

# Python

# Tuples

# Tuple Attributes

- use `dir(tuple)` command

The screenshot shows a Jupyter Notebook interface with a 'Console' tab. It displays the output of the Python command `dir(tuple)`. The output lists numerous methods and attributes starting with double underscores (dunder methods), which are local to the tuple class. The list includes: \_\_add\_\_, \_\_class\_\_, \_\_contains\_\_, \_\_delattr\_\_, \_\_doc\_\_, \_\_eq\_\_, \_\_format\_\_, \_\_ge\_\_, \_\_getattribute\_\_, \_\_getitem\_\_, \_\_getnewargs\_\_, \_\_getslice\_\_, \_\_gt\_\_, \_\_hash\_\_, \_\_init\_\_, \_\_iter\_\_, \_\_le\_\_, \_\_len\_\_, \_\_lt\_\_, \_\_mul\_\_, \_\_ne\_\_, \_\_new\_\_, \_\_reduce\_\_, \_\_reduce\_ex\_\_, \_\_repr\_\_, \_\_rmul\_\_, \_\_setattr\_\_, \_\_sizeof\_\_, \_\_str\_\_, \_\_subclasshook\_\_, \_\_count, and \_\_index\_\_. The console also shows the standard status bar at the bottom with 'CRLF', 'Encoding: UTF-8-GUESSED', 'Line: 5', 'Column: 1', and 'Memory: 36 %'.

```
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
 '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__',
 '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__',
 '__setattr__', '__sizeof__', '__str__', '__subclasshook__',
 '__count', '__index__']
>>>
```

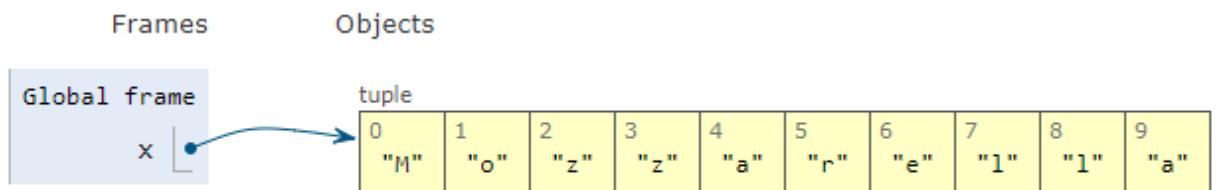
- names starting with `__` refer to “local” class names/methods

# 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		

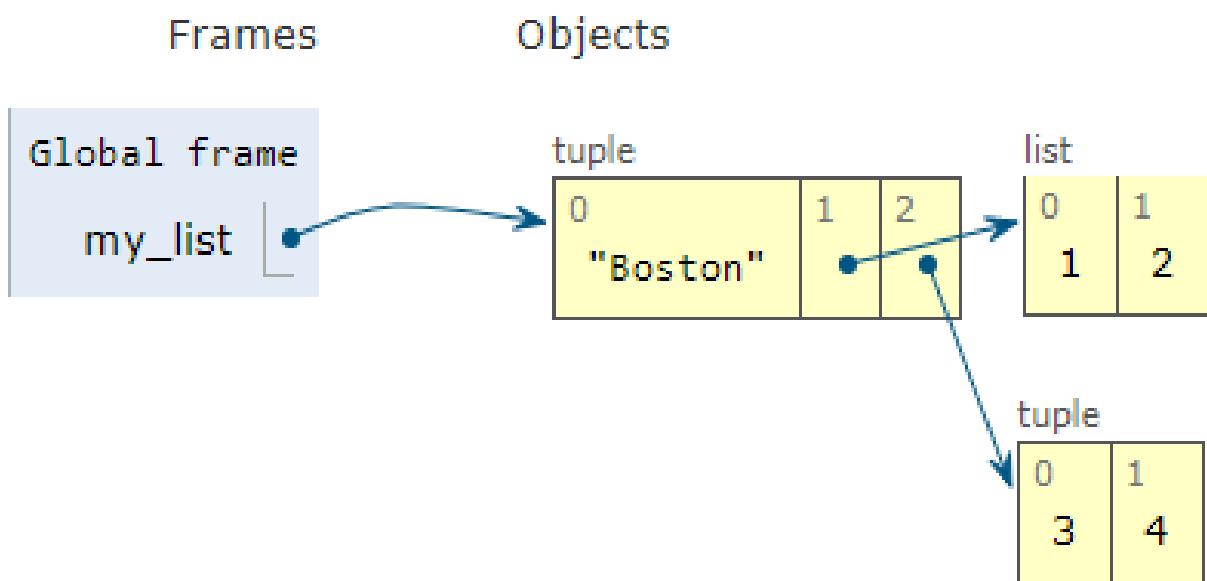
```
>>> x = ('M','o','z','z','a','r','e','l','l',a)
```



# Tuple with Mixed Types

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

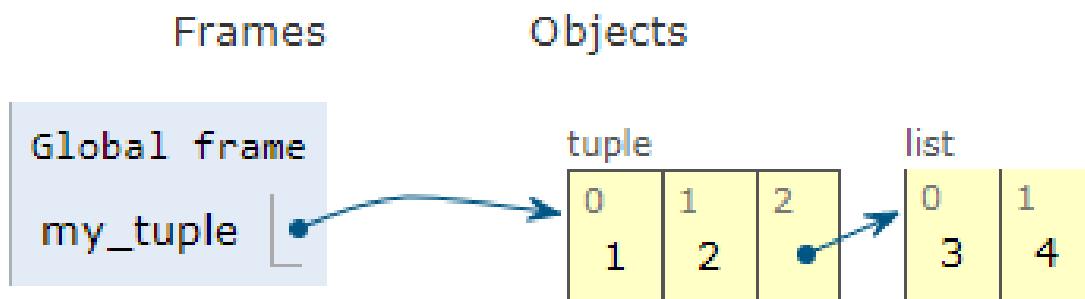
```
>>> my_tuple = ("Boston", [1, 2], (4, 5))
```



