Module 4

This is a single, concatenated file, suitable for printing or saving as a PDF for offline viewing. Please note that some animations or images may not work.

Module 4 Study Guide and Deliverables

Theme: Collections in Detail

Readings:
- Chapter 7 and Chapter 9
- Module Lecture Notes

Topics: Sets, Tuples, Dictionaries, Stacks, Queues, Singly Linked Lists, Doubly Linked Lists, Sorting, Searching

Assignments Assignment 4 due on Tuesday, April 13 at 6:00 PM ET

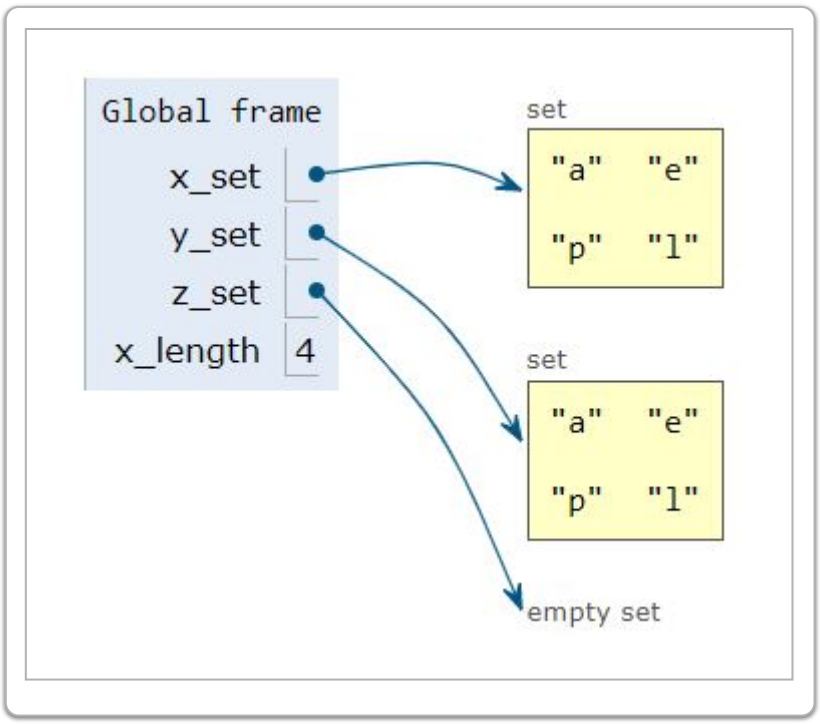
Assessments Quiz 4:
- Available Friday, April 9 at 6:00 AM ET
- Due on Tuesday, April 13 at 6:00 PM ET

Live Classrooms:
- Tuesday, April 6, 8:00 - 9:30 PM ET
- Thursday, April 8, 6:00 - 7:30 PM ET
- Facilitator Session: Friday, April 9, at 8:00 PM ET

Sets

A Python Set

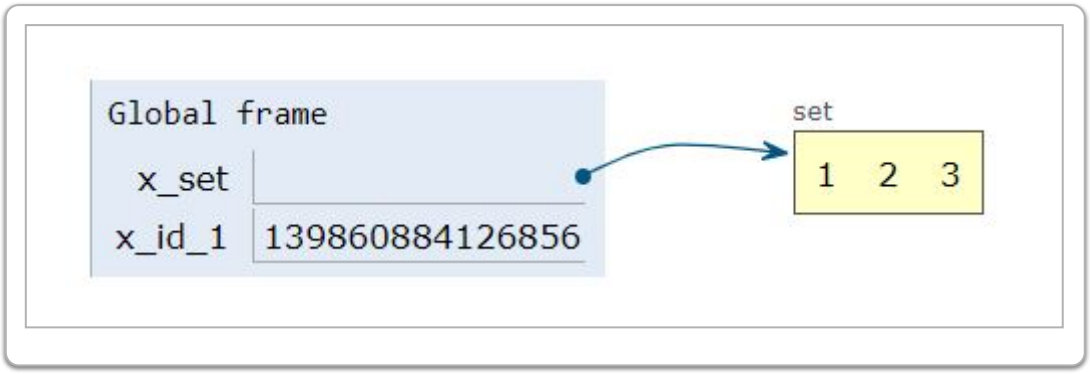
```
x_set = {'a','p','p','l','e'}
y_set = set('apple ')
z_set = set()
x_length = len(x_set)
```



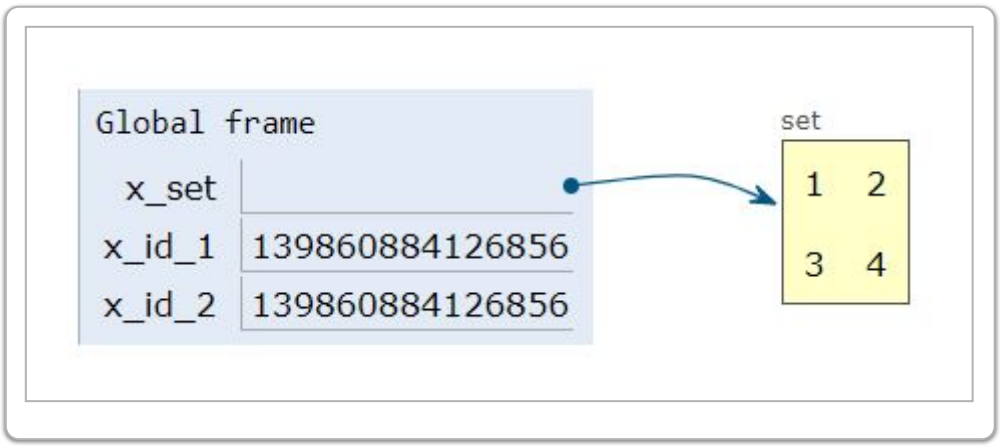
- un-ordered & mutable
- unique hashable elements

