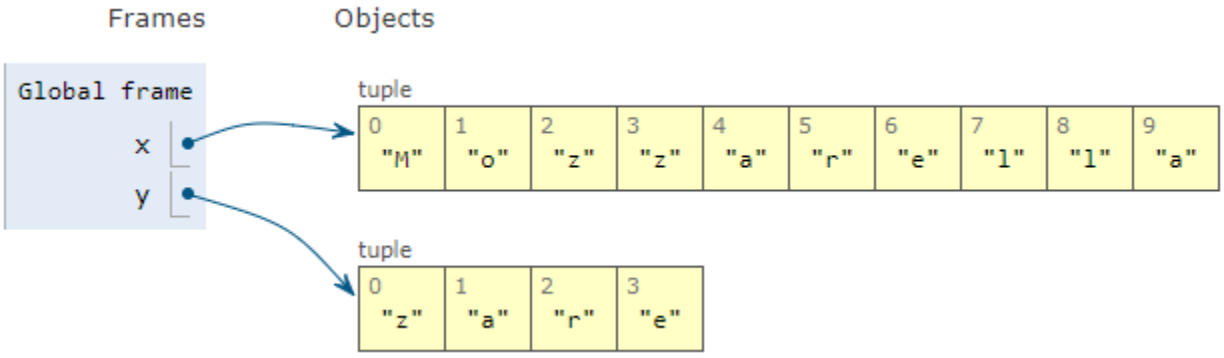
```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
>>> y = x[2]
>>> y
>>> z
```



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	

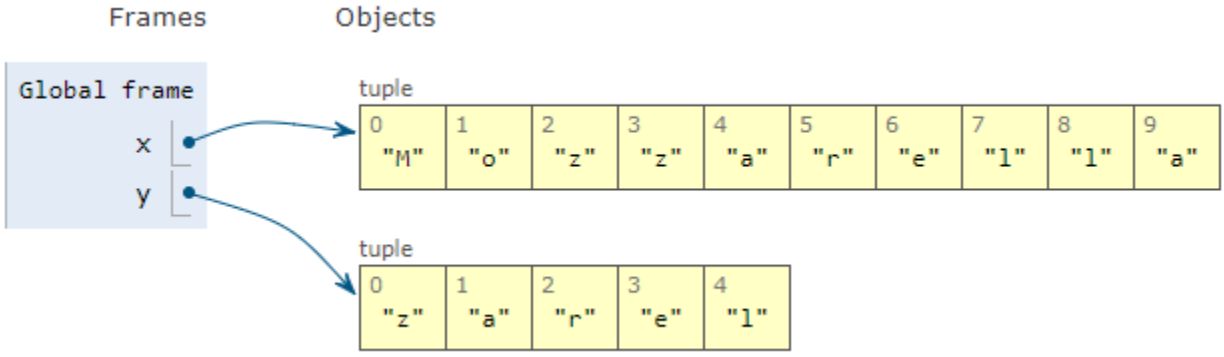
```
>>> x = (‘M’,‘o’,‘z’,‘z’,‘a’,‘r’,‘e’,‘l’,‘l’,a’)
>>> y = x[3 : 7]
>>> y
>>> ( ‘z’, ‘a’, ‘r’, ‘e’ )
```



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	

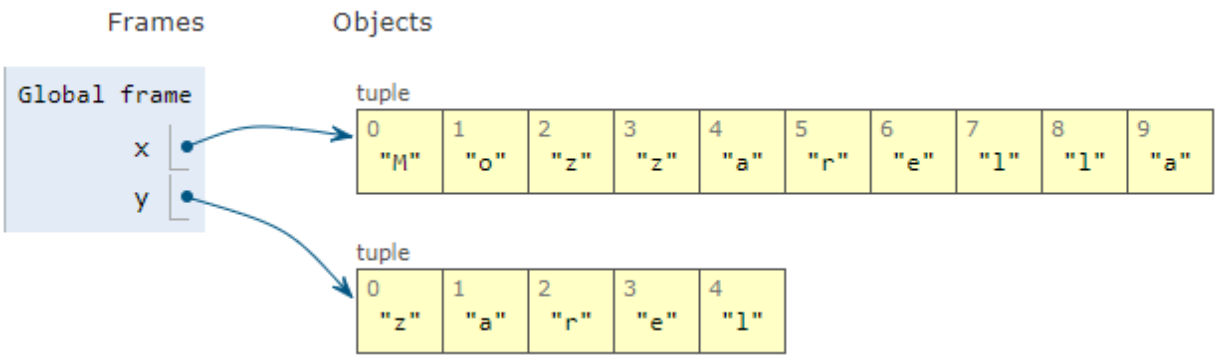
```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
>>> y = x[−7 : −2]
>>> y
>>> ( 'z', 'a', 'r', 'e', 'l' )
```



Slicing with Positive and Negative Indices

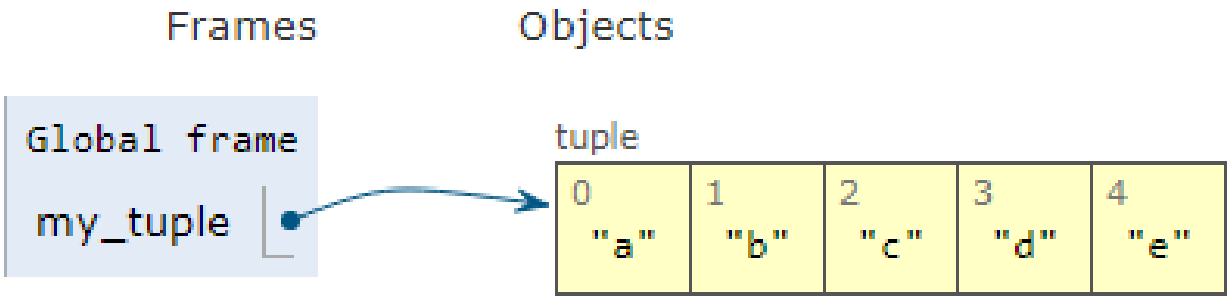
	0	1	2	3	4	5	6	7	8	9	
(	'M',	'o',	'z',	'z',	'a',	'r',	'e',	'l',	'l',	'a'	)
	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1	

```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
>>> y = x[3 : −2]
>>> y
>>> ( 'z', 'a', 'r', 'e', 'l' )
```

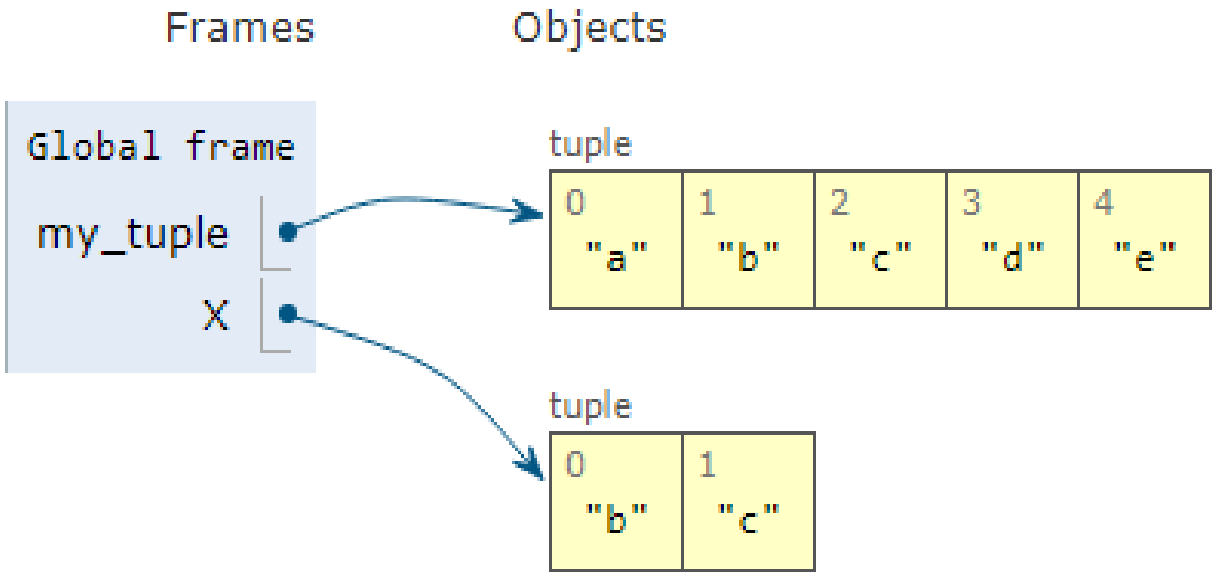


Tuple Slicing

>> my_tuple = (“a”, “b”, “c”, “d”, “e”)

