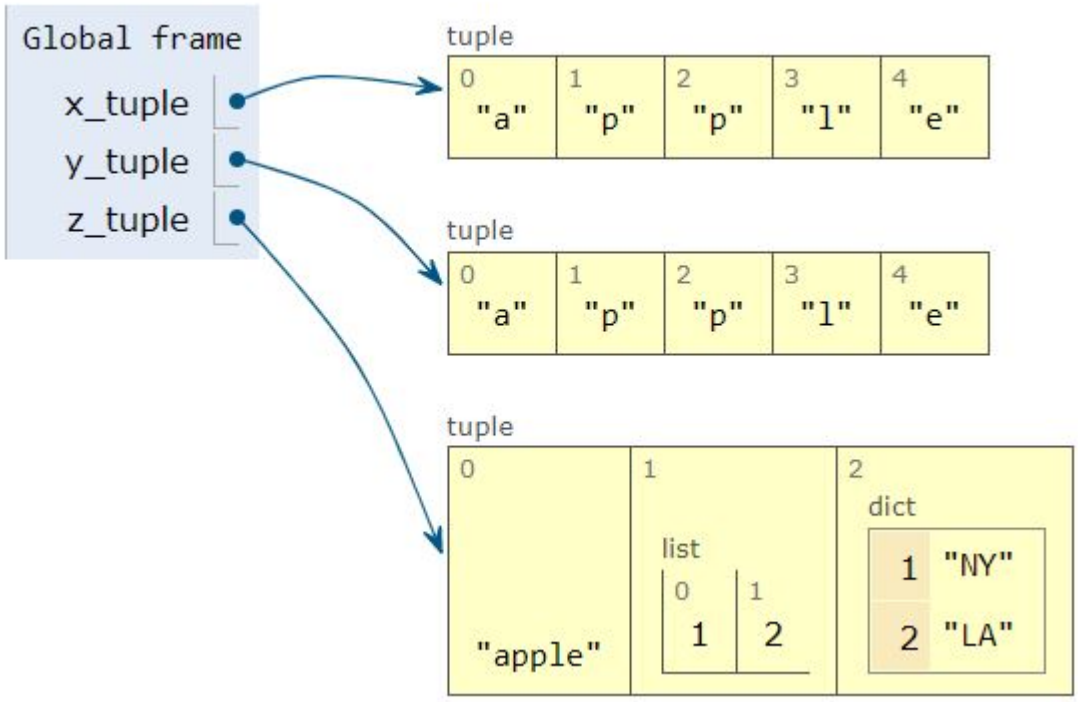
TUPLES

A Python Tuple

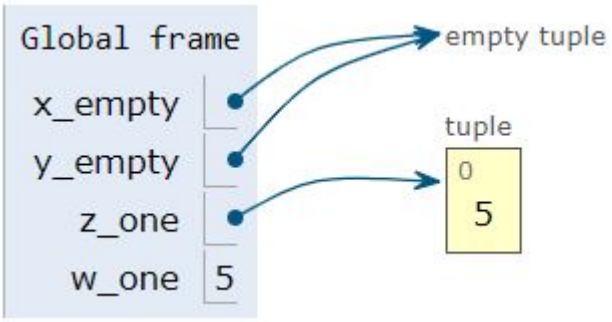
```
x_tuple = ('a','p','p','l','e')
y_tuple = tuple('apple')
z_tuple = ('apple', [1,2], {1:'NY', 2: 'LA'})
```



- immutable, ordered collection
- can contain any objects

Special Cases: Tuples

```
x_empty = tuple()
y_empty = ()          # same, more "pythonic'
z_one = (5, )         # one element tuple
w_one = (5)           # same as w_one = 5
```

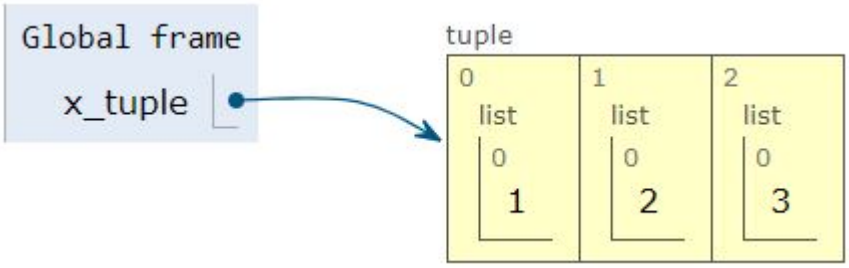


- extra brackets for expressions
- syntax for *sinple* tuples:
  tuple = (element,)

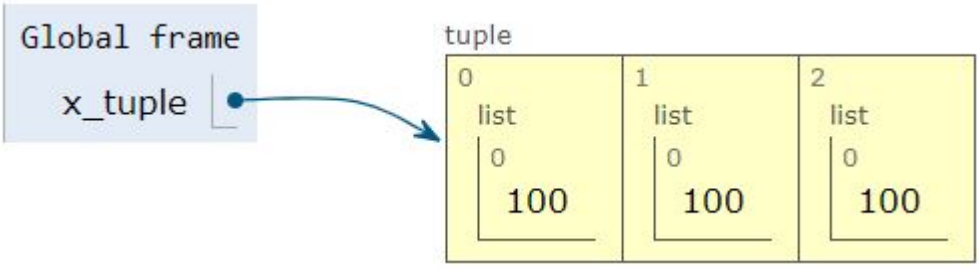
Immutability of Tuples

- consider the following task:

```
x_tuple = ([1], [2], [3])
```



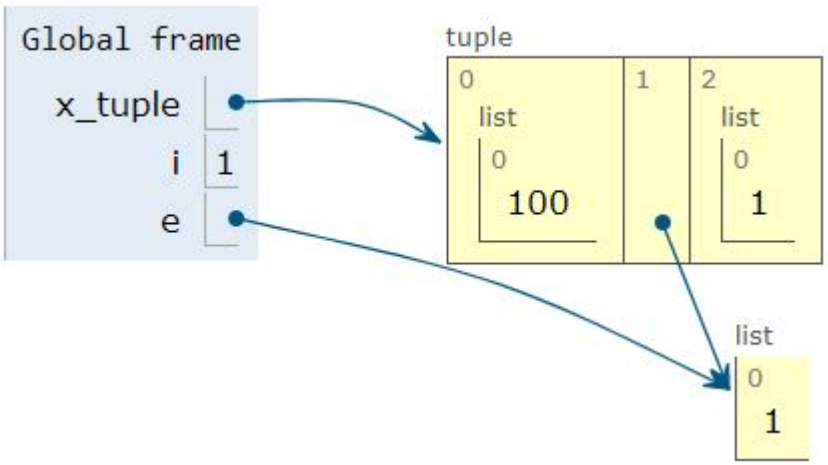
- modify to obtain:



Solution

- can modify mutable elements

```
x_tuple = ([1], [1], [1])
for i, e in enumerate(x_tuple):
    x_tuple[i][0] = 100
```

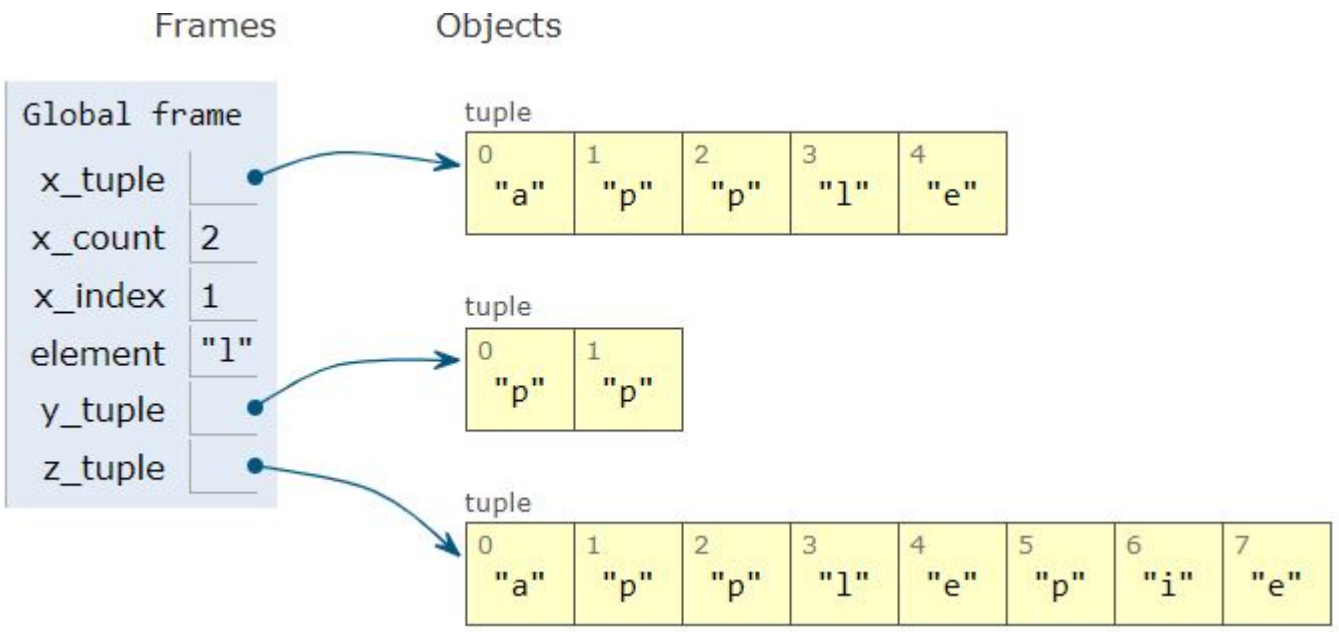


- but cannot replace elements

```
for i, e in enumerate(x_tuple):
    x_tuple[i] = [100]          # illegal
```

Examples of Methods

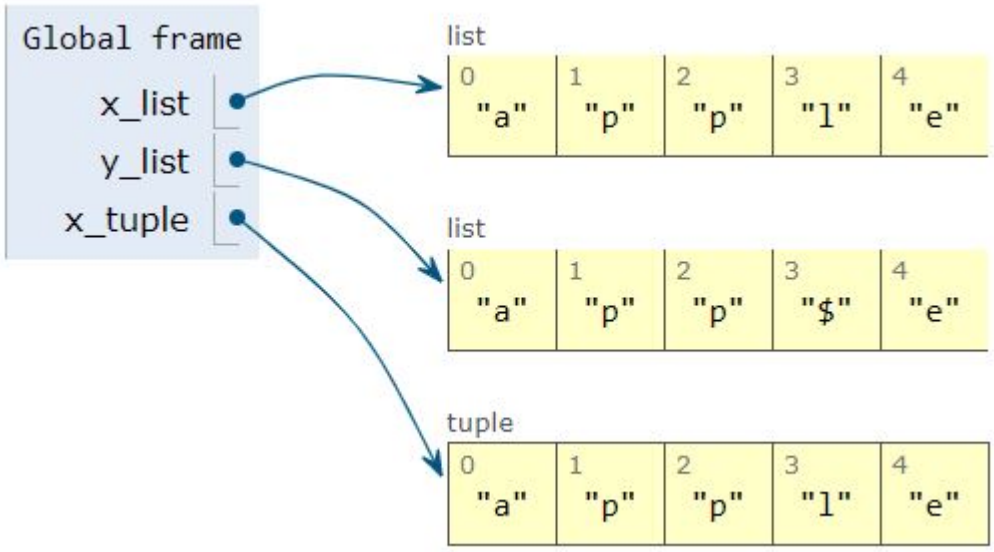
```
x_tuple = ('a','p','p','l','e')
x_count = x_tuple.count('p')
x_index = x_tuple.index('p')
element = x_tuple[3]          # indexing
y_tuple = x_tuple[1 : 3]      # slicing
z_tuple = x_tuple + ('p','i','e')
```