Sets and Mutability

```
x_set = {1, 3, 3, 2}
x_id_1 = id(x_set)
```



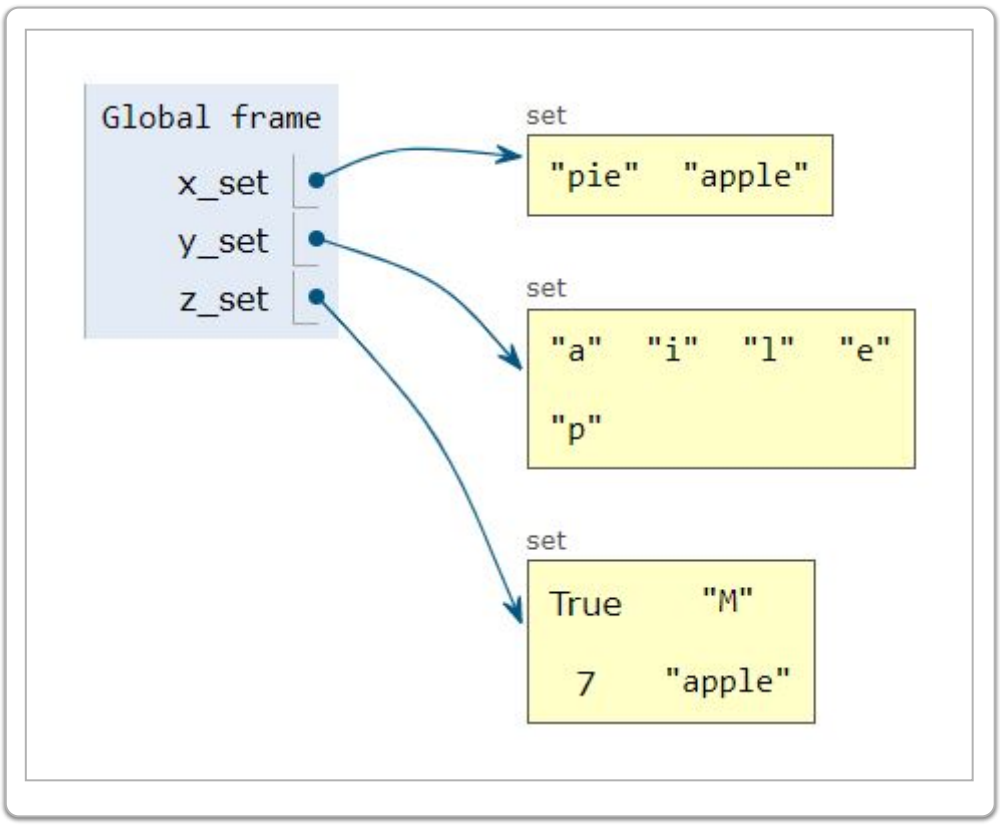
```
x_set.add(2) # duplicate: not added
x_set.add(4) # new element: added
x_id_2 = id(x_set)
```



- sets are mutable

Sets from Primitive Types

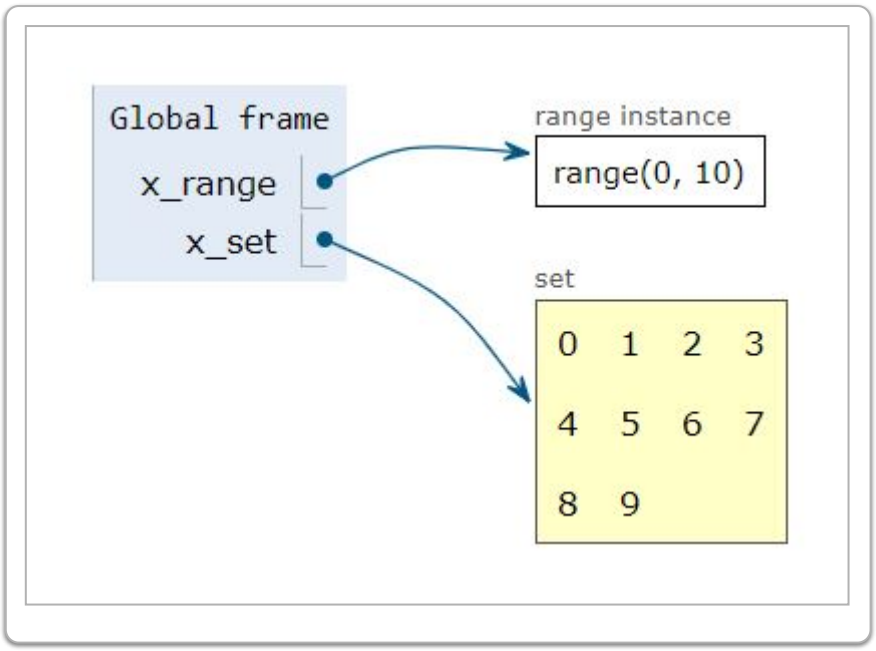
```
x_set = {'apple', 'pie'}
y_set = set('applepie')
z_set = {'apple', 7, True, 'M', 2+3j}
```



- all primitive types are hashable!