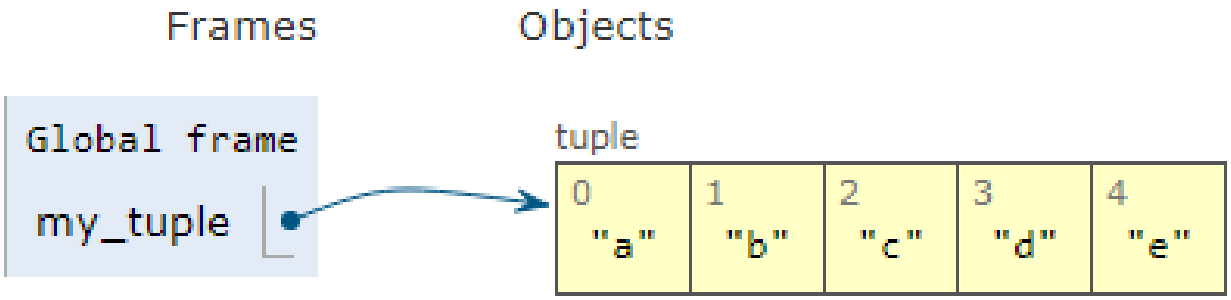


>> X = my_tuple[1:3]

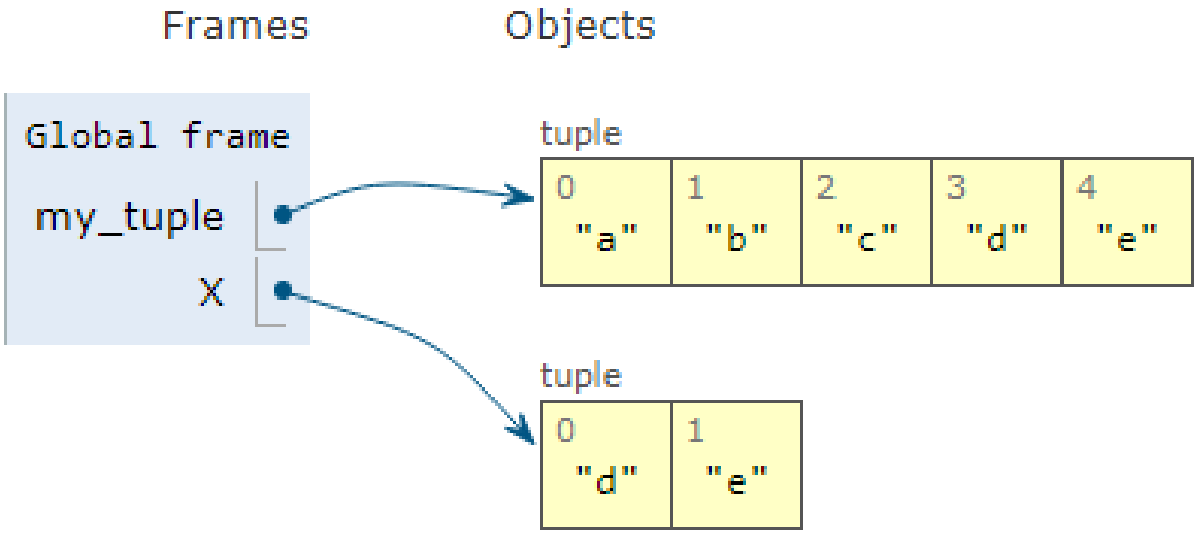


Tuple Slicing (cont'd)

>> my_tuple = ("a", "b", "c", "d", "e")



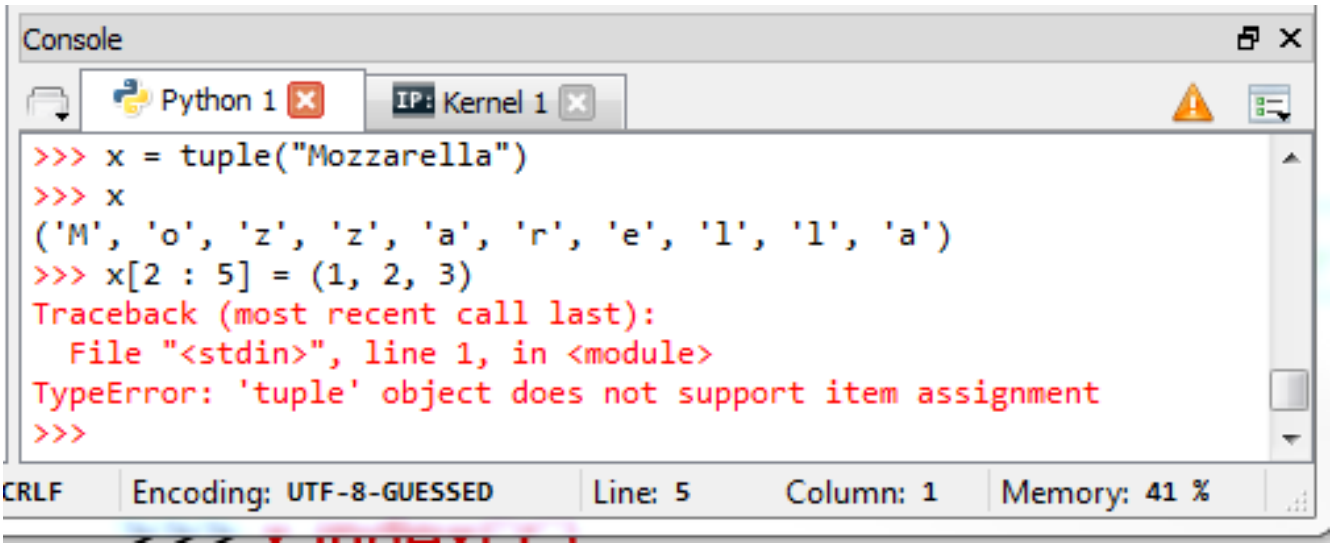
>> X = my_tuple[3:]



Tuple Slice Replacement

	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	

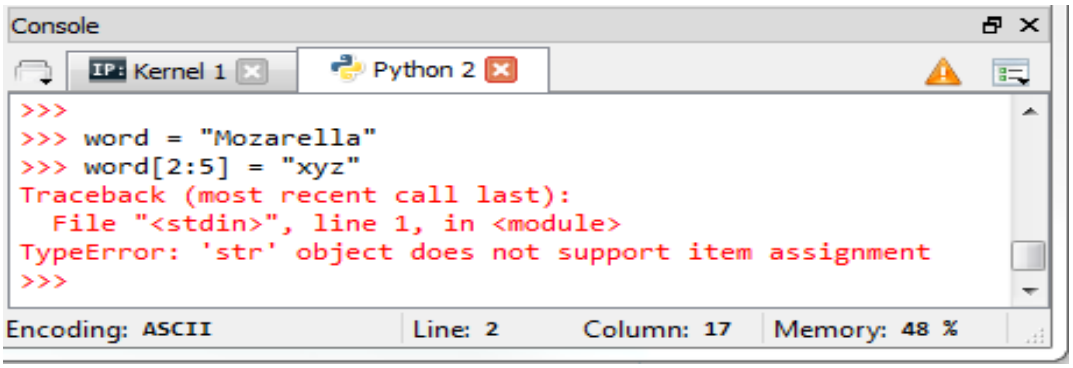
- tuples are immutable (like strings)
- cannot change sub-tuples without creating new tuple!!!



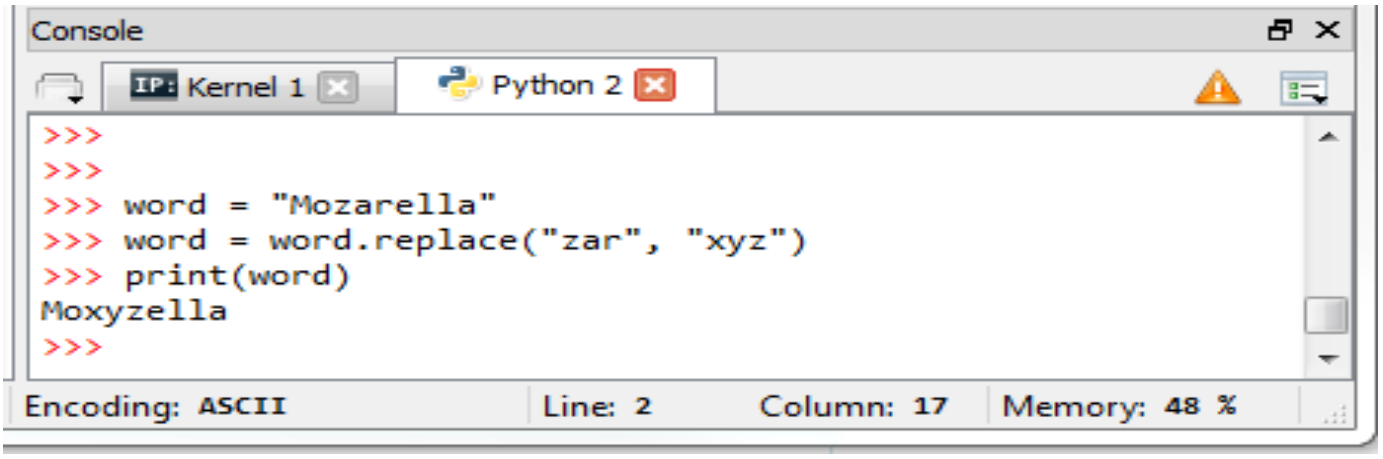
Valid vs. Invalid Slicing

	0	1	2	3	4	5	6	7	8	
(	'M'	'o'	'z'	'a'	'r'	'e'	'l'	'l'	'a'	)
	-9	-8	-7	-6	-5	-4	-3	-2	-1	

- why is this illegal?