- limited number of methods

Lists vs. Tuples

```
x_list      = ['a','p','p','l','e']
y_list      =  ['a','p','p','l','e']
y_list[3]   = '$'       # can replace
x_tuple     = ('a','p','p','l','e')
x_tuple[3]  = '$'       # TypeError !!!
```



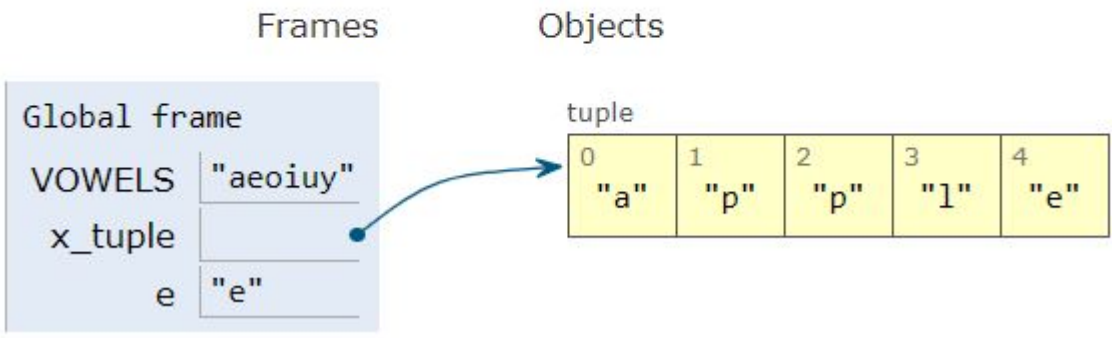
- both are ordered collections
- lists more flexible

Iteration Over Tuples

```
# print vowels in a tuple
VOWELS  = 'aeoiuy'
x_tuple = ('a','p','p','l','e')

for e in x_tuple:
    if e in VOWELS:
        print(e)
```

Print output (drag lower right corner to resize)

```
a
e
```



Exercise(s):

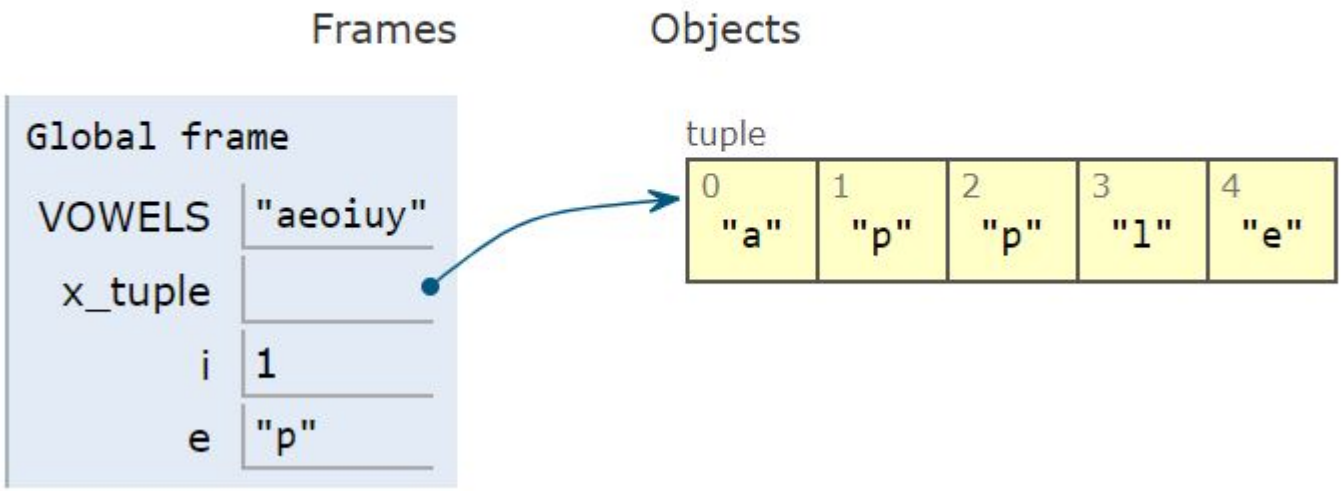
- print consonants in *x_tuple*:

```
x_tuple=tuple("saturday")
```

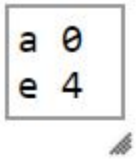
Iteration: *enumerate*()

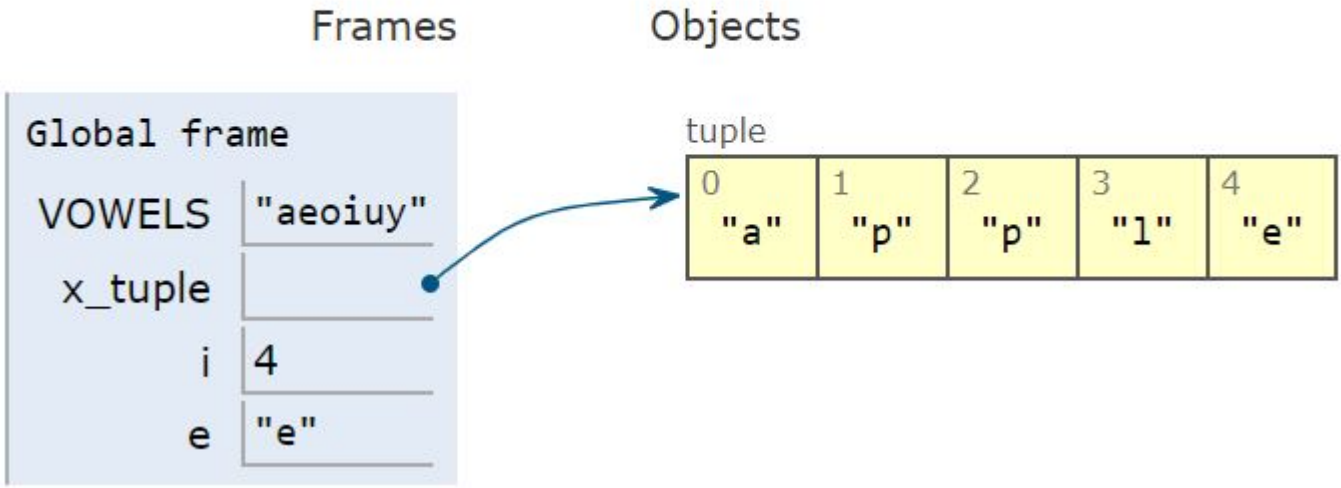
```
# print vowels and positions from tuple
VOWELS   = 'aeoiuy'
x_tuple   = ['a','p','p','l','e']

for i,e in enumerate(x_tuple):
    if e in VOWELS:
        print(e,i)
```

```
a 0
```