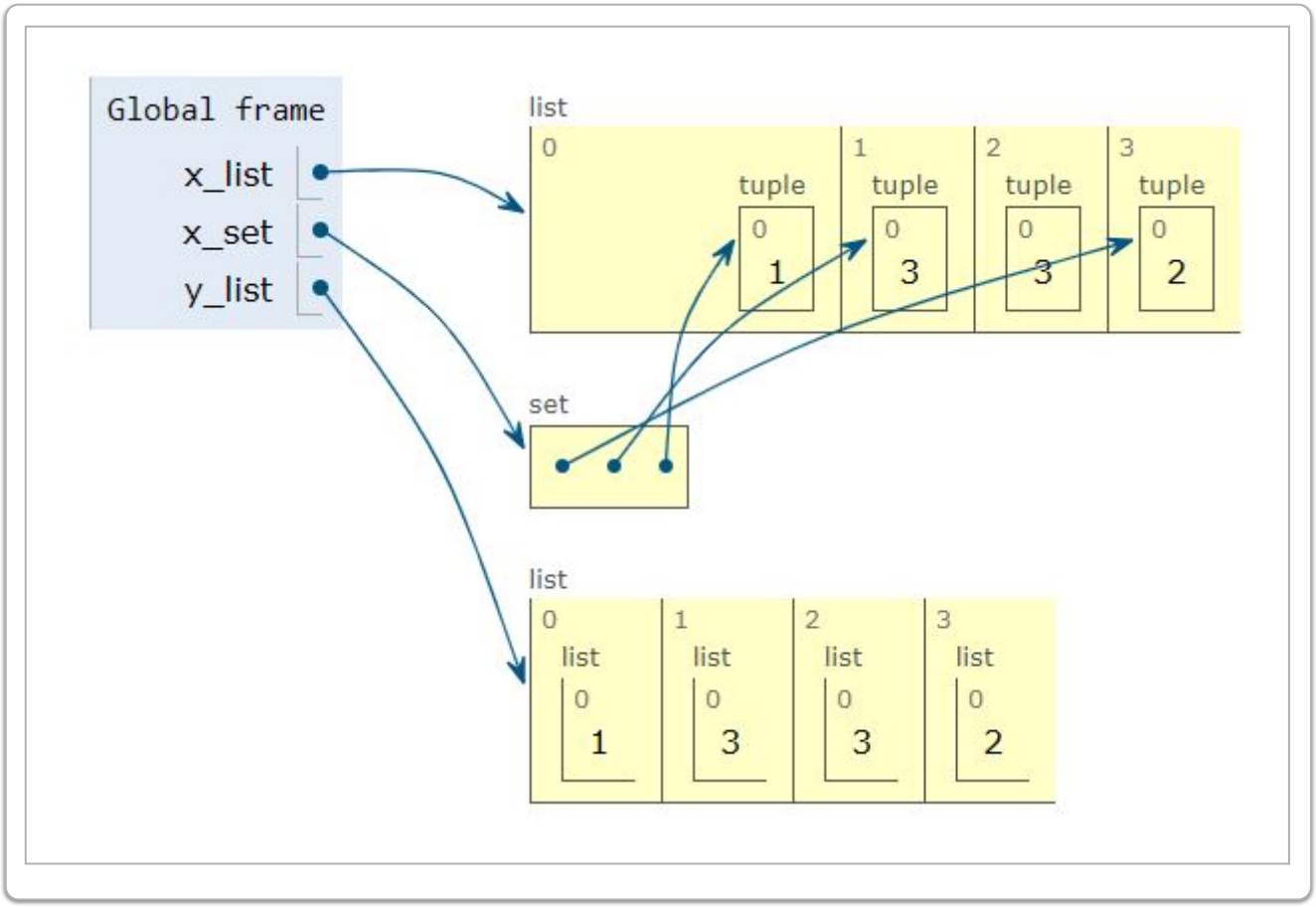
Sets from Ranges

```
x_set = set(range(10))
```