- but this is legal !



Tuple 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	

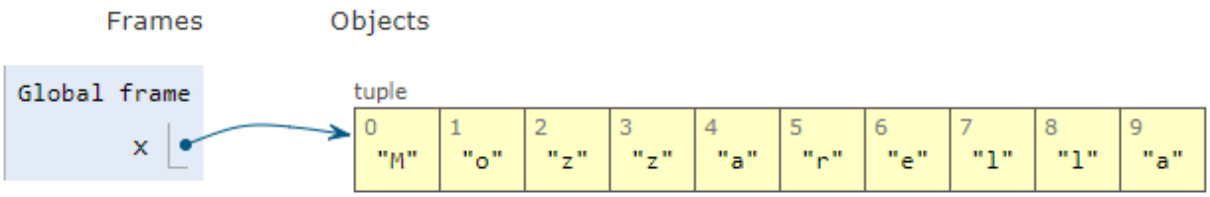
- what is the result of these operations?

```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
>>> y = x[−1]
>>> z = x[2 : 6]
>>> w = x[−7 : −3]
>>> v = x[ : ]
```

Tuple 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	

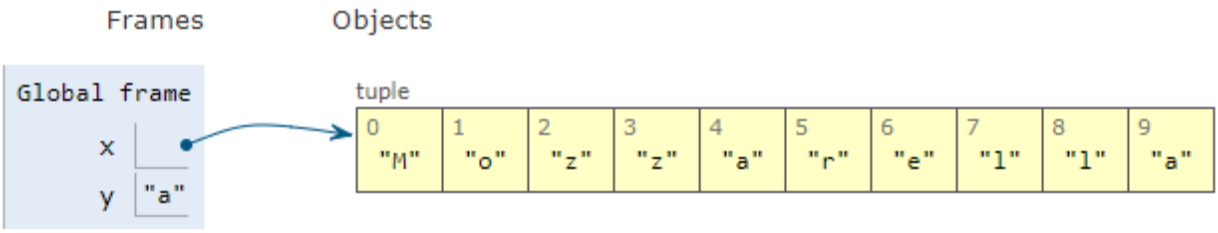
```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
```



Tuple 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	

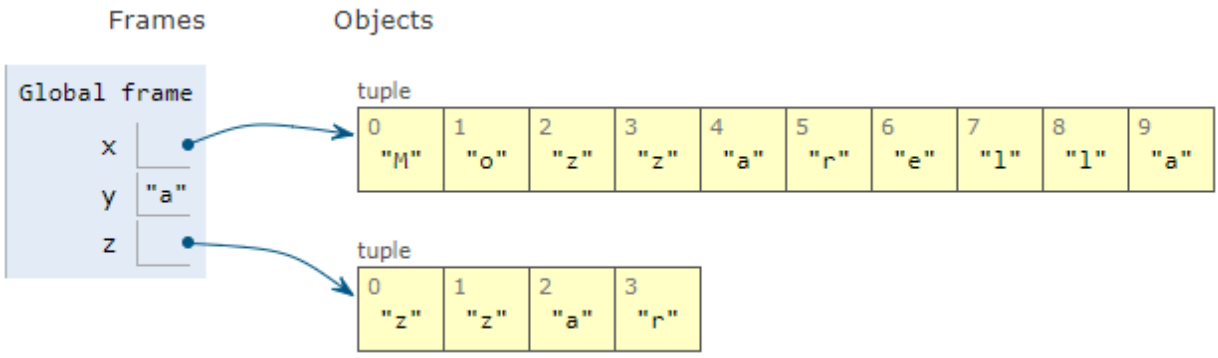
```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
>>> y = x[−1]
>>> a
```



Tuple 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	

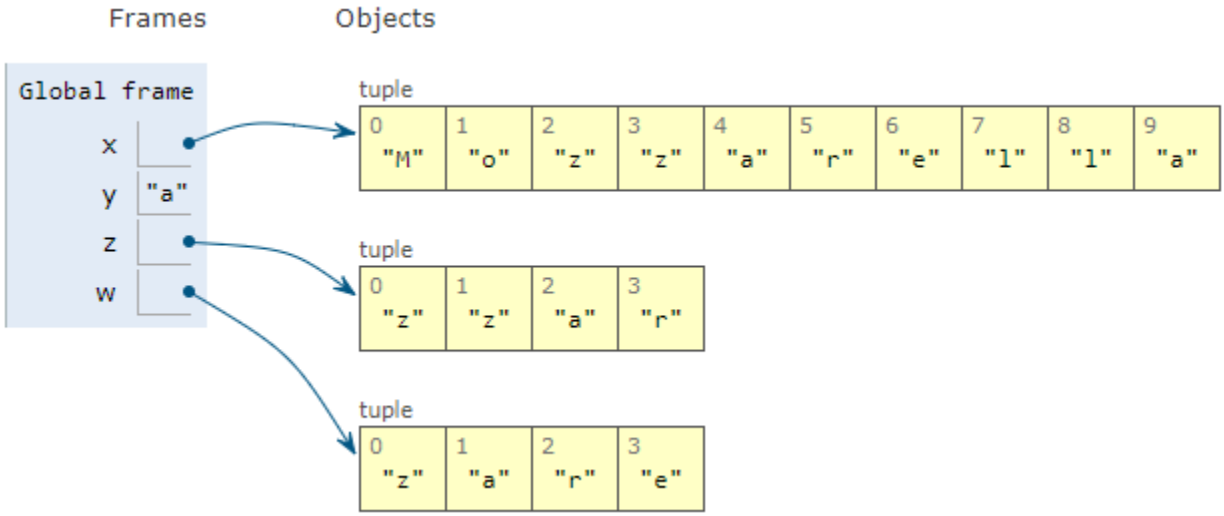
```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
>>> y = x[−1]
>>> z = x[2 : 6]
```



Tuple 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	

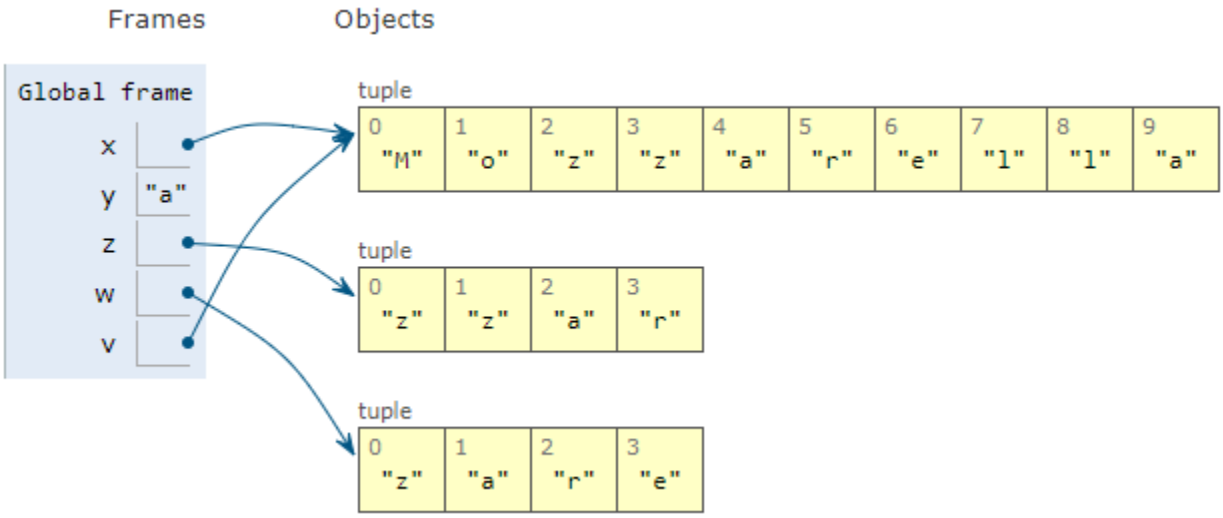
```
>>> x = ('M','o','z','z','a','r','e','l','l',a')
>>> y = x[−1]
>>> z = x[2 : 6]
>>> w = x[−7 : −3]
```