Iteration: *enumerate*() (cont'd)





- get both index and element
- use in lists, strings, tuples

Exercise(s):

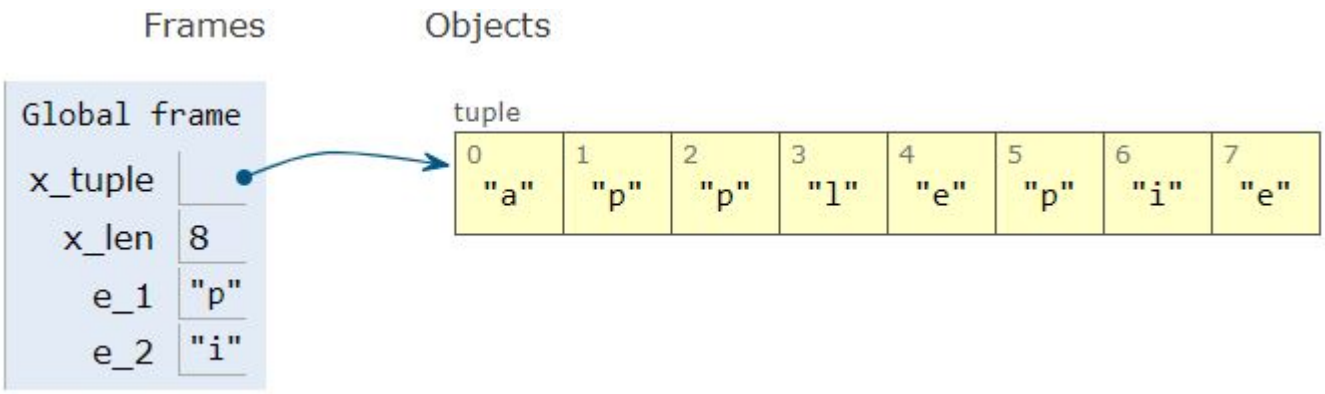
- print consonants and their positions in *x_tuple*:

```
x_tuple=tuple("sunday")
```

- write code without using *enumerate*()

Tuple Indexing

```
x_tuple = ('a','p','p','l','e','p','i','e')
x_len   = len(x_tuple)
e_1     = x_tuple[1]
e_2     = x_tuple[-2]
```

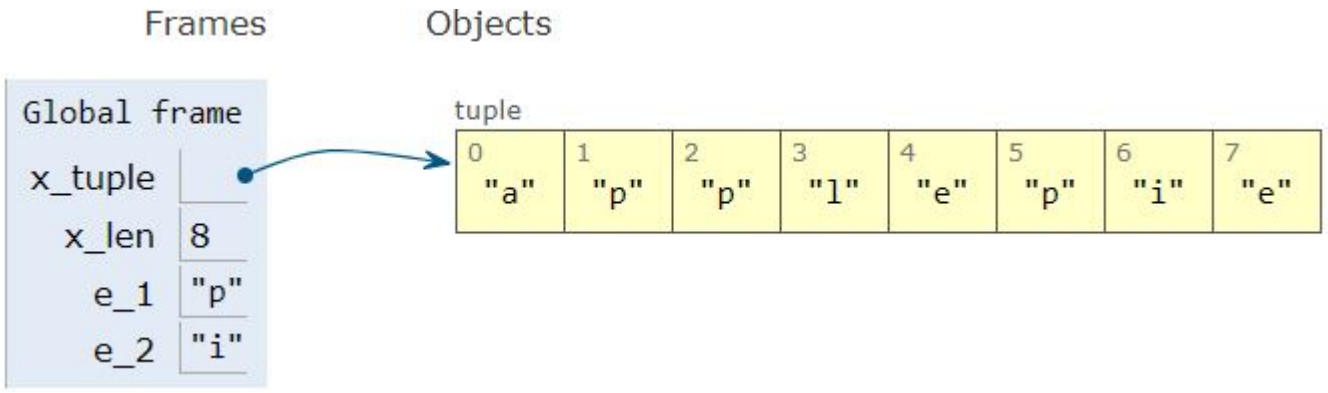


0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

- positive and negative indices

Tuple Indexing (cont'd)

```
x_tuple = ('a','p','p','l','e','p','i','e')
x_len   = len(x_tuple)
e_1     = x_tuple[1]
e_2     = x_tuple[-2]
```