Sets from Lists

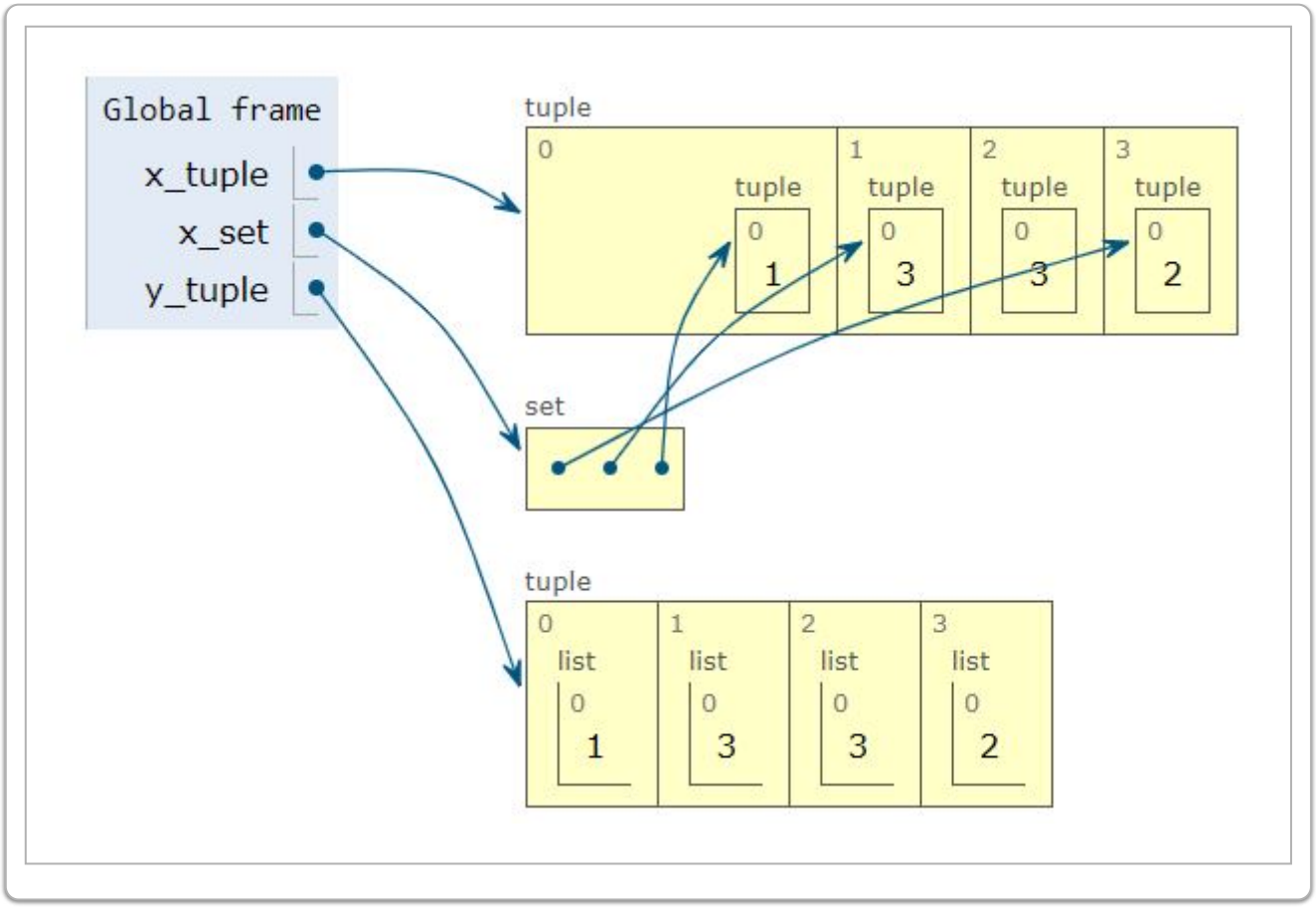
```
x_list = [ (1,), (3,), (3,), (2,)]
x_set = set(x_list)
y_list = [ [1], [3], [3], [2] ]
y_set = set(y_list) # illegal (unhashable)
```



- hashable elements only!

Sets from Tuples

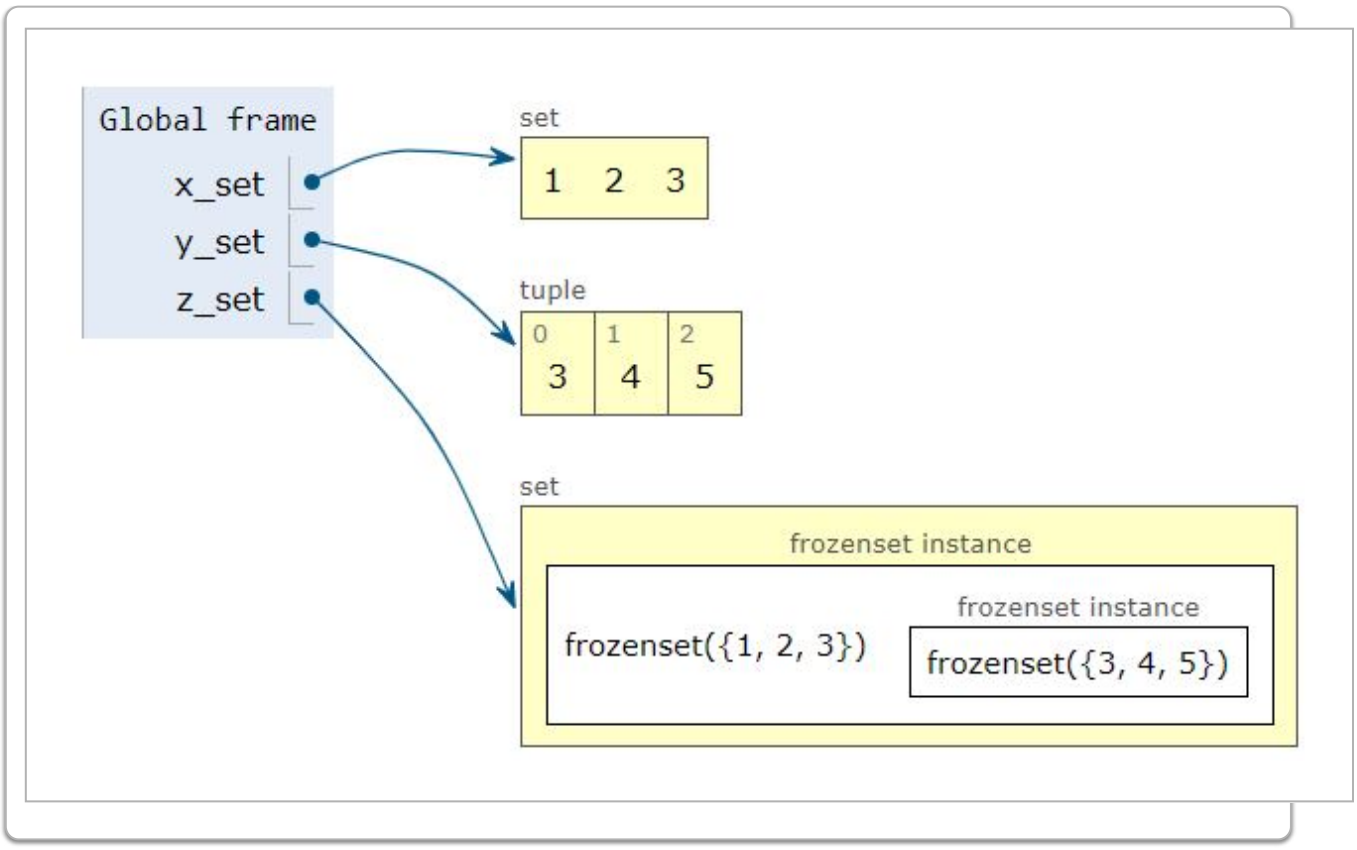
```
x_tuple = ( (1,), (3,), (3,), (2,))
x_set = set(x_tuple)
y_tuple = ( [1], [3], [3], [2] ) # illegal
```