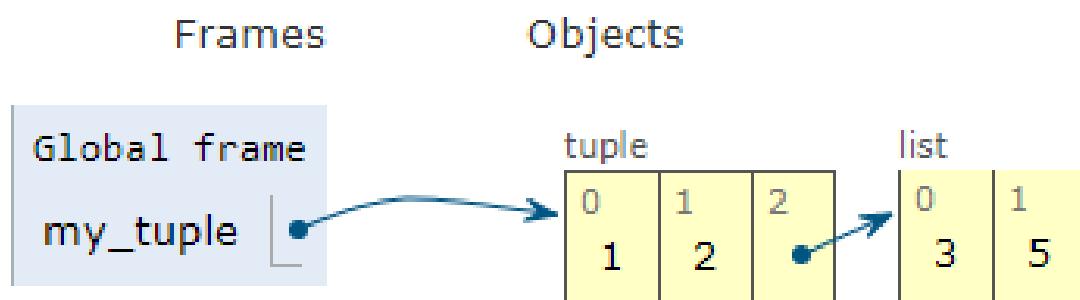
# Tuple Immutability

- 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 = "Mozzarella", "Brie"
```

```
>>> choices
```

```
("Mozzarella", "Brie")
```

- unpacking: distribute tuple elements across variables

```
>>> choices = "Mozzarella", "Brie"
```

```
>>> choice_1, choice_2 = choices
```

```
>>> choice_2
```

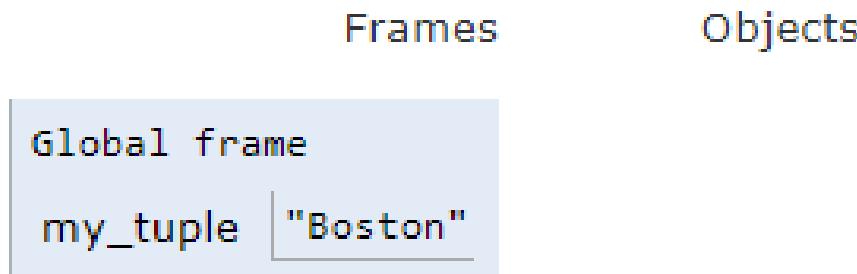
```
"Brie"
```

- similar unpacking for lists

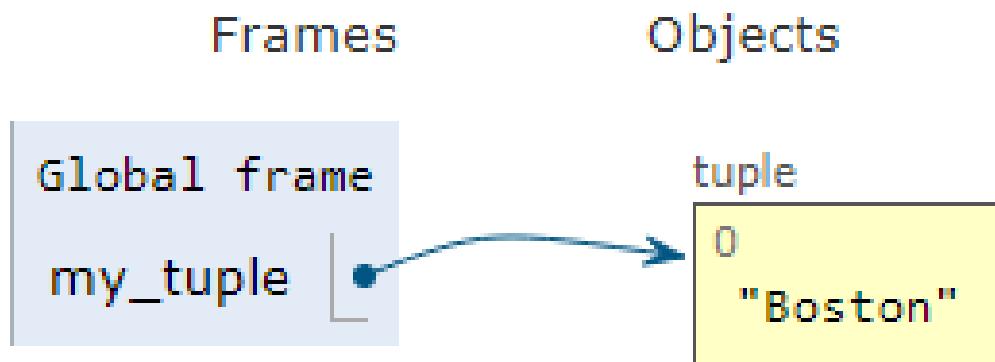
# One Element Tuple

- 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')

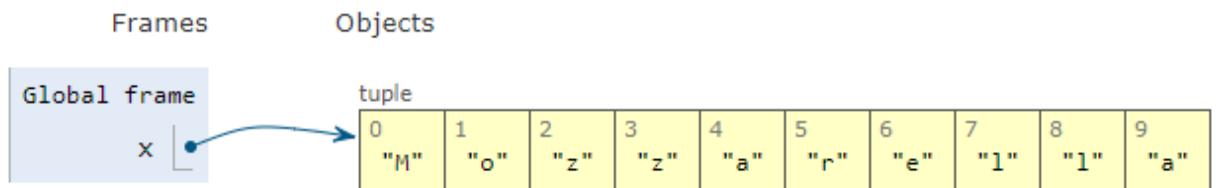
# Tuple Indexing

- 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

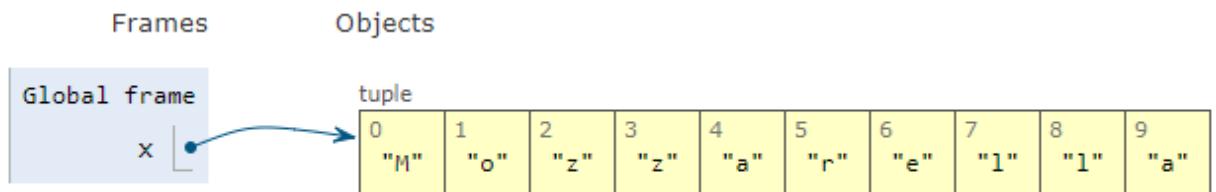


# Tuple Slicing

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