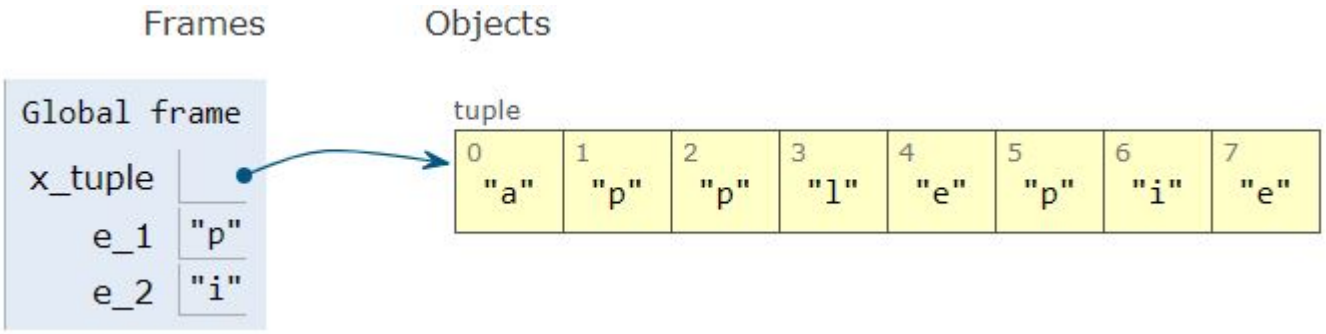


0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

- positive = negative + length

IndexError in Indexing

```
x_tuple = ('a','p','p','l','e','p','i','e')
e_1     = x_tuple[1]      # OK
e_2     = x_tuple[-2]     # OK
e_3     = x_tuple[10]     # IndexError
e_4     = x_tuple[-10]    # IndexError
```



0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

- must handle IndexError

Exercise(s):

- compute elements from *x_tuple*:

```
x_tuple=tuple("tuesday")
```

(a) a = x_tuple[0]

(b) b = x_tuple[1]

(c) c = x_tuple[-1]

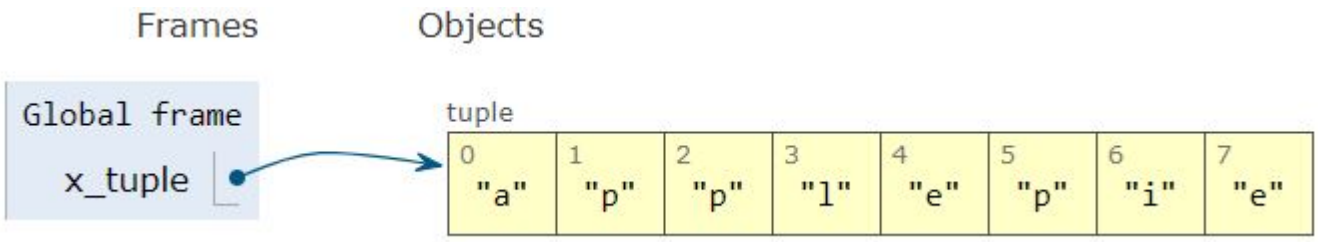
(d) d = x_tuple[5]

(e) e = x_tuple[-5]

(f) f = x_tuple[25]

Tuple Slicing

```
x_tuple = ('a','p','p','l','e','p','i','e')
```



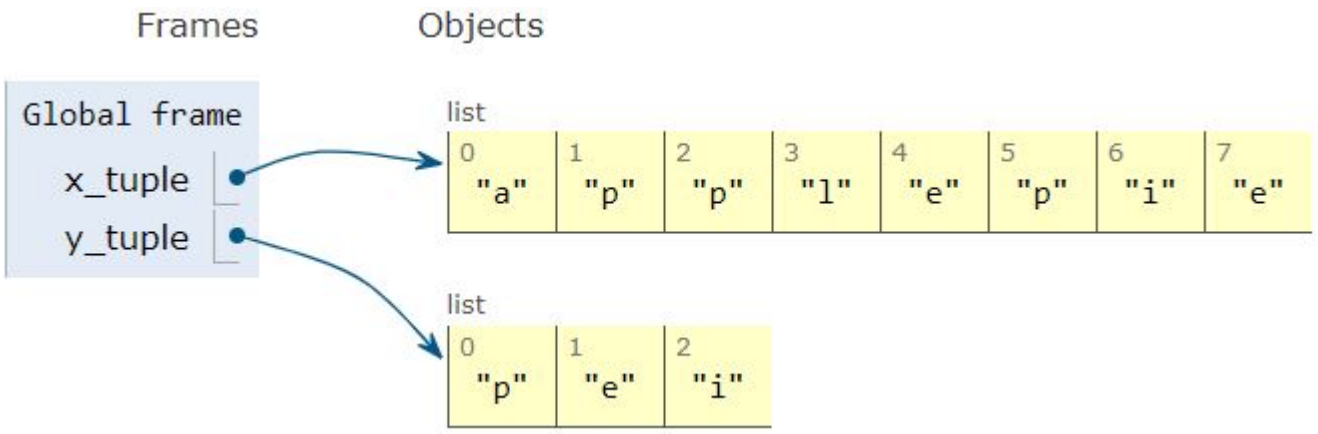
0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

[start : end + 1 : step]

- use both pos & neg indices
- negative step for reversals

Tuple Slicing (cont'd)

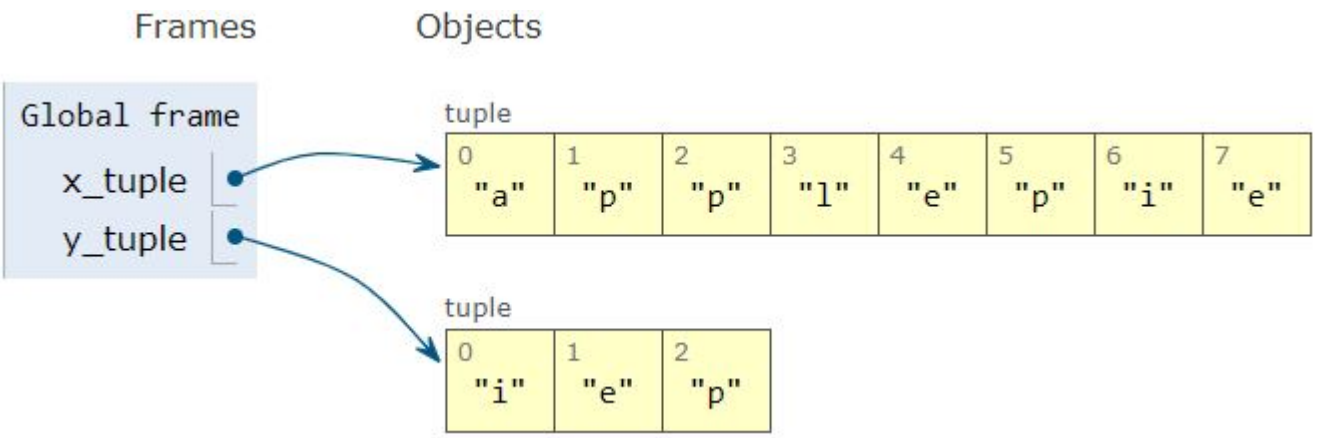
```
x_tuple = ('a','p','p','l','e','p','i','e')

y_tuple = x_tuple[ 2 :  7 : 2]
y_tuple = x_tuple[-6 : -1 : 2]
y_tuple = x_tuple[ 2 : -1 : 2]
y_tuple = x_tuple[-6 :  7 : 2]
```