Tuple 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 = ('M','o','z','z','a','r','e','l','l','a')
>>> y = x[−1]
>>> z = x[2 : 6]
>>> w = x[−7 : −3]
>>> v = x[ : ]
```

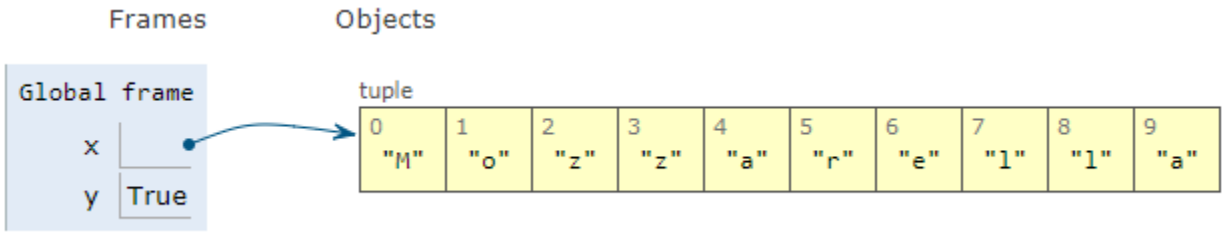


- example of a "shallow" copy

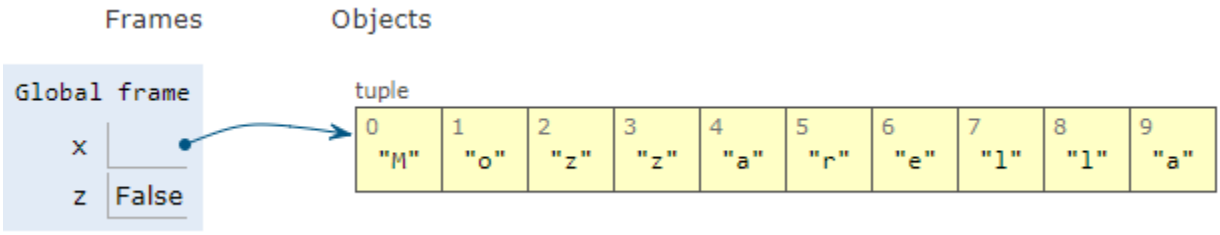
Tuple Membership: *in*, *not in*

	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 = ('M','o','z','z','a','r','e','l','l',a')
>>> y = 'r' in x
```

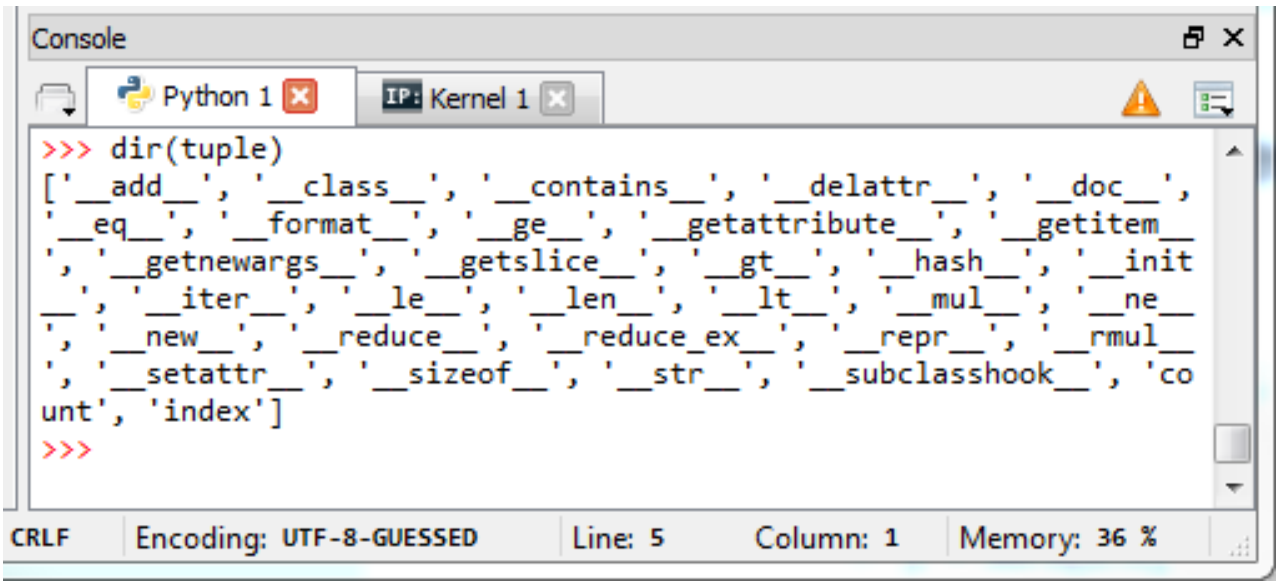



```
>>> z = "a" not in x
```



Tuple Attributes

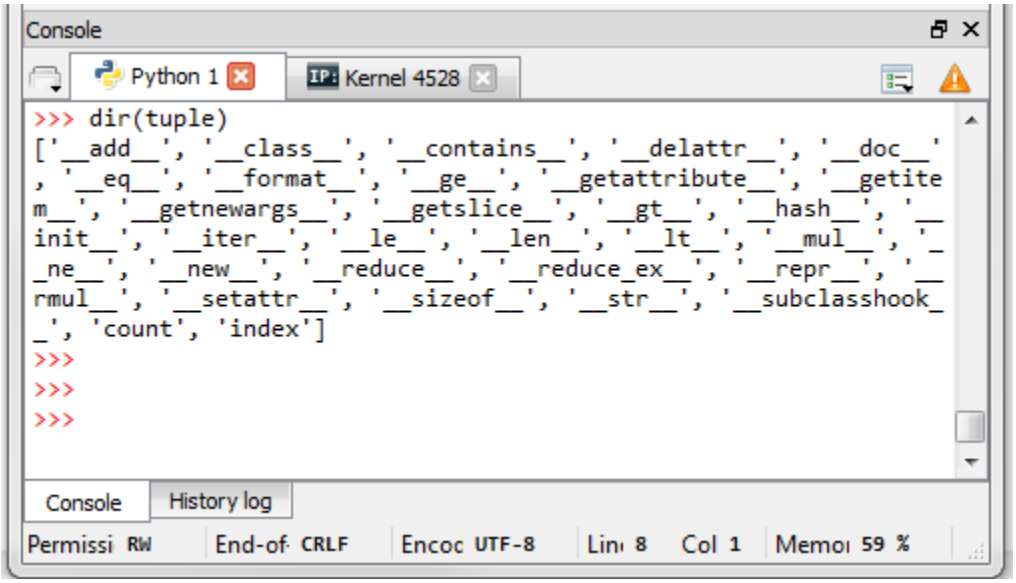
- use dir(tuple) command



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

tuple Type Methods

```
>>> dir(tuple)
```



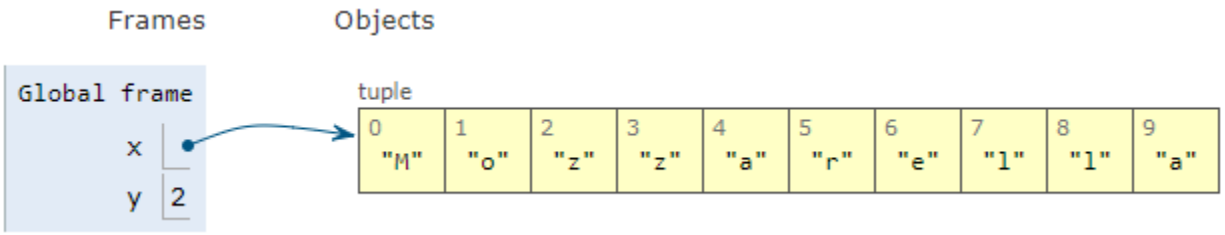

```
>>> ('Mozzarella', 'Brie').__len__()
2
>>> ('Mozzarella', 'Brie').__sizeof__()
40
```