- hashable elements only!

Sets from Frozen Sets

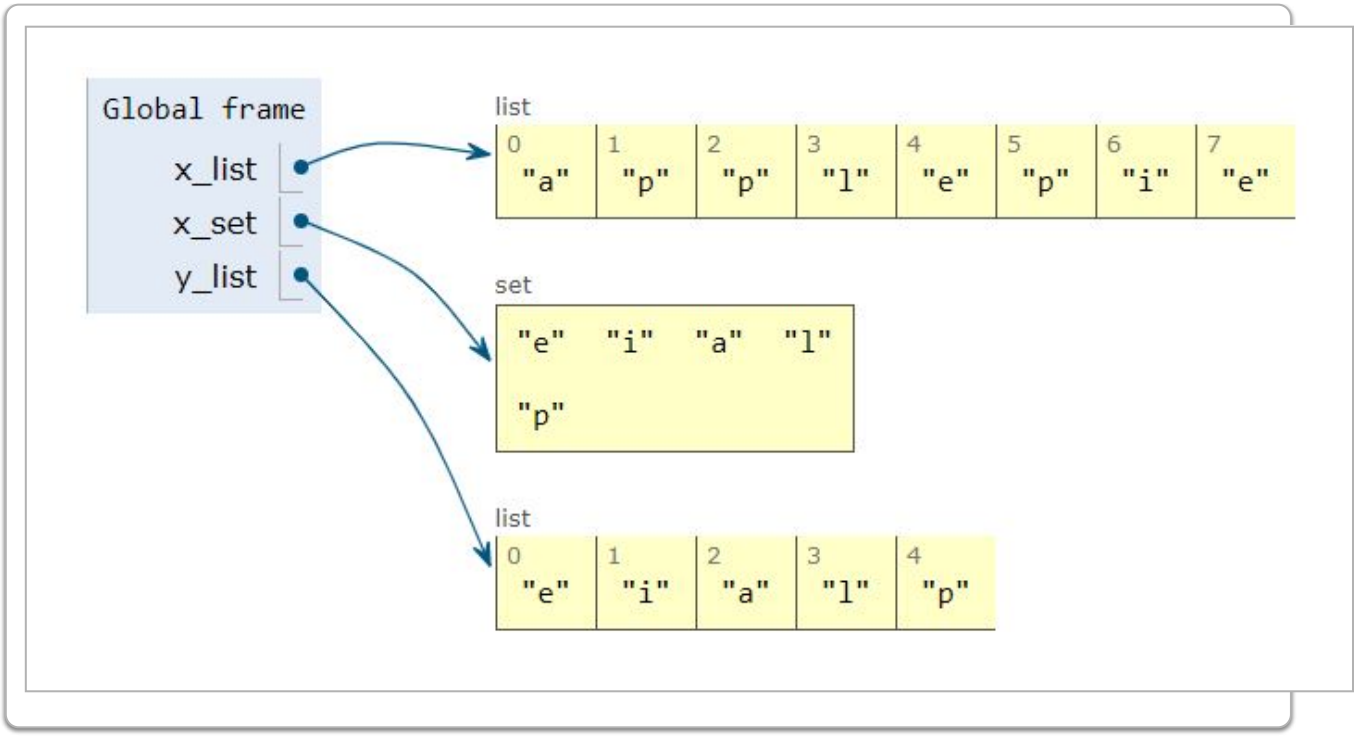
```
x_set = {1 ,2 ,3}
y_set = (3 ,4 ,5)

z_set = { frozenset(x_set), frozenset(y_set) }
w_set = { x_set, y_set } # illegal
```



- make sets immutable ('frozen')