```
>>> x = ('M','o','z','z','a','r','e','T','T',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		

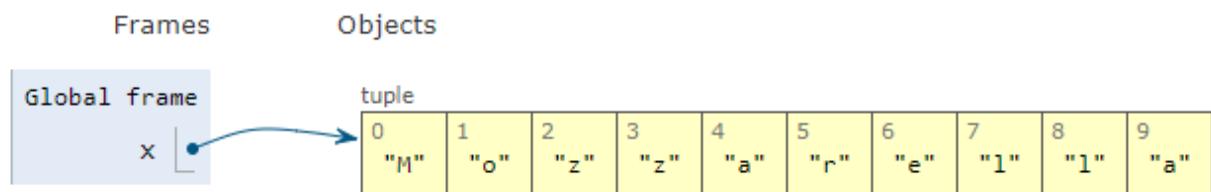


- indexing starts at 0, ends at  $(\text{length} - 1)$
  - can refer to slices but positive or negative indices

# 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)
```



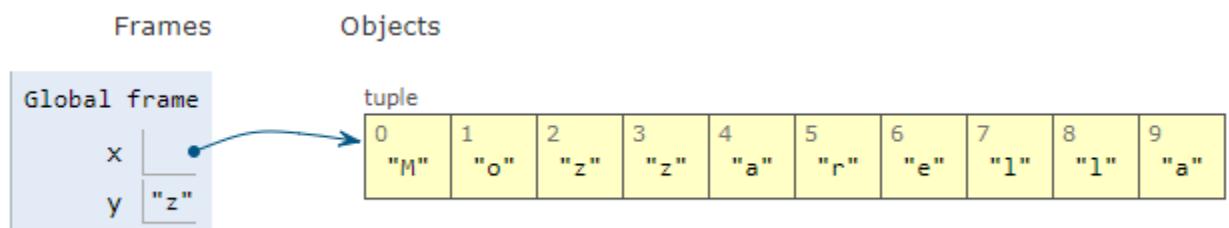
- slice operator:

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

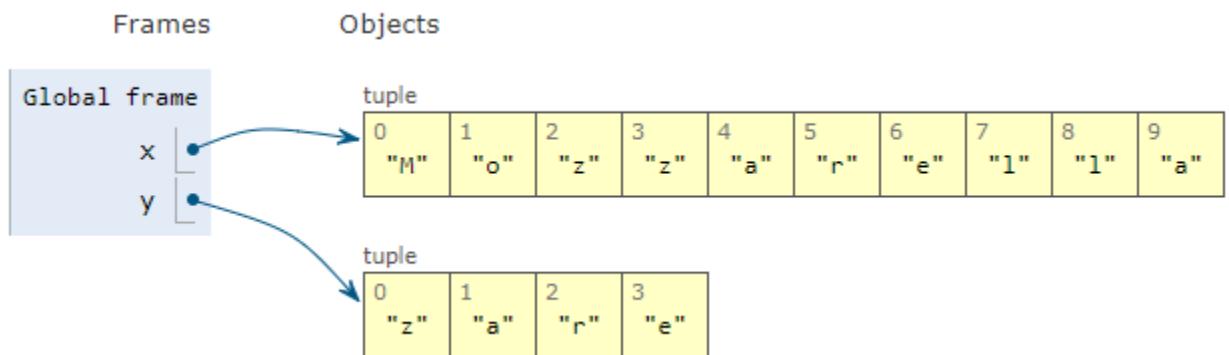
```
>>> x = ('M','o','z','z','a','r','e','l','l',a)  
>>> y = x[2]  
>>> y  
>>> z
```



# 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		

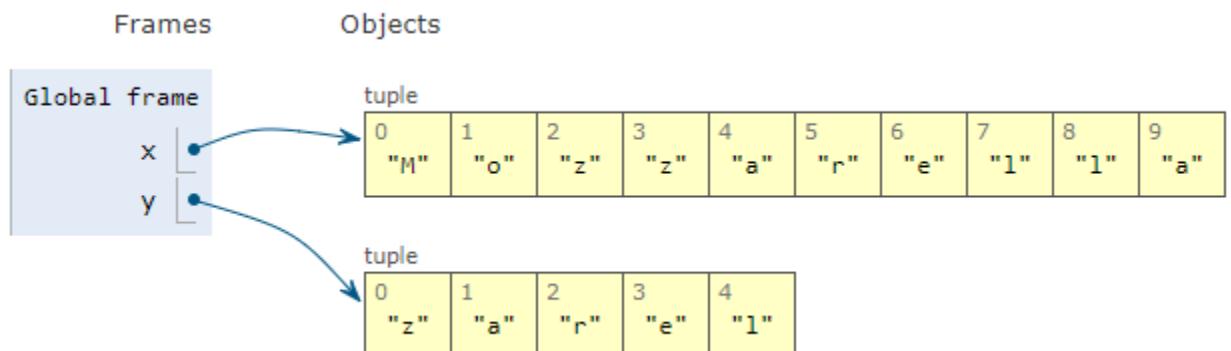
```
>>> x = ('M','o','z','z','a','r','e','l','l',a)  