0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

Tuple Slicing (cont'd)

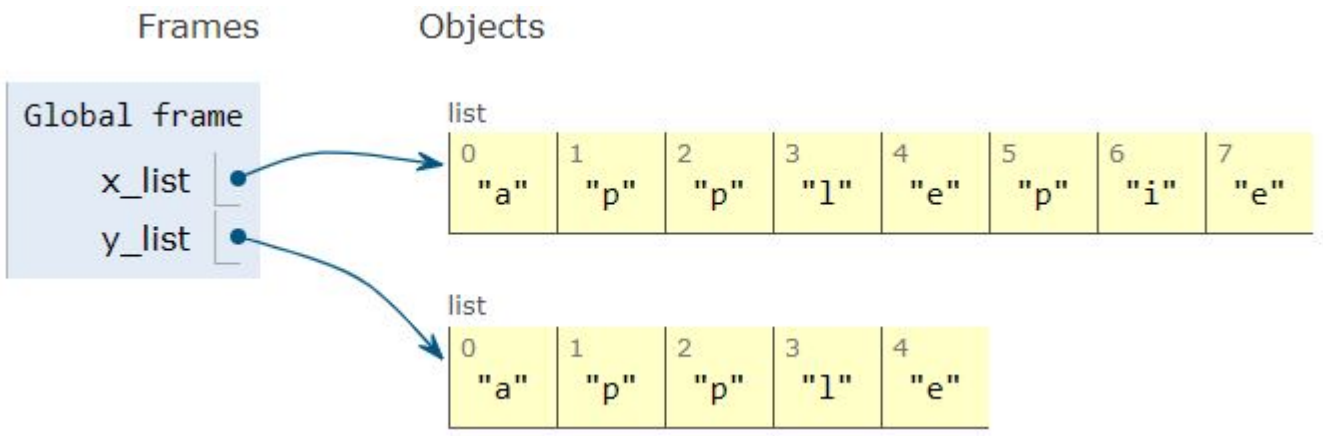
```
x_tuple = ('a','p','p','l','e','p','i','e')
y_tuple = x_tuple[ 6 :  1 : -2] # in reverse
y_tuple = x_tuple[-2 : -7 : -2]
y_tuple = x_tuple[ 6 : -7 : -2]
y_tuple = x_tuple[-2 :  1 : -2]
```



0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

Tuple Slicing (cont'd)

```
x_tuple = ('a','p','p','l','e','p','i','e')

y_tuple = x_tuple[ 0 : 5 : 1 ]
y_tuple = x_tuple[   : 5 : 1 ]  # assume defaults
y_tuple = x_tuple[   : 5 ]
```



0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

Exercise(s):

- compute slices from *x_tuple*:

```
x_tuple=tuple("wednesday")
```

(a) a = x_tuple[0 : 10 : 2]

(b) b = x_tuple[1 : 9 : 3]

(c) c = x_tuple[-10 : -2 : 3]

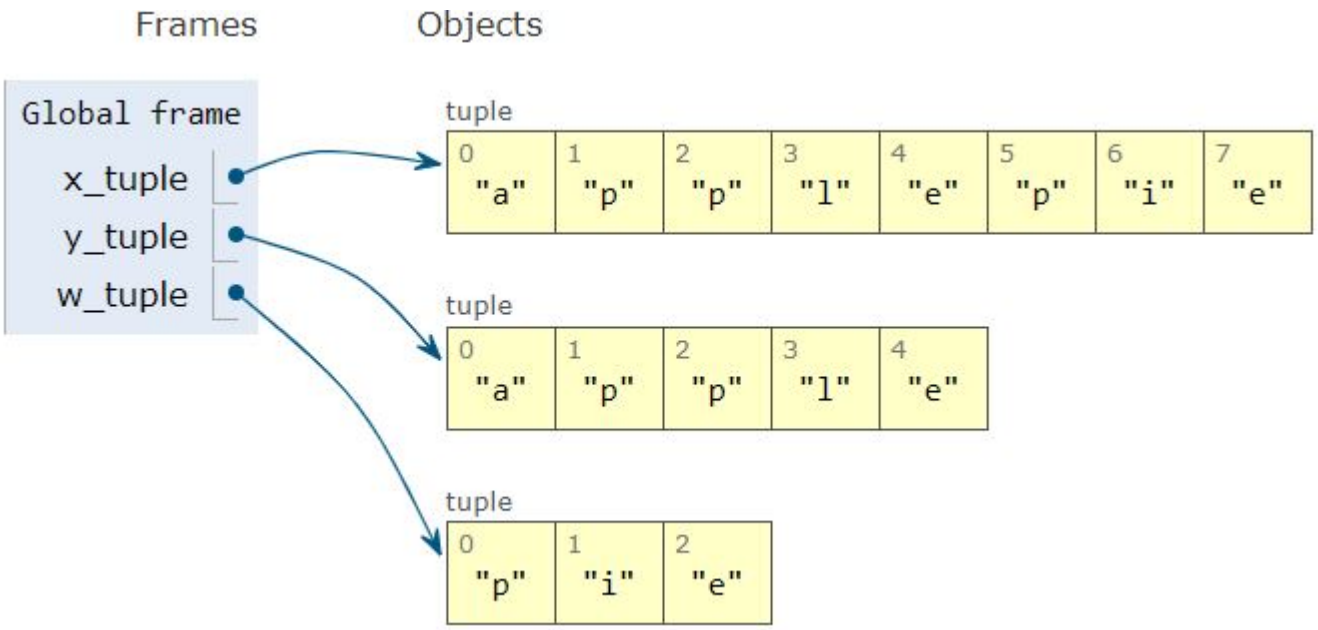
(d) d = x_tuple[0 : -1 : 1]