Example: Remove Duplicates

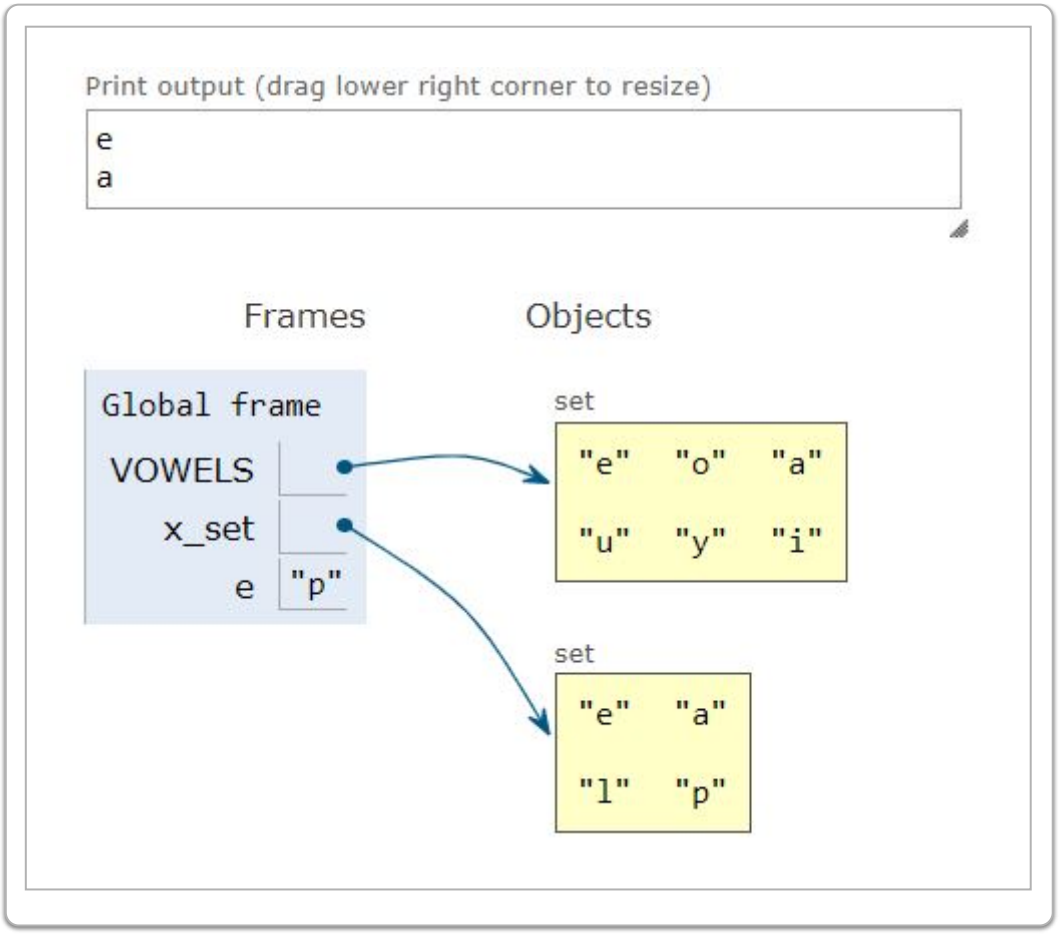
```
x_list = list('applepie')
x_set = set(x_list)
y_list = list(x_set)
```



- no guarantee for ordering

Membership and Iteration

```
VOWELS = set('aeoiuy')
x_set = {'a','p','p','l','e'}
for e in x_set:
    if e in VOWELS:
        print(e)
```