>>> y = x[3 : 7]  
>>> y  
>>> ('z', 'a', 'r', 'e')
```



# 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		

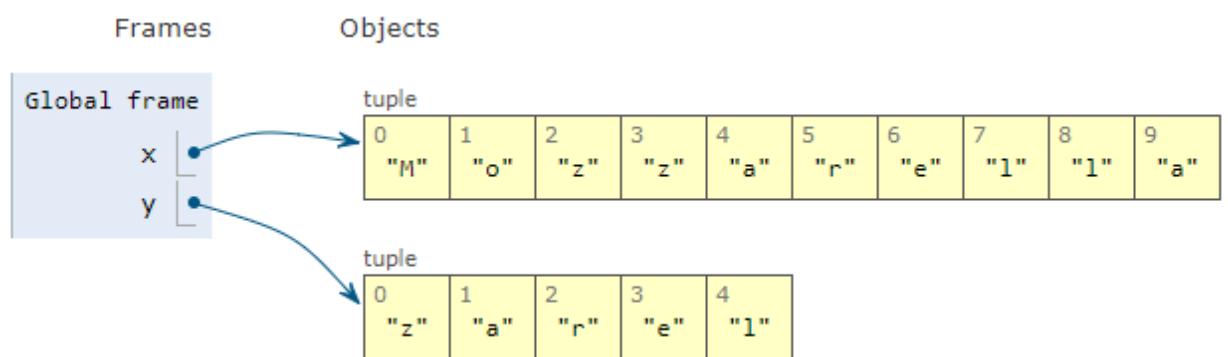
```
>>> x = ('M','o','z','z','a','r','e','l','l',a)
>>> y = x[-7 : -2]
>>> y
>>> ( 'z', 'a', 'r', 'e', 'l' )
```



# Slicing with Positive and Negative Indices

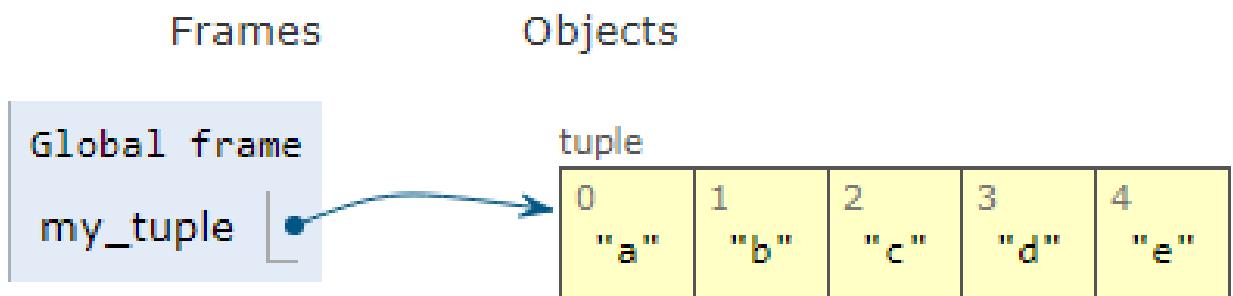
	0	1	2	3	4	5	6	7	8	9	
(	‘M’	‘o’	‘z’	‘z’	‘a’	‘r’	‘e’	‘l’	‘l’	‘a’	)
	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	

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

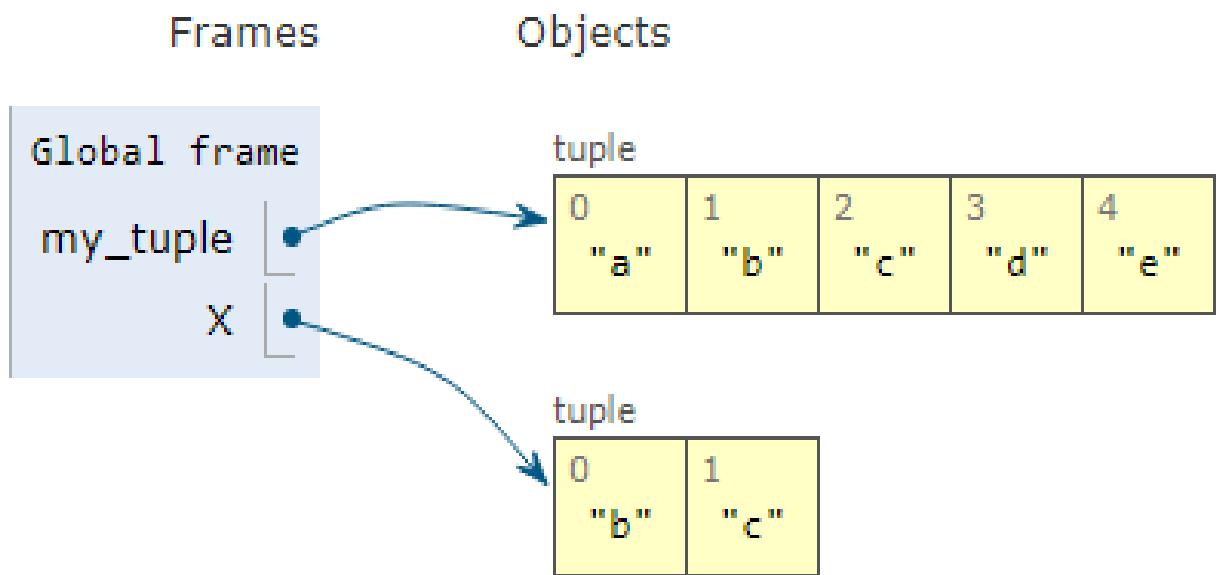


# Tuple Slicing

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

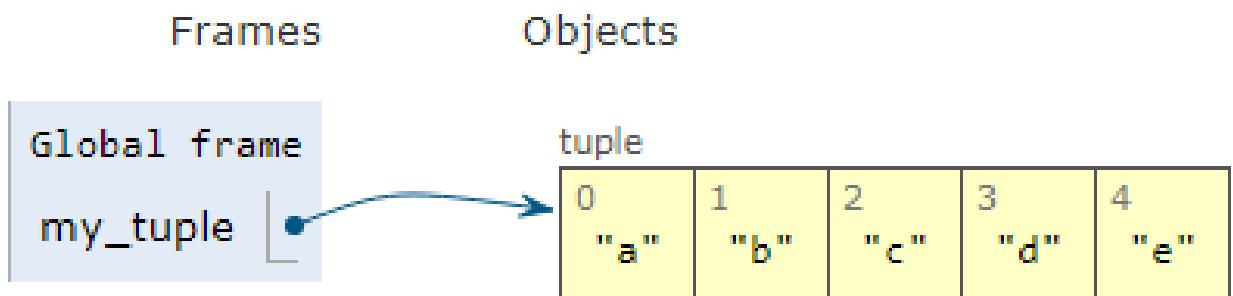