(e) e = x_tuple[0 : -2 : 3]

(f) f = x_tuple[30 : 5 : 5]

Slicing with Defaults

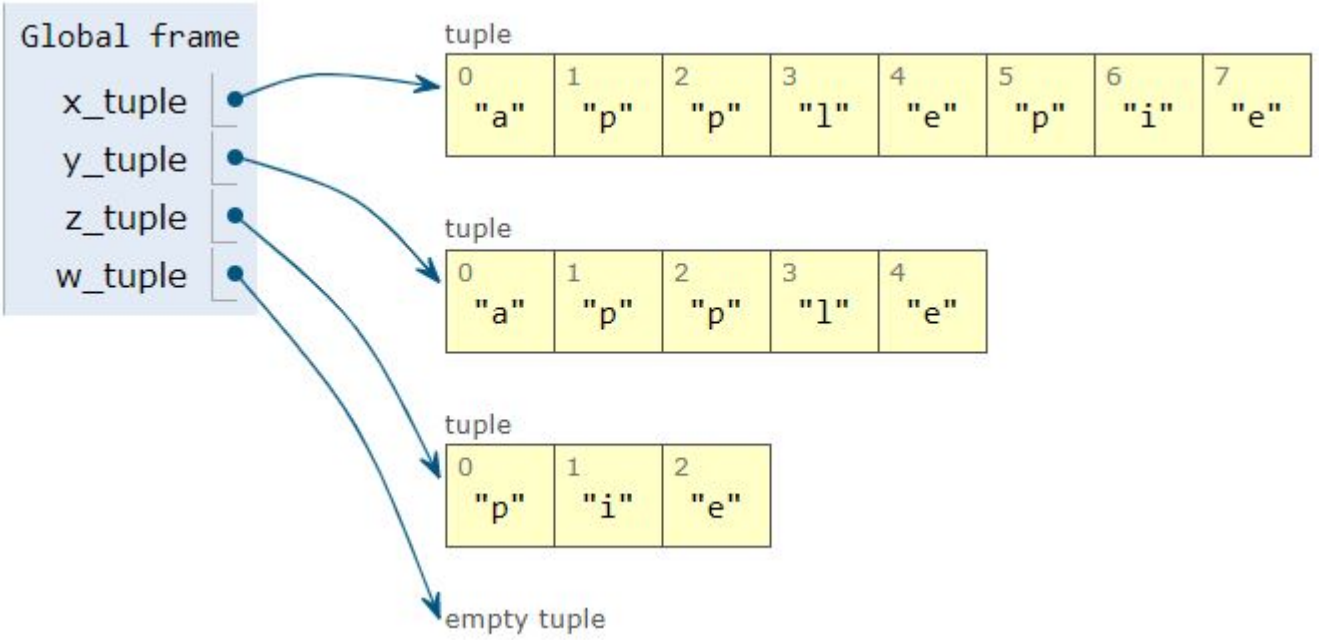
```
x_tuple = ('a','p','p','l','e','p','i','e')
y_tuple = x_tuple[   : 5 ]
w_tuple = x_tuple[ 5 :   ]
```



0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

"Out-of-Bound" Slicing

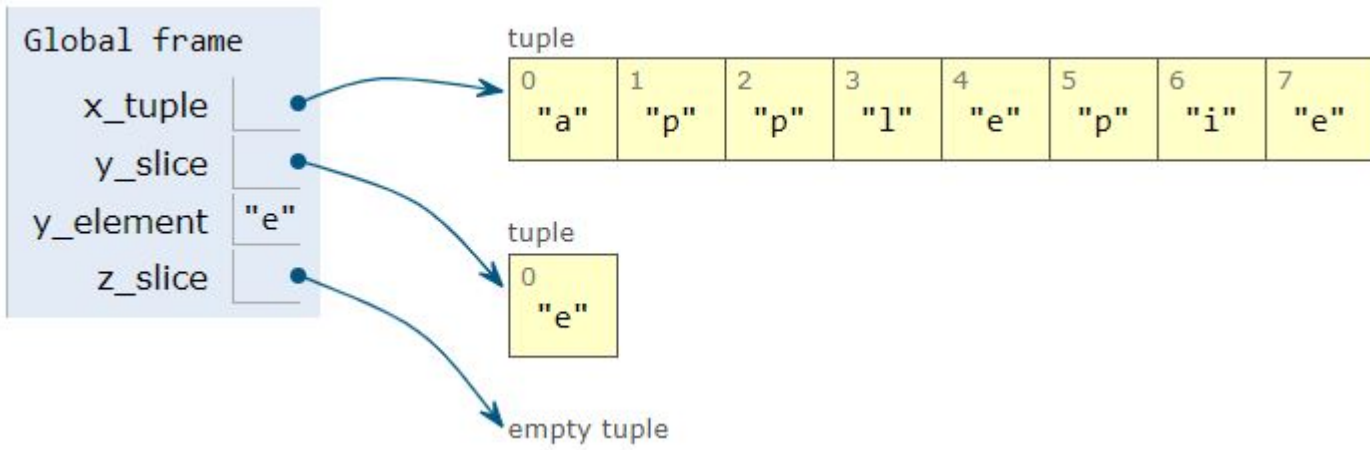
```
x_tuple = ('a','p','p','l','e','p','i','e')
y_tuple = x_tuple[ -100 : 5   ]
z_tuple = x_tuple[    5 : 500 ]
w_tuple = x_tuple[  400 : 500 ]
```



- "largest" sub-list
- no error!