- iterable but not indexed

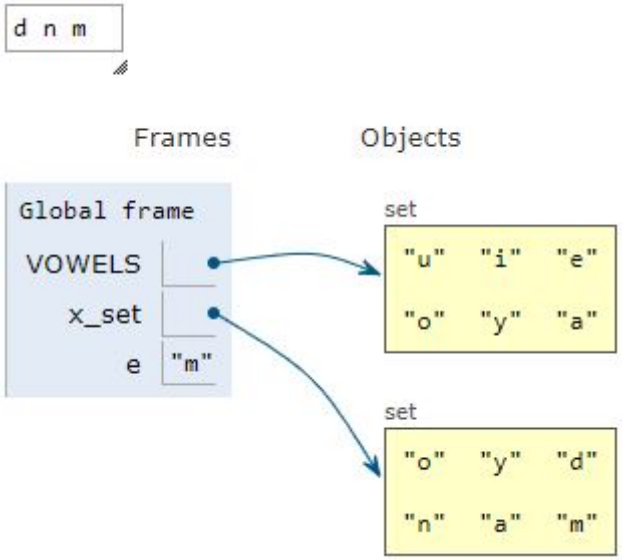
Test Yourself: 4.1.01

Print consonants in *x_set*:

```
x_set = set("monday")
```

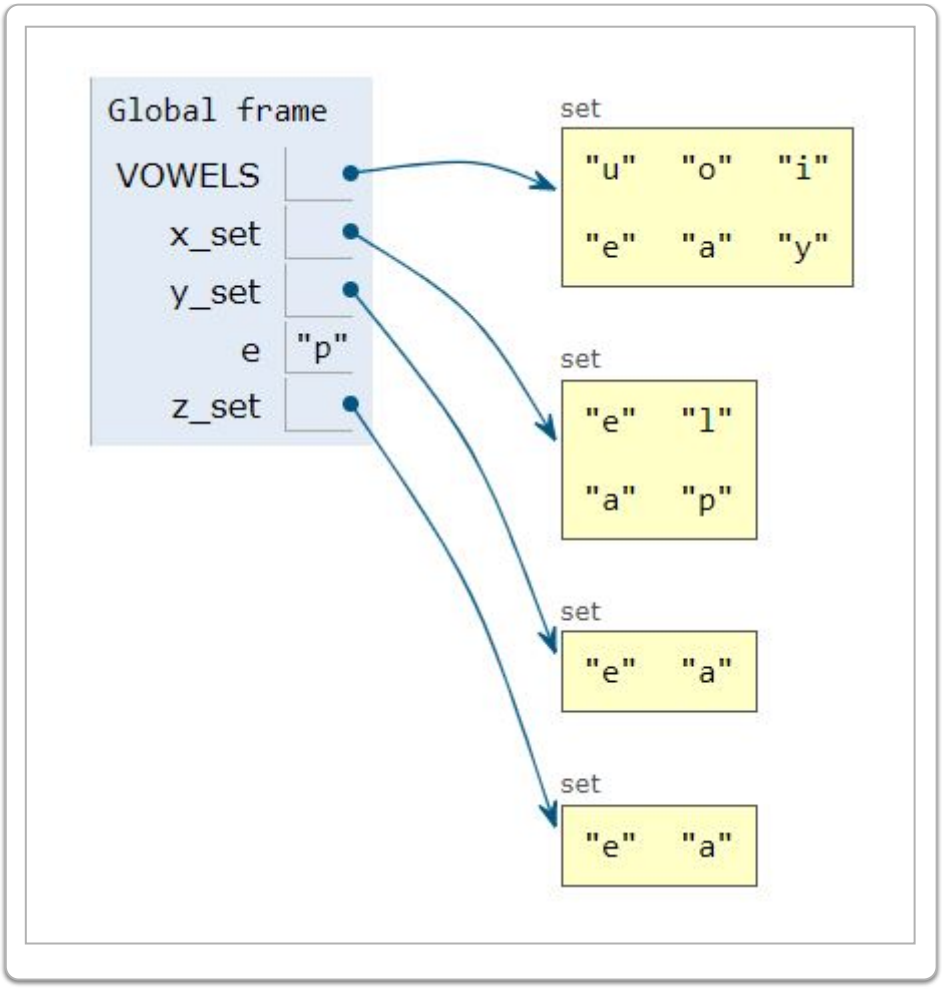
Solution:

```
VOWELS = set("aeoiuy")
x_set = set("monday")
for e in x_set:
    if e not in VOWELS:
        print(e, end = " ")
```



Set *comprehension*

```
VOWELS = set('aeiouy')
x_set = {'a','p','p','l','e'}
y_set = set()
for e in x_set:
    if e in VOWELS:
            y_set.add(e)
z_set = { e for e in x_set if e in VOWELS }
```



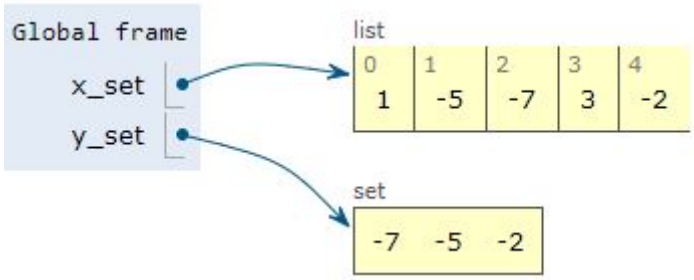
Test Yourself: 4.1.02

Use set comprehension to construct *y_set* with negative elements from *x_set*:

```
x_set = [1, -5, -7, 3, -2]
y_set = [-5, -7, -2]
```

Solution:

```
x_set = [1, -5, -7, 3, -2]
y_set = {e for e in x_set if e < 0}
```



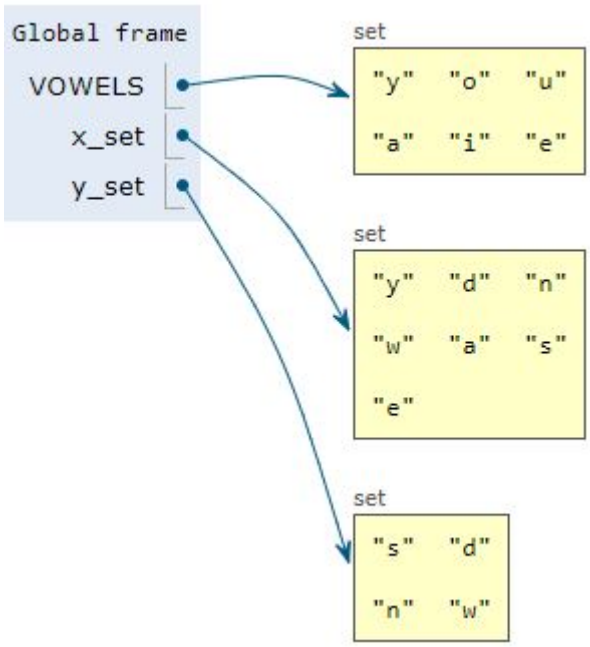
Test Yourself: 4.1.03

Use set comprehension to construct a list of consonants in *x_set*:

```
x_set = set("wednesday")
```