Tuple Methods

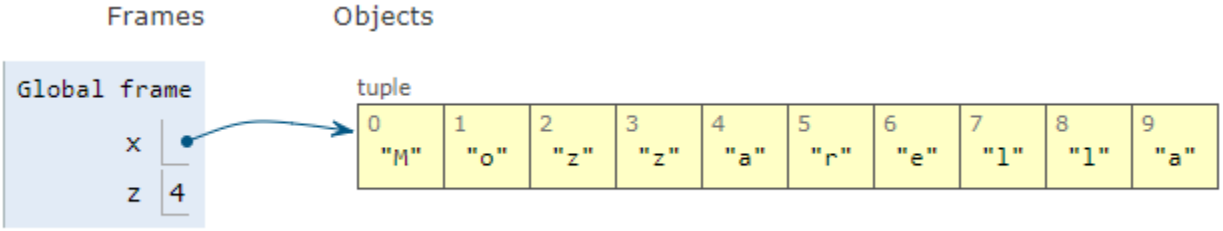
	0	1	2	3	4	5	6	7	8	9	
(	‘M’,	‘o’,	‘z’,	‘z’,	‘a’,	‘r’,	‘e’,	‘l’,	‘l’,	‘a’	)
	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1	

- cannot add/remove items for tuples
- two methods: *count*(x) and *index*(x)

```
>>> x = (‘M’,‘o’,‘z’,‘z’,‘a’,‘r’,‘e’,‘l’,‘l’,a’)
>>> y = x.count(‘a’)
```

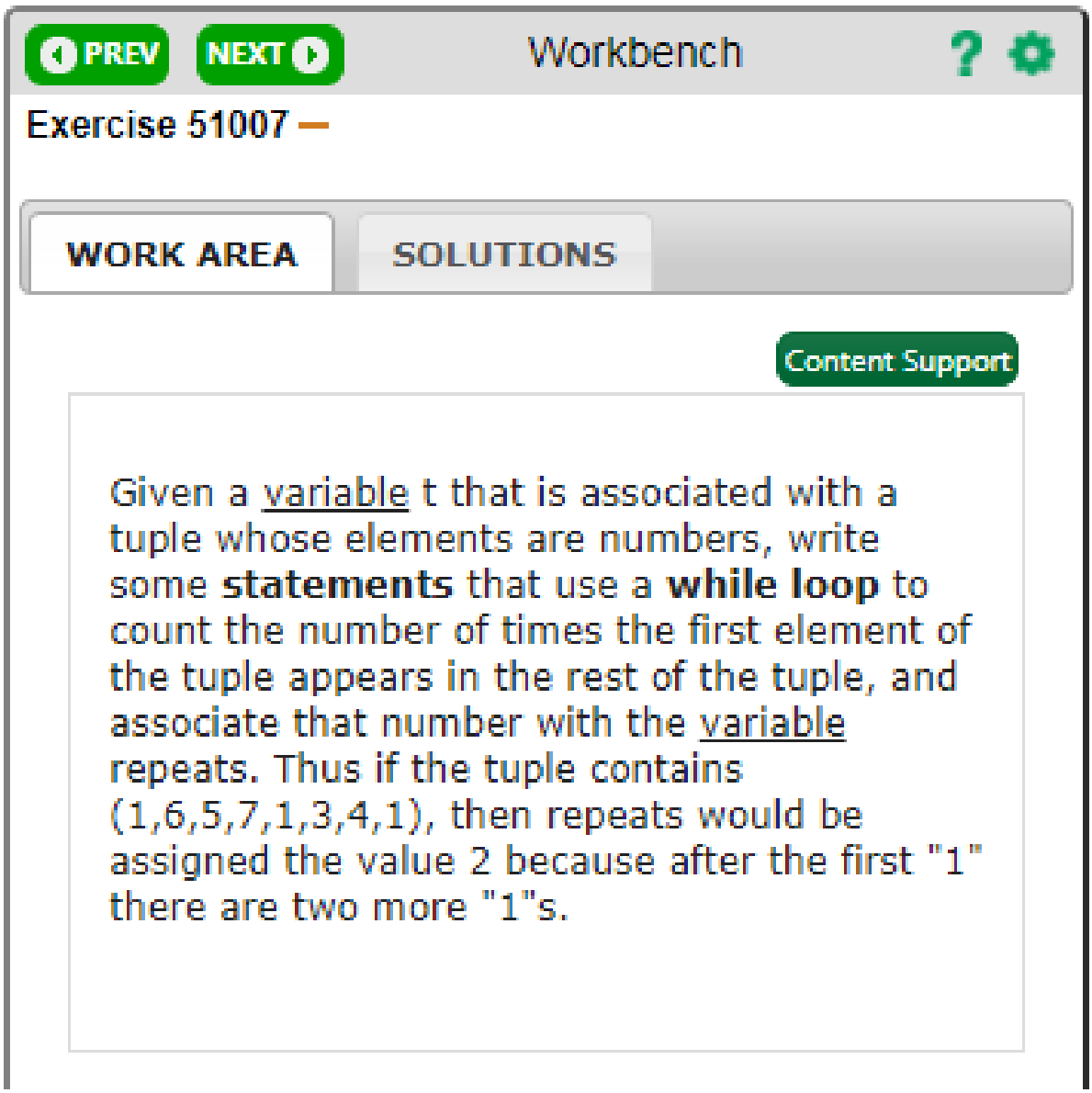



```
>>> z = x.index(‘a’)
```



Review Problems

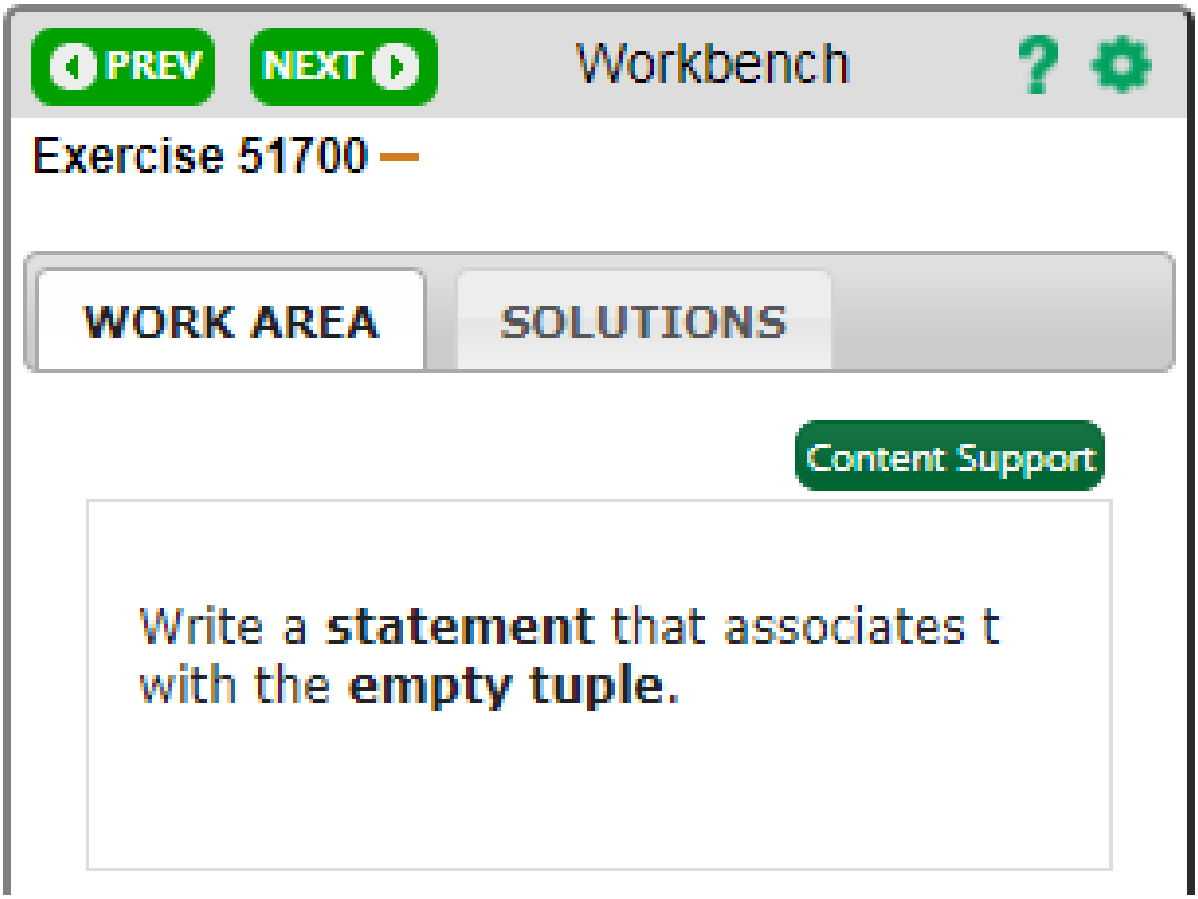
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