```
>> X = my_tuple[1:3]
```

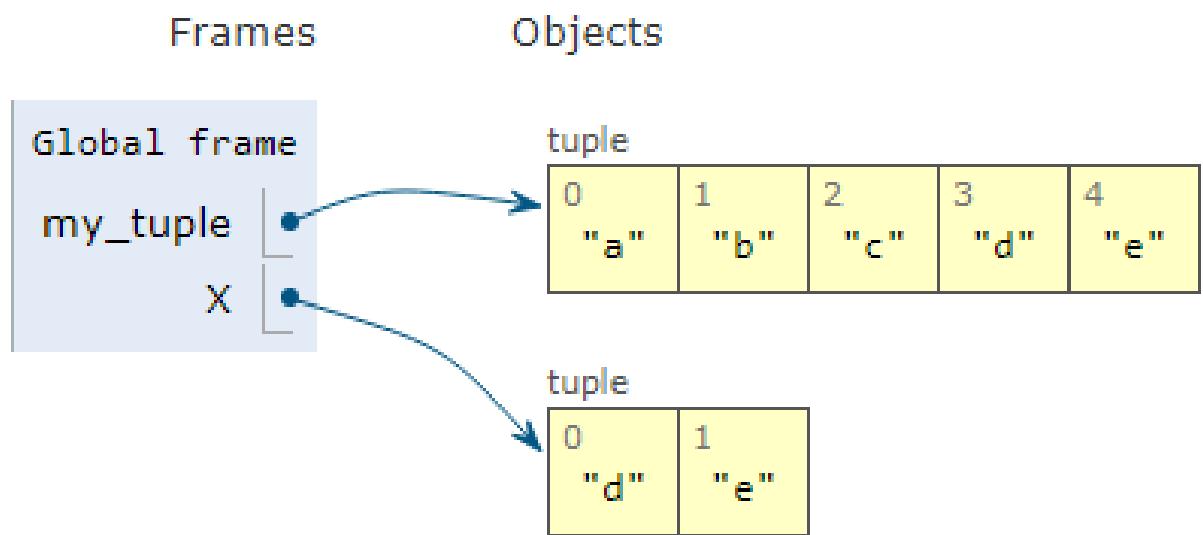


# Tuple Slicing (cont'd)

```
>> 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!!!

The screenshot shows a Jupyter Notebook interface with a 'Console' tab. It has two tabs at the top: 'Python 1' and 'IP: Kernel 1'. The Python tab is active. The code entered is:

```
>>> x = tuple("Mozzarella")
>>> x
('M', 'o', 'z', 'z', 'a', 'r', 'e', 'l', 'l', 'a')
>>> x[2 : 5] = (1, 2, 3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

The output shows the tuple `x` being created from the string "Mozzarella". When attempting to assign a new value to a slice of the tuple (specifically `x[2 : 5]`), a `TypeError` is raised because tuples are immutable.

# 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?

The screenshot shows a Jupyter Notebook interface with a 'Console' tab selected. The Python 2 kernel is active. The user has run the following code:

```
>>>
>>> word = "Mozarella"
>>> word[2:5] = "xyz"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>>
```

The output shows a `TypeError`: 'str' object does not support item assignment.

- but this is legal !

The screenshot shows a Jupyter Notebook interface with a 'Console' tab selected. The Python 2 kernel is active. The user has run the following code:

```
>>>
>>>
>>> word = "Mozarella"
>>> word = word.replace("zar", "xyz")
>>> print(word)
Moxyzella
>>>
```

The output shows the modified string `Moxyzella`.

# 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]
```