Slicing vs. Indexing

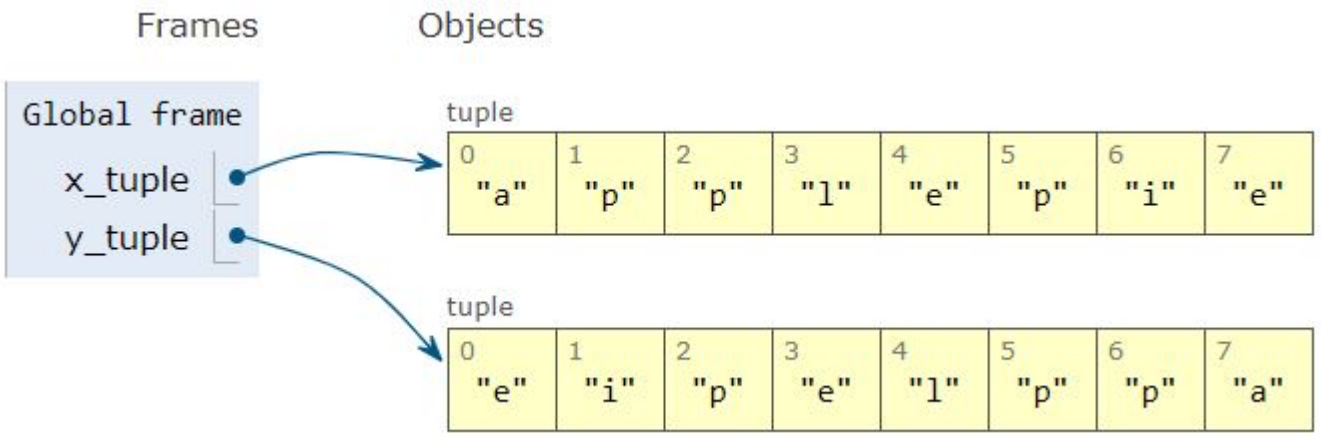
```
x_tuple   = ('a','p','p','l','e','p','i','e')
y_slice   = x_tuple[   4 : 5  ]
y_element = x_tuple[   4 ]
z_slice   = x_tuple[ 100 : 101]
z_element = x_tuple[ 100 ]      # error
```



0	1	2	3	4	5	6	7
a	p	p	l	e	p	i	e
−8	−7	−6	−5	−4	−3	−2	−1

Tuple Reversal

```
x_tuple = ('a','p','p','l','e','p','i','e')
y_tuple = x_tuple[ : : -1 ] # new object
```



- no "in-place" reversal
- tuples are immutable !!!

Exercise(s):

- compute slices from *x_tuple*:

```
x_tuplet=tuple("thursday")
```

(a) a = x_tuple[10 : 0 : -1]

(b) b = x_tuple[10 : : -2]

(c) c = x_tuple[: : -2]

(d) d = x_tuple[: : -3]

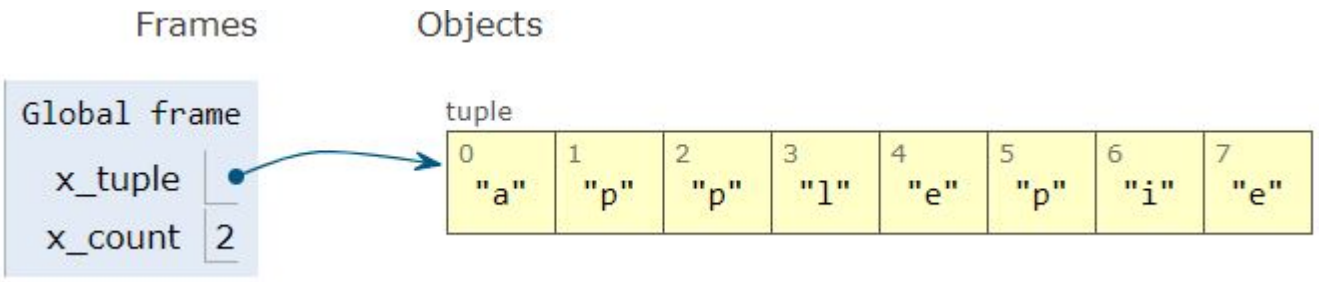
(e) e = x_tuple[: : -4]

(f) f = x_tuple[0 : -1 : -1]

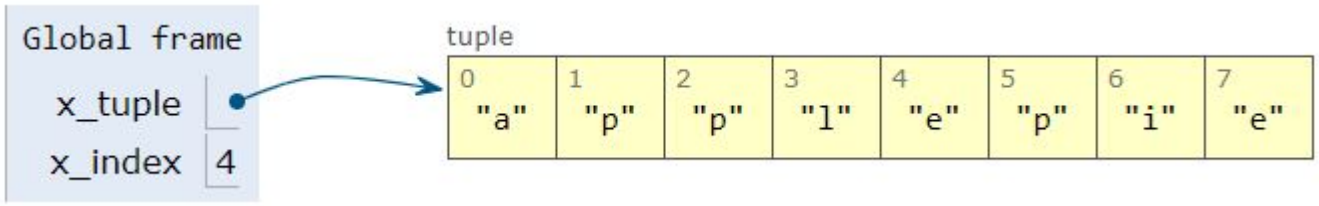
Tuple Methods

- methods: *count* (), *index*()

```
x_tuple = ('a','p','p','l','e','p','i','e')
x_count = x_tuple.count('e') # count 'e'
```




```
x_tuple = ('a','p','p','l','e','p','i','e')
x_index = x_tuple.index('e')  # index first 'e'
```



Exercise(s):

- count consonants in *x_tuple*:

```
x_tuple=tuple("friday")
```

- print indices for first occurence of consonants in *y_tuple*:

```
y_tuple=tuple("March")
```

Summary:

- ordered collection
- supports indexing & slicing
- immutable
- can contain mutable elements
- methods: *count*() & *index*()
- more restricted than lists
- but more memory efficient