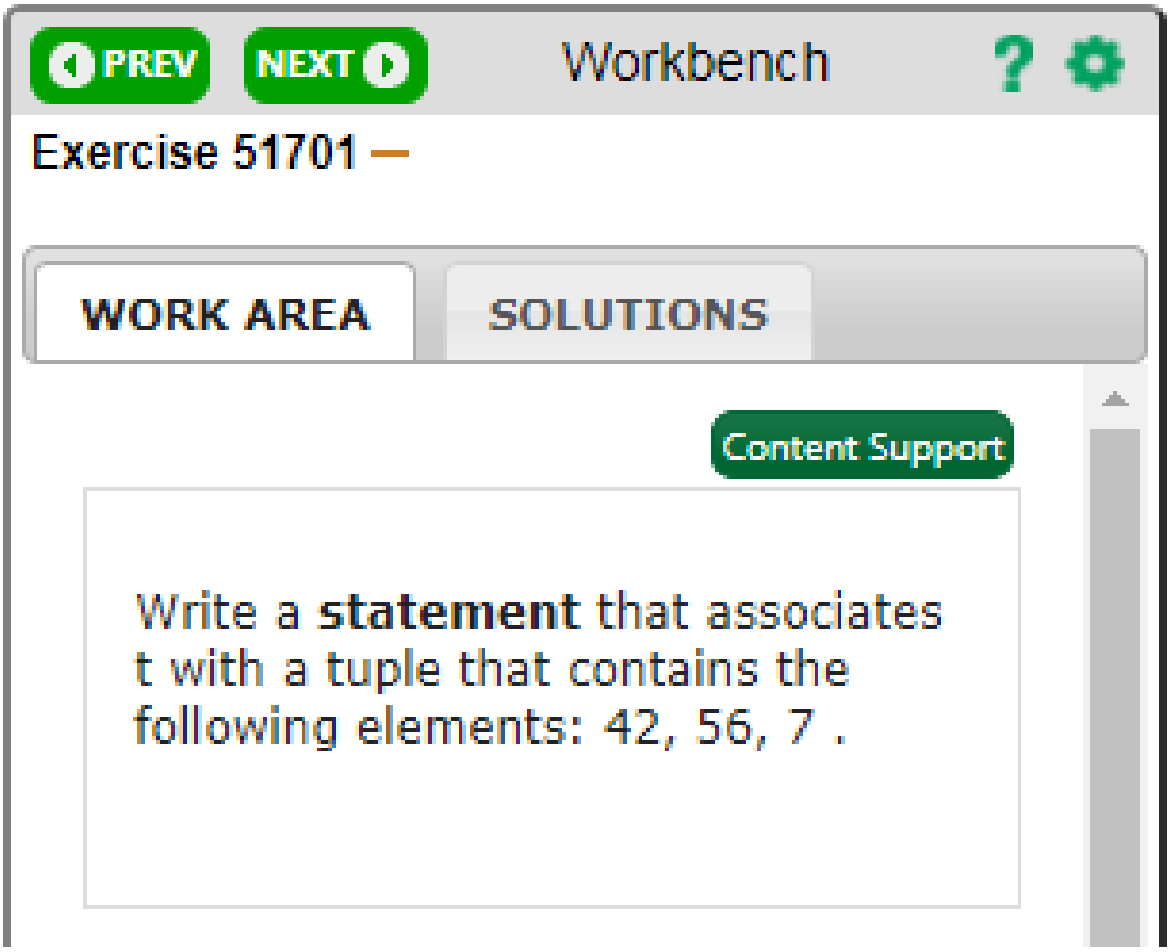
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

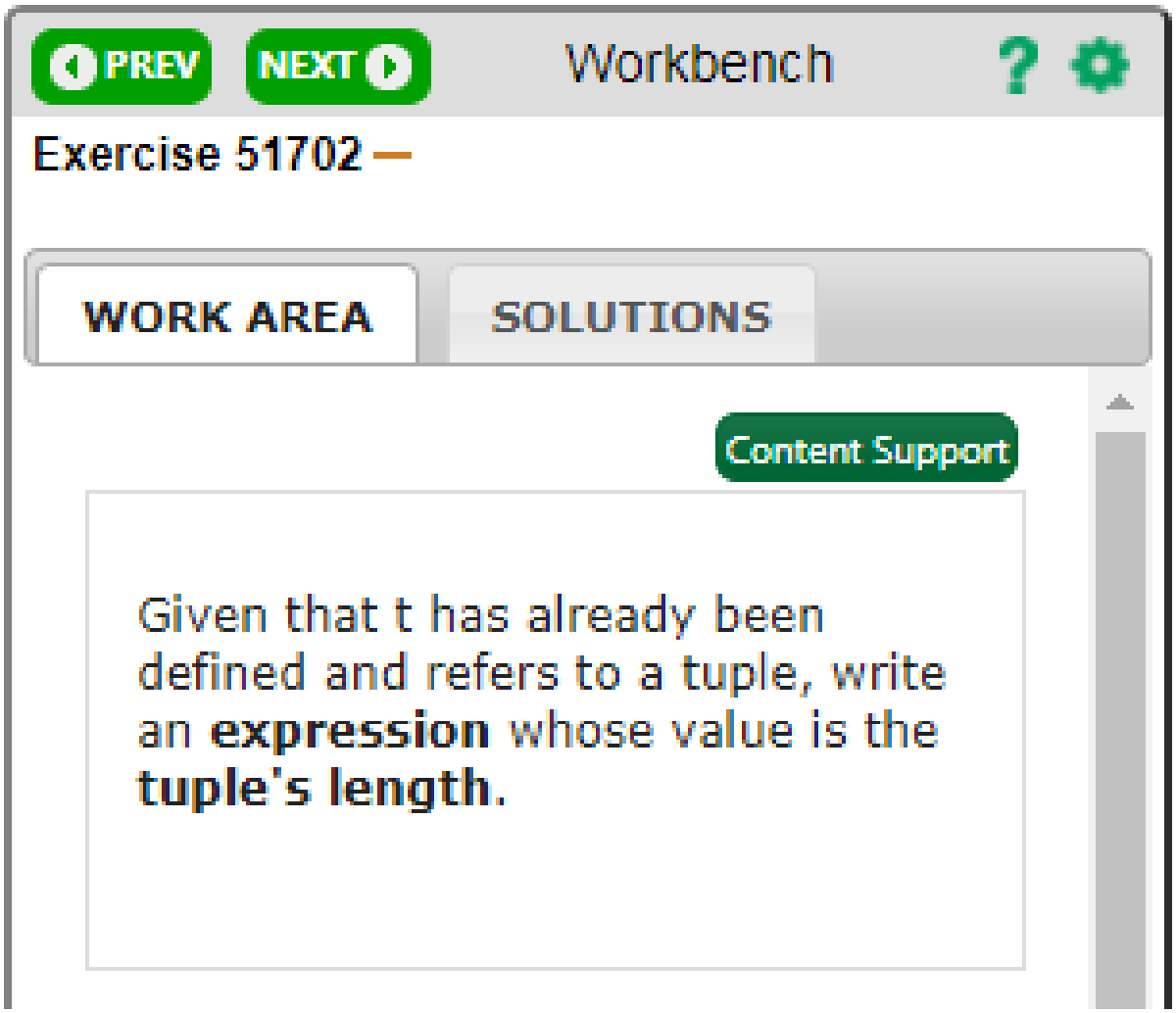
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