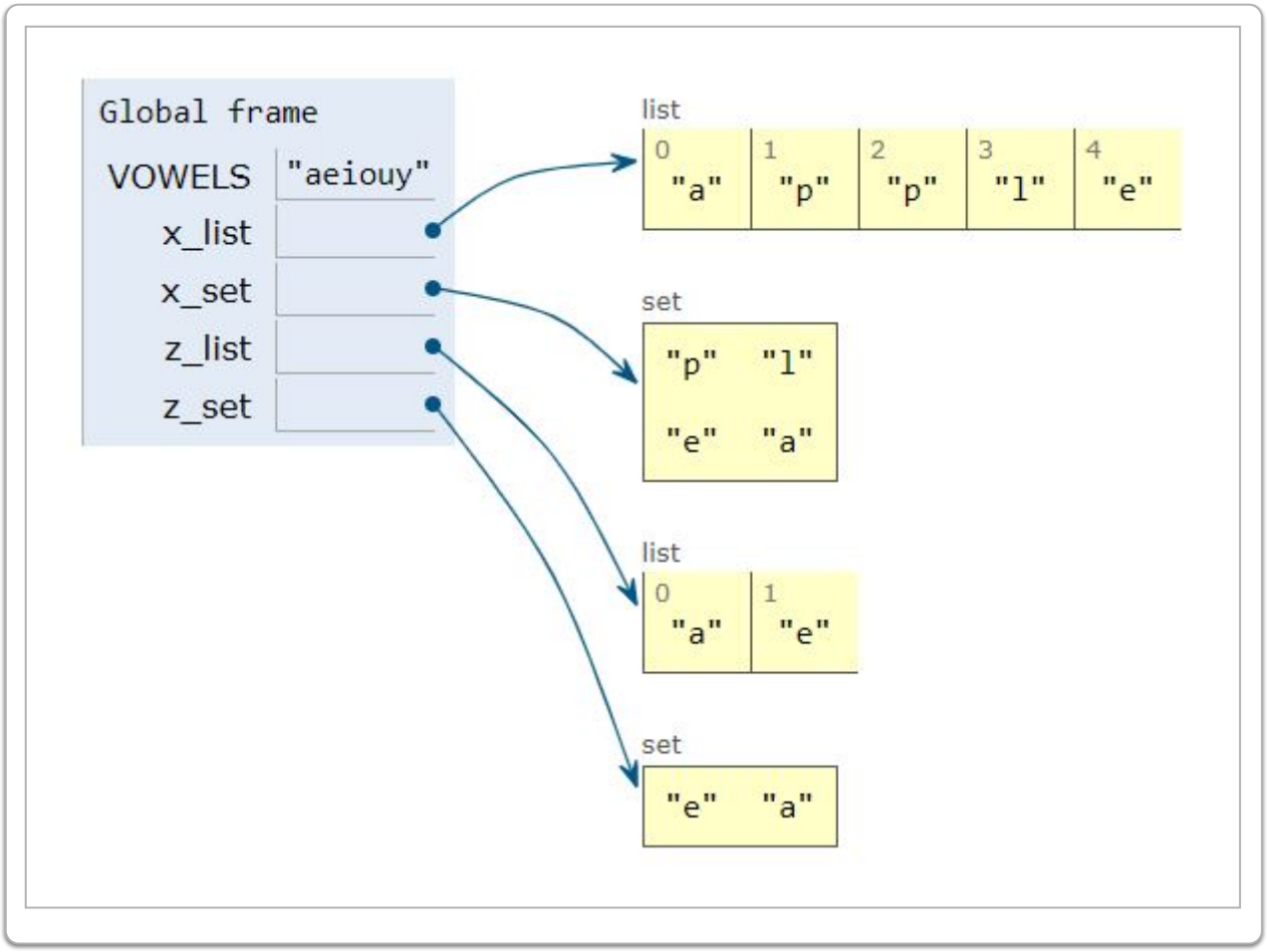
Solution:

```
VOWELS = set("aeoiuy")
x_set = set("wednesday")
y_set = {e for e in x_set if e not in VOWELS}
```



Set/List *comprehension*

```
VOWELS = 'aeiouy'
x_list = ['a','p','p','l','e']
x_set = {'a','p','p','l','e'}
z_list = [ e for e in x_list if e in VOWELS ]
z_set = { e for e in x_set if e in VOWELS }
```



update() Method

```
x_set = {'p','i','n','e'}
y_set = {'a','p','p','l','e'}
z_set = {'a','p','p','l','e'}
z_set.update(x_set) # merge two sets
```

Test Yourself: 4.1.04

Use *update*() to transform *x_set* into *y_set*:

```
x_set = {1, 2, 3, 4, 5}
y_set = {1, 2, 3, 7, 8}
```

Solution:

```
x_set = {1, 2, 3, 4, 5}
y_set = x_set.copy()
y_set.remove(4); y_set.remove(5)
y_set.update({7, 8})
```

add() & *clear*() Methods

```
x_set.add(10)
```

```
x_set.clear()
```

Test Yourself: 4.1.05

Change (in-place) the contents of x_set from:

```
x_set = {1, 2, 3}
```

to:

```
x_set = {4, 5, 6}
```

Solution:

```
x_set = {1, 2, 3}
y_set = x_set.copy()
x_set.clear()
x_set.add(4)
x_set.add(5)
x_set.add(6)
```

copy() & *pop*() Methods

```
y_set = x_set.copy()
```

```
x_value = x_set.pop() # remove random
```

Test Yourself: 4.1.06

Compute the sum of two elements from *x_set* chosen at random:

```
x_set = set(range(10))
```

Solution:

```
x_set = set(range(10))
f_random = x_set.pop()
s_random = x_set.pop()
print ("sum is ", f_random + s_random)
```

difference() Method(s)

```
z_set = x_set.difference(y_set)
```

```
x_set.difference_update(y_set)
```

Test Yourself: 4.1.07

Show two ways to construct a set containing elements from *x_set* but not from *y_set*:

```
x_set = {1, 2, 3, 4, 5, 6}
y_set = {3, 4}
```

Solution:

```
x_set = {1, 2, 3, 4, 5, 6}

w_set = x_set.copy()
w_set.difference_update(y_set)

z_set = x_set.difference(y_set)
```