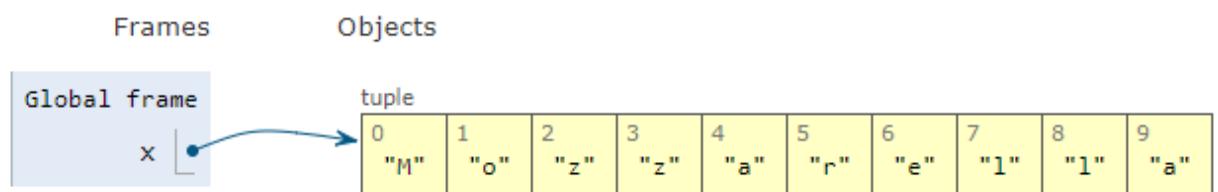
```
>>> w = x[-7 : -3]
```

```
>>> 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

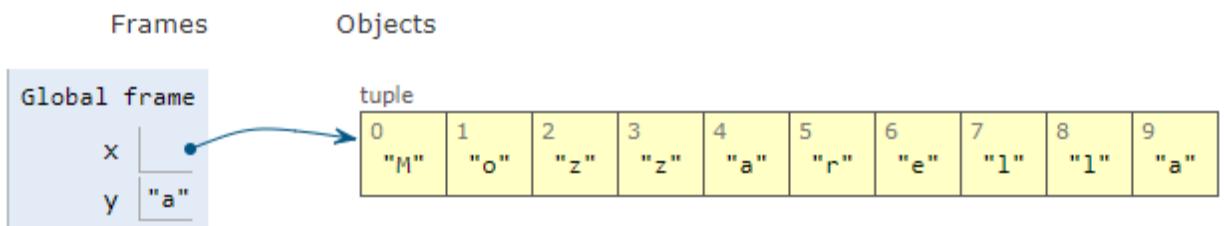
```
>>> x = ('M','o','z','z','a','r','e','l','l',a)
```



# 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		

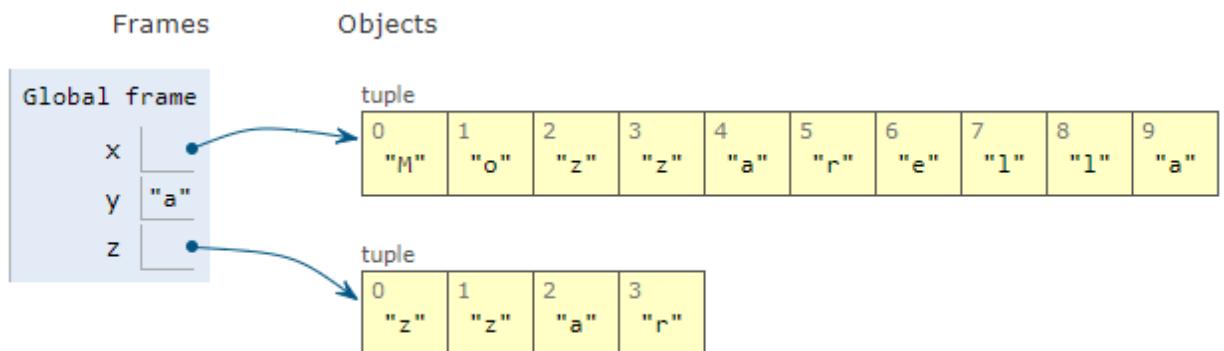
```
>>> x = ('M', 'o', 'z', 'z', 'a', 'r', 'e', 'l', 'l', a)  
>>> y = x[-1]  
>>> a
```



# 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		

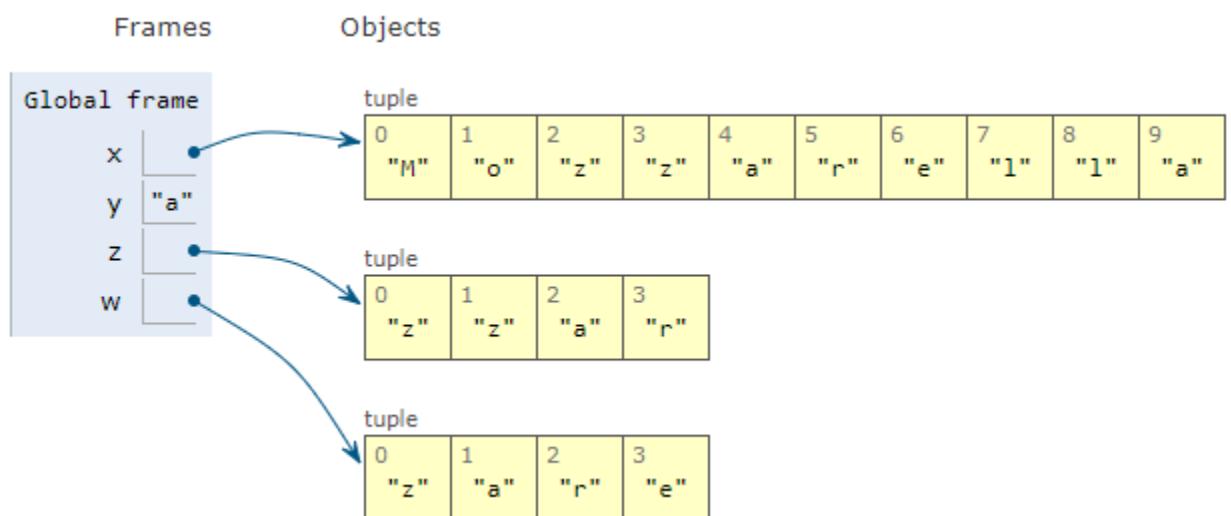
```
>>> x = ('M', 'o', 'z', 'z', 'a', 'r', 'e', 'l', 'l', a)  
>>> y = x[-1]  
>>> z = x[2 : 6]
```



# 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

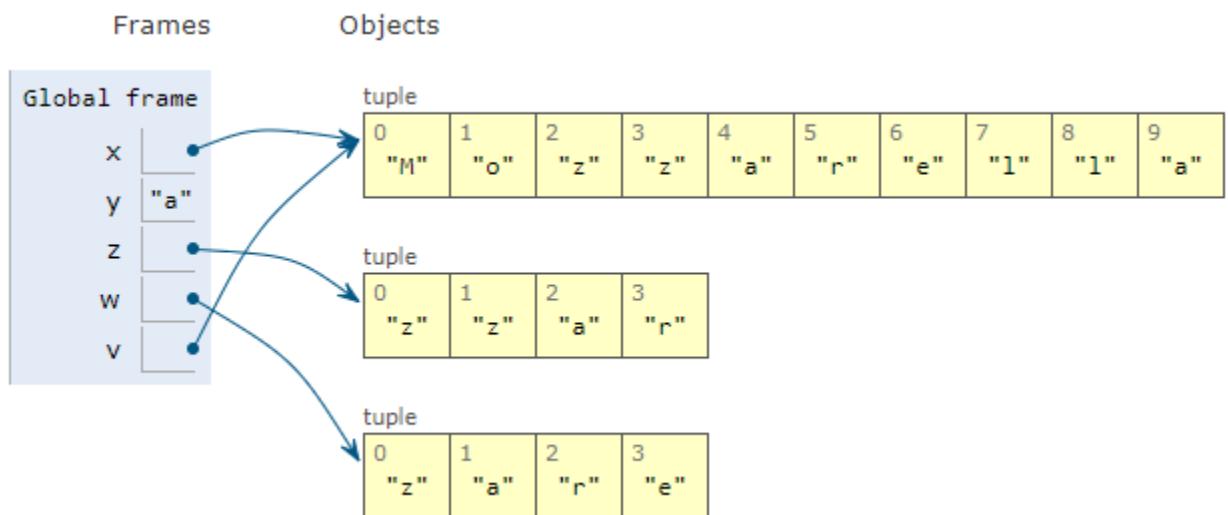
```
>>> x = ('M','o','z','z','a','r','e','l','l',a)  
>>> y = x[-1]  
>>> z = x[2 : 6]  
>>> w = x[-7 : -3]
```



# 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[ : ]
```