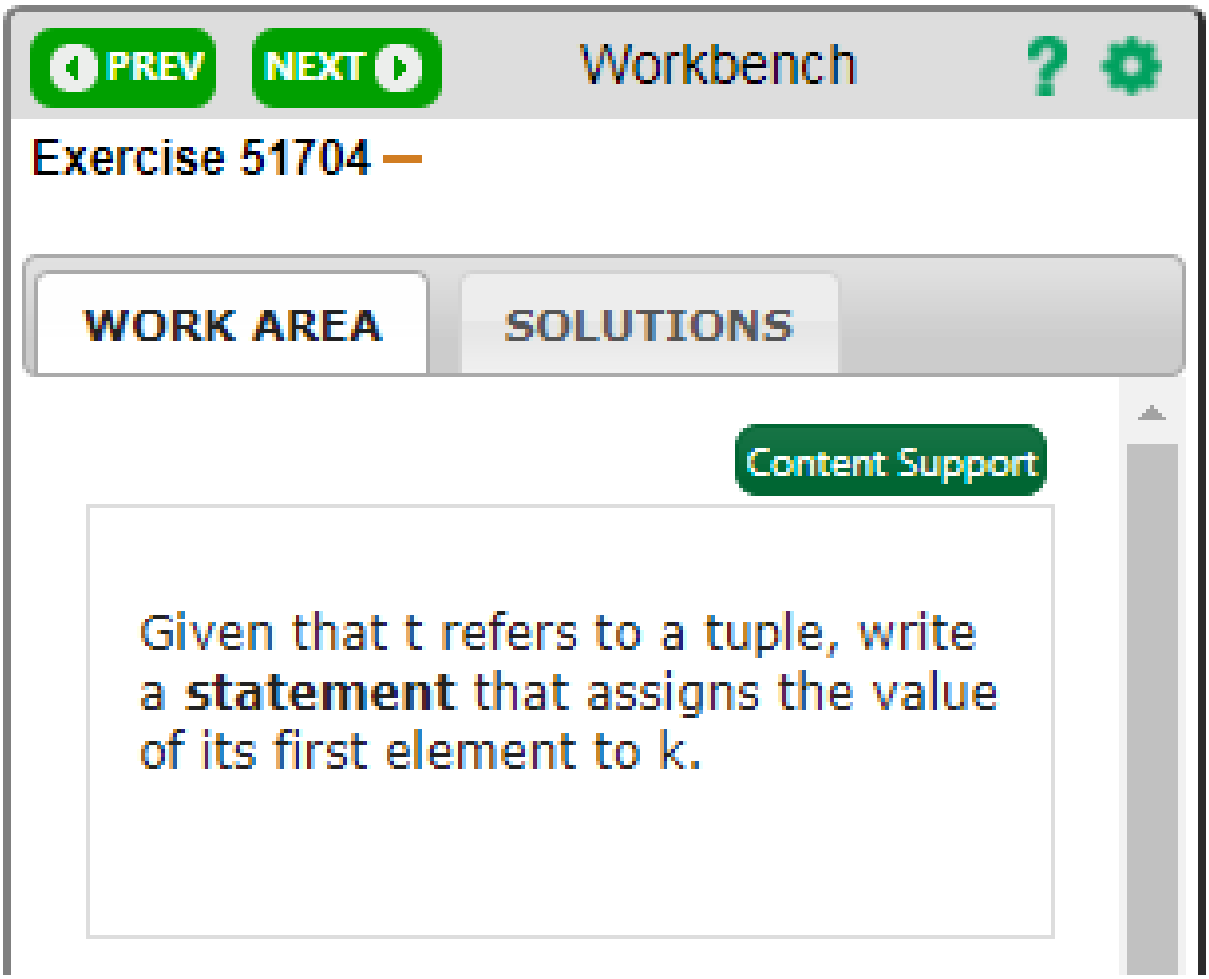
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

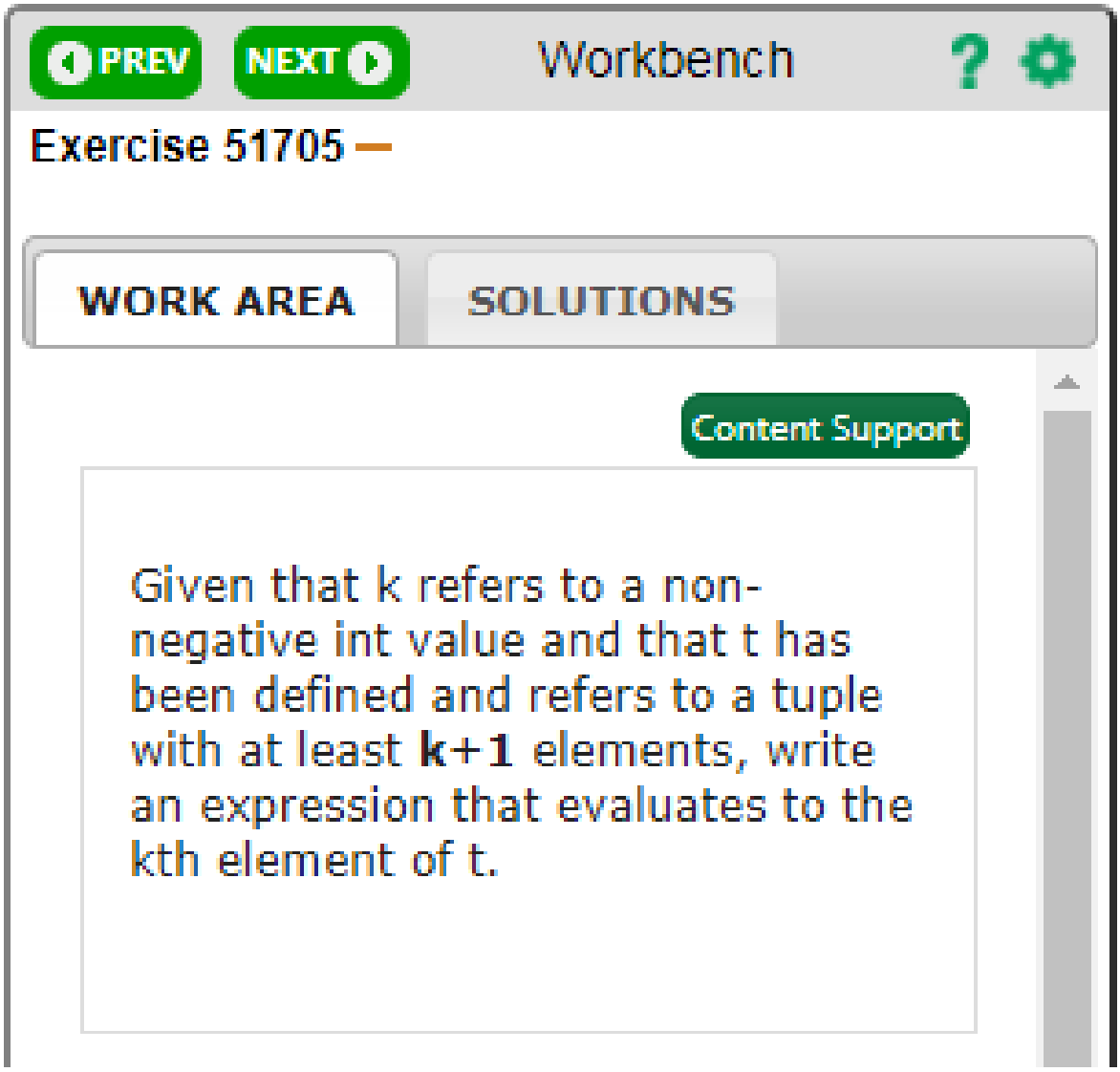
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

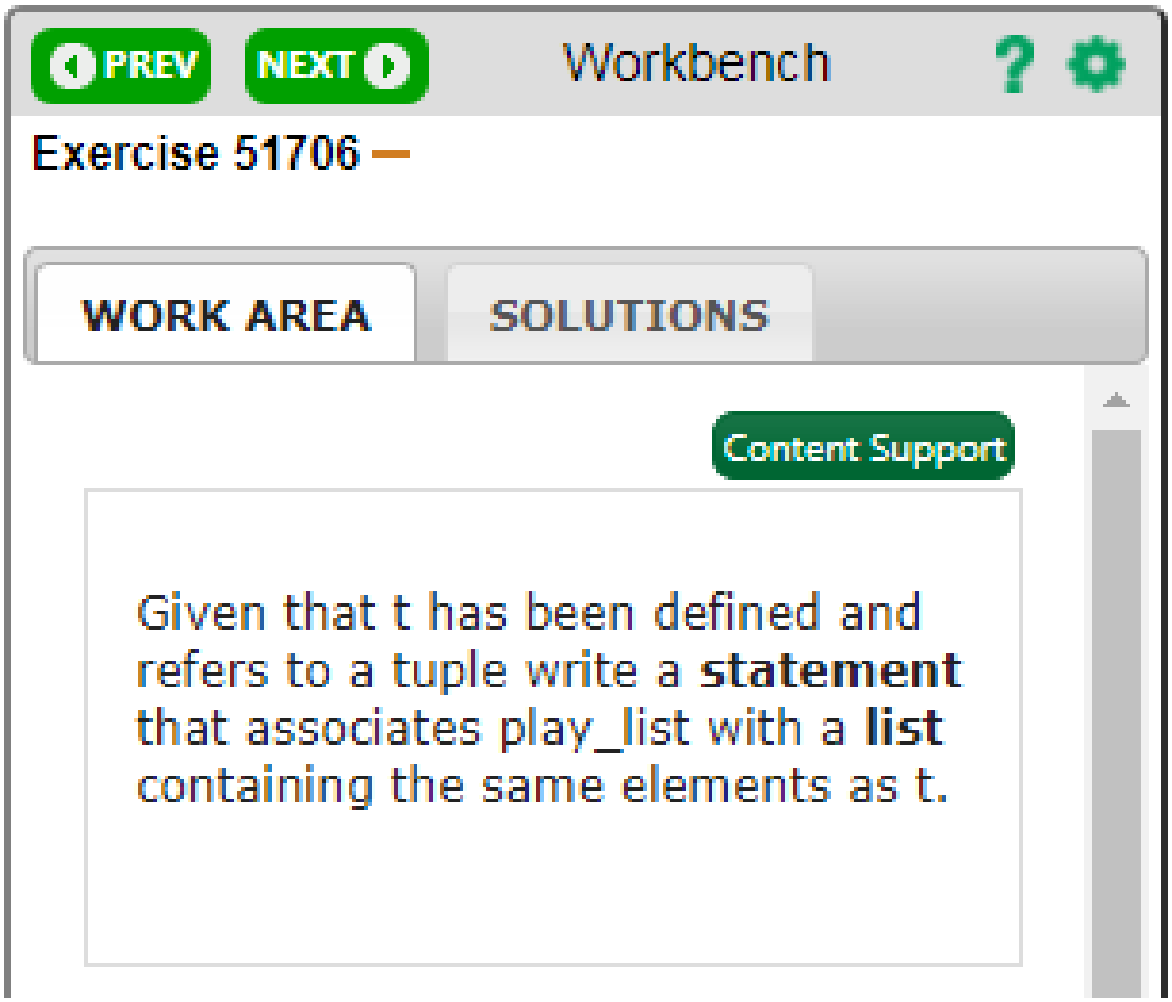
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

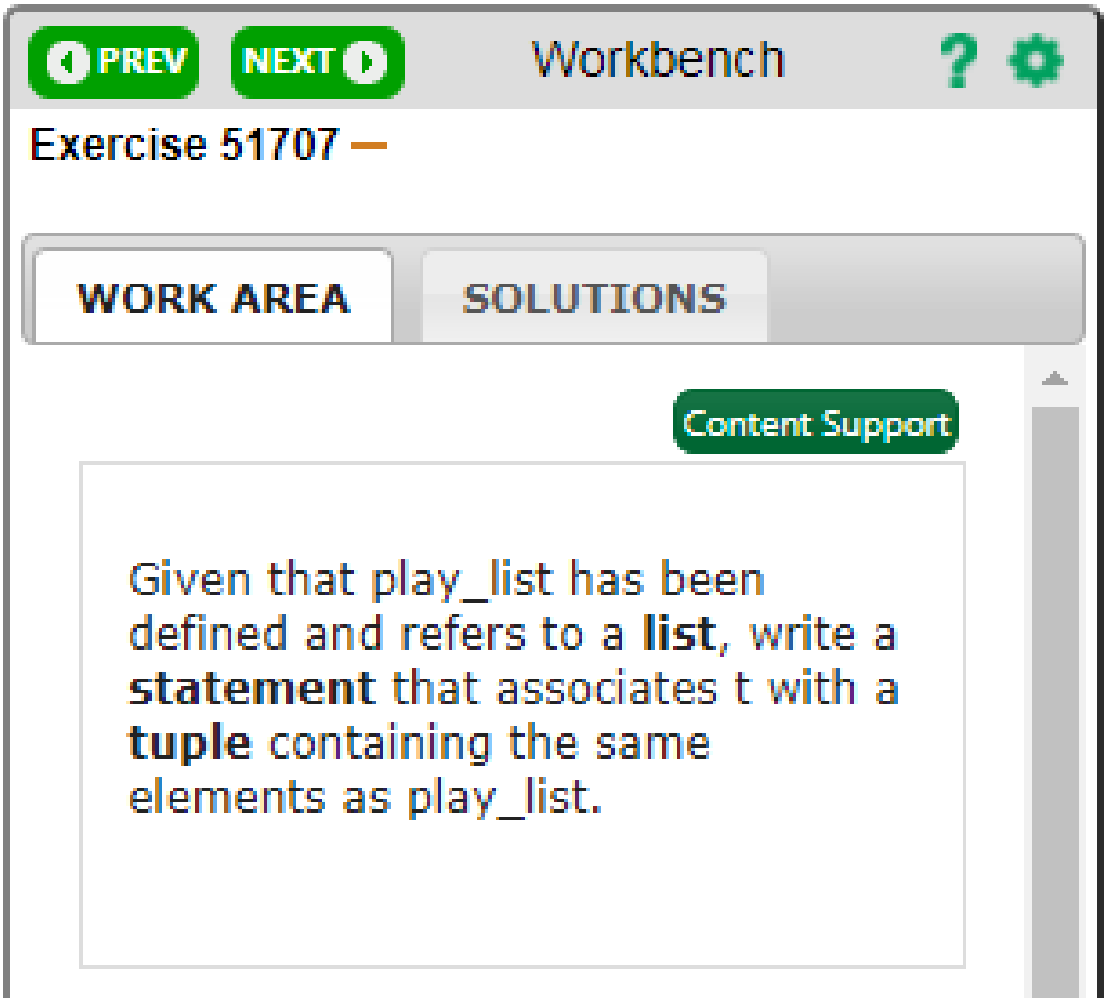
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

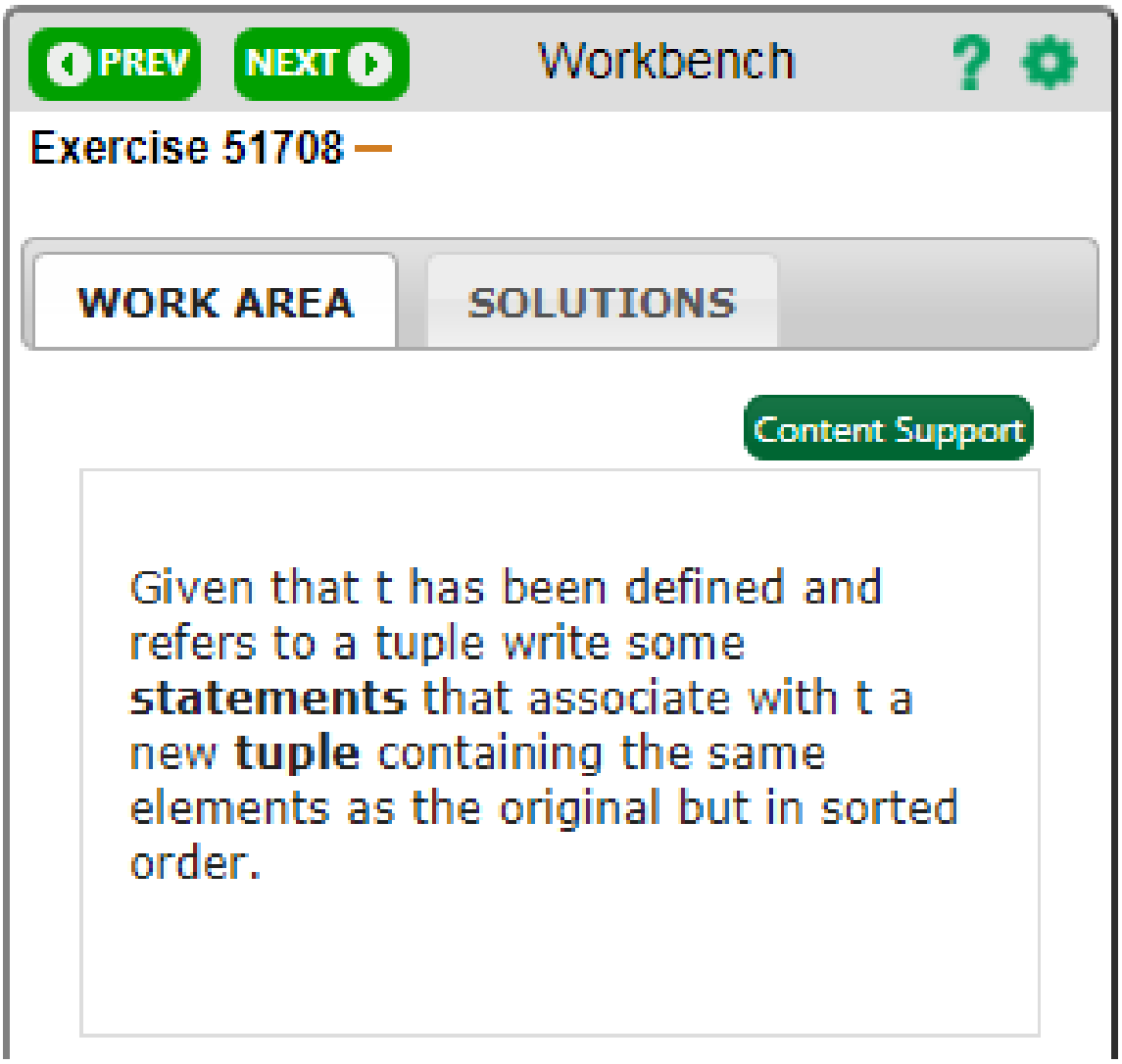
Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank

Programming Exercise



Programming Exercise Worksheet

this page is intentionally left blank