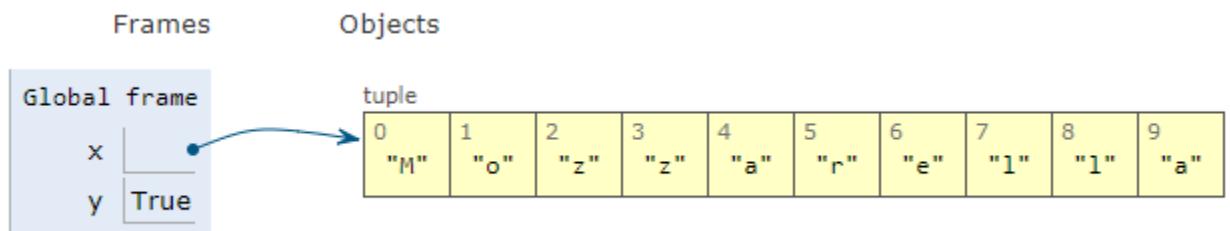


- example of a “shallow” copy

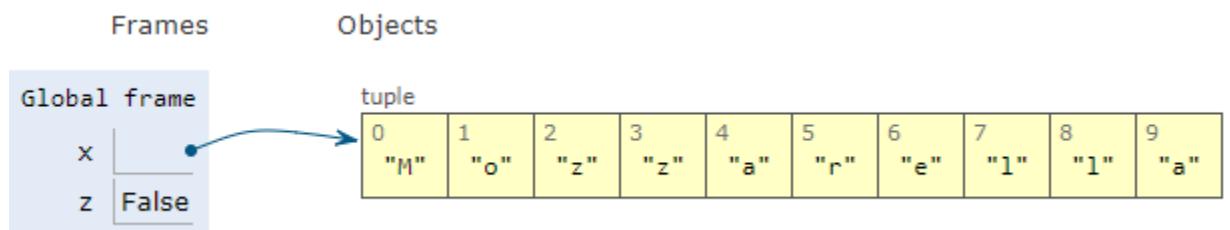
# Tuple Membership: *in*, *not in*

	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

The screenshot shows a Jupyter Notebook interface with a 'Console' tab. In the top bar, there are tabs for 'Python 1' and 'IP: Kernel 1'. The main area displays the following Python code and its output:

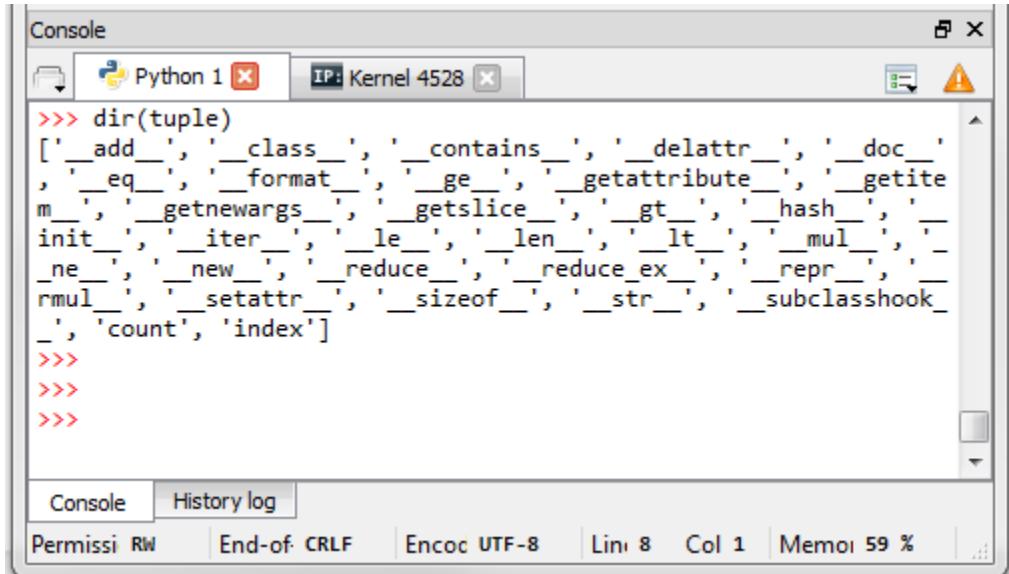
```
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
 '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__',
 '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__',
 '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'co
unt', 'index']
>>>
```

Below the code, the status bar shows: CRLF, Encoding: UTF-8-GUESSED, Line: 5, Column: 1, Memory: 36 %.

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

# *tuple* Type Methods

>>> **dir(tuple)**



```
Console
Python 1 IP: Kernel 4528
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
 '__getnewargs__', '__getslice__', '__gt__', '__hash__',
 '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
 '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__',
 'count', 'index']
>>>
>>>
>>>
```

Console History log

Permissions RW End-of-CRLF Encoding UTF-8 Line 8 Col 1 Memory 59 %

>>> ('Mozzarella', 'Brie').**\_\_len\_\_()**

2

>>> ('Mozzarella', 'Brie').**\_\_sizeof\_\_()**

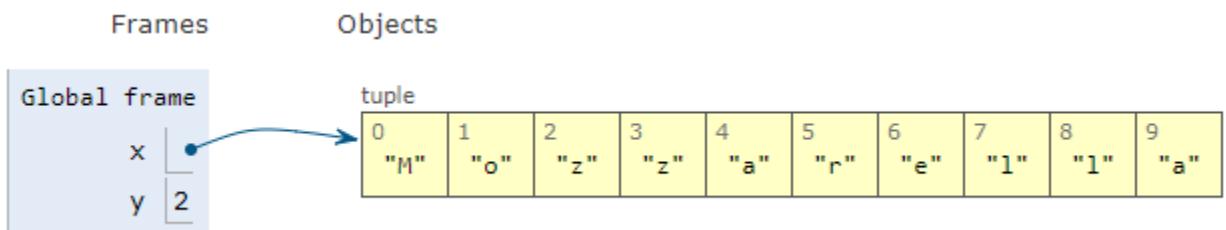
40

# Tuple Methods

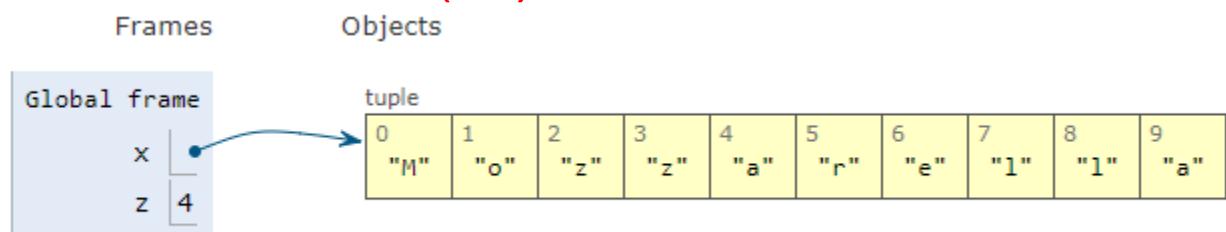
(	0	1	2	3	4	5	6	7	8	9	)
	'M'	'o'	'z'	'z'	'a'	'r'	'e'	'l'	'l'	'a'	
	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	

- 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

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon ('?') and settings icon ('gear') on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located below the tabs. The main content area contains the following text:

Given a variable `t` that is associated with a tuple whose elements are numbers, write some **statements** that use a **while loop** to count the number of times the first element of the tuple appears in the rest of the tuple, and associate that number with the variable `repeats`. Thus if the tuple contains `(1,6,5,7,1,3,4,1)`, then `repeats` would be assigned the value 2 because after the first "1" there are two more "1"s.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and red, and two other icons in question mark and gear shapes. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located to the right of the tabs. The main area contains the following text:

Exercise 51700 –

Write a **statement** that associates t with the **empty tuple**.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon (?) and settings gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text:

Exercise 51701 –

Write a **statement** that associates **t** with a tuple that contains the following elements: **42, 56, 7**.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and orange, and a help icon ('?') and settings icon ('⚙️'). Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located in the top right of the main area. The main content area contains the following text:

Given that `t` has already been defined and refers to a tuple, write an expression whose value is the tuple's length.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon with a gear and a question mark on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text:

Given that `t` refers to a tuple, write a **statement** that assigns the value of its **first element** to `k`.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and orange, and a help icon with a question mark and settings gear. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located in the top right of the main area. The main content area contains the following text:

Given that `k` refers to a non-negative int value and that `t` has been defined and refers to a tuple with at least `k+1` elements, write an expression that evaluates to the `k`th element of `t`.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and gear icon on the right. Below these is the title 'Exercise 51706 —'. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located below the tabs. The main area contains the following text:

Given that `t` has been defined and refers to a tuple write a **statement** that associates `play_list` with a **list** containing the same elements as `t`.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and red, and two other small icons. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located in the top right corner of the main area. The main content area contains the following text:

Given that `play_list` has been defined and refers to a `list`, write a `statement` that associates `t` with a `tuple` containing the same elements as `play_list`.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon (?) and settings icon (gear) on the right. Below these are two tabs: 'WORK AREA' (selected) and 'SOLUTIONS'. A 'Content Support' button is located at the top right of the main content area. The main content area contains the following text:

Given that `t` has been defined and refers to a tuple write some **statements** that associate with `t` a new **tuple** containing the same elements as the original but in sorted order.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**