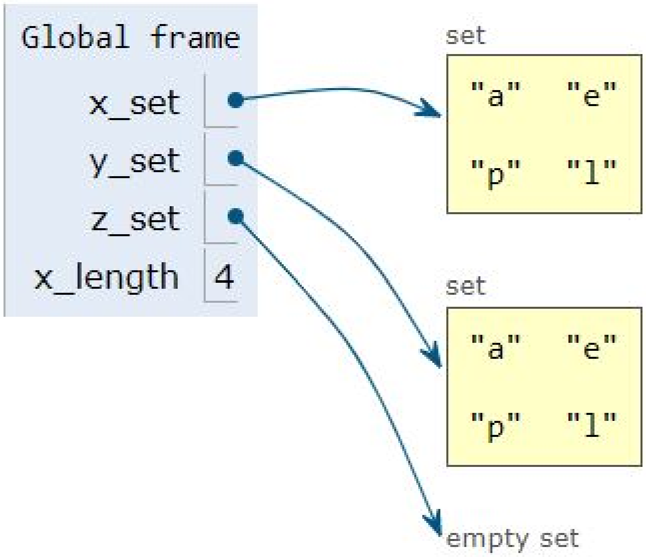
SETS

A Python Set

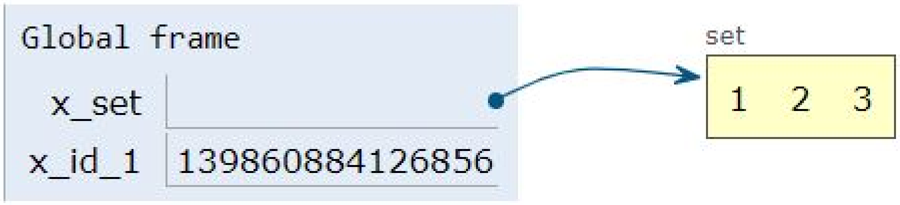
```
x_set    = {’a’,’p’,’p’,’l’,’e’}
y_set    = set(’apple’)
z_set    = set()
x_length = len(x_set)
```



- un-ordered & mutable
- unique hashable elements

Sets and Mutability

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



Global frame

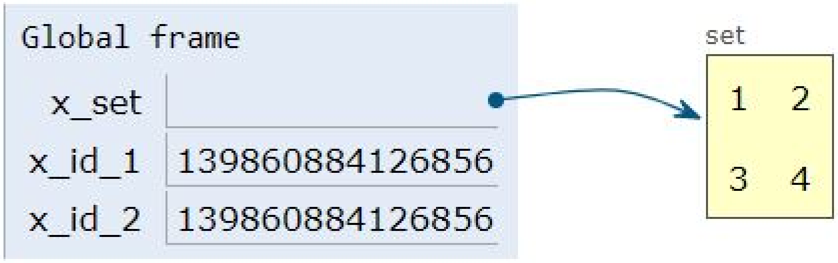
x_set

x_id_1 139860884126856

set

1 2 3

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



Global frame

x_set

x_id_1 139860884126856

x_id_2 139860884126856

set

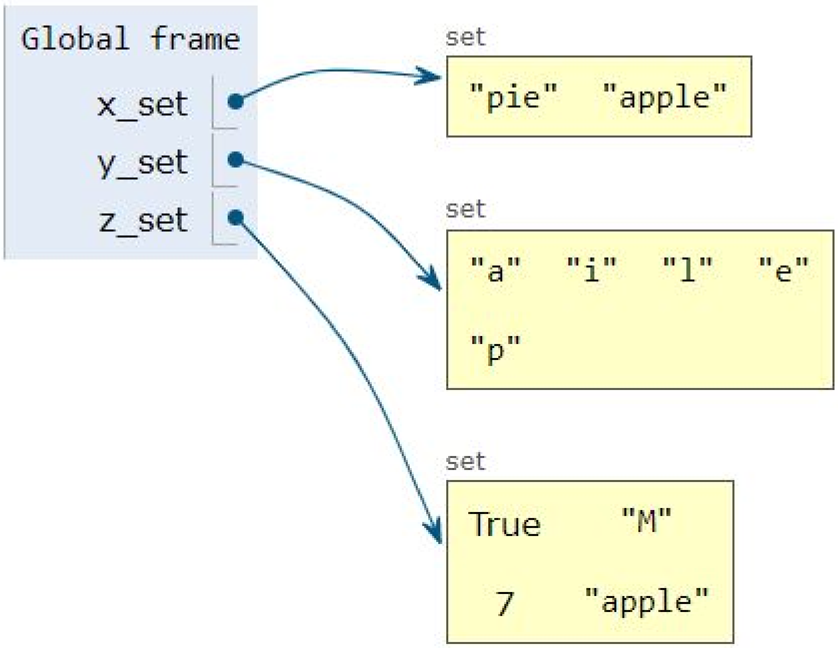
1 2

3 4

- sets are mutable

Sets from Primitive Types

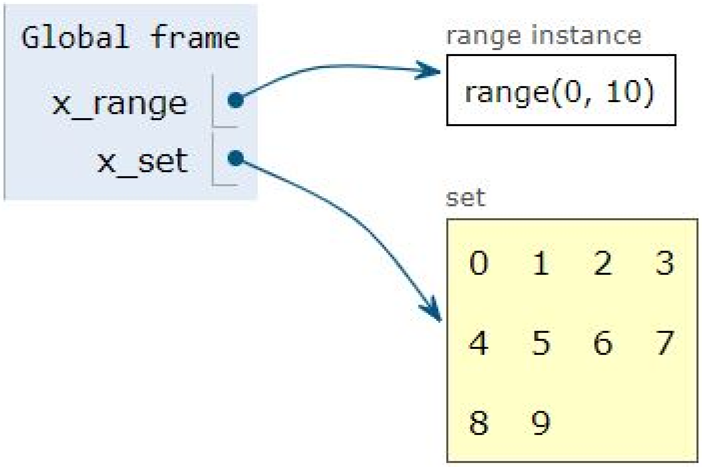
```
x_set = {’apple’, ’pie’}
y_set = set(’applepie’)
z_set  = {’apple’, 7, True, ’M’, 2+3j}
```



- all primitive types are hashable!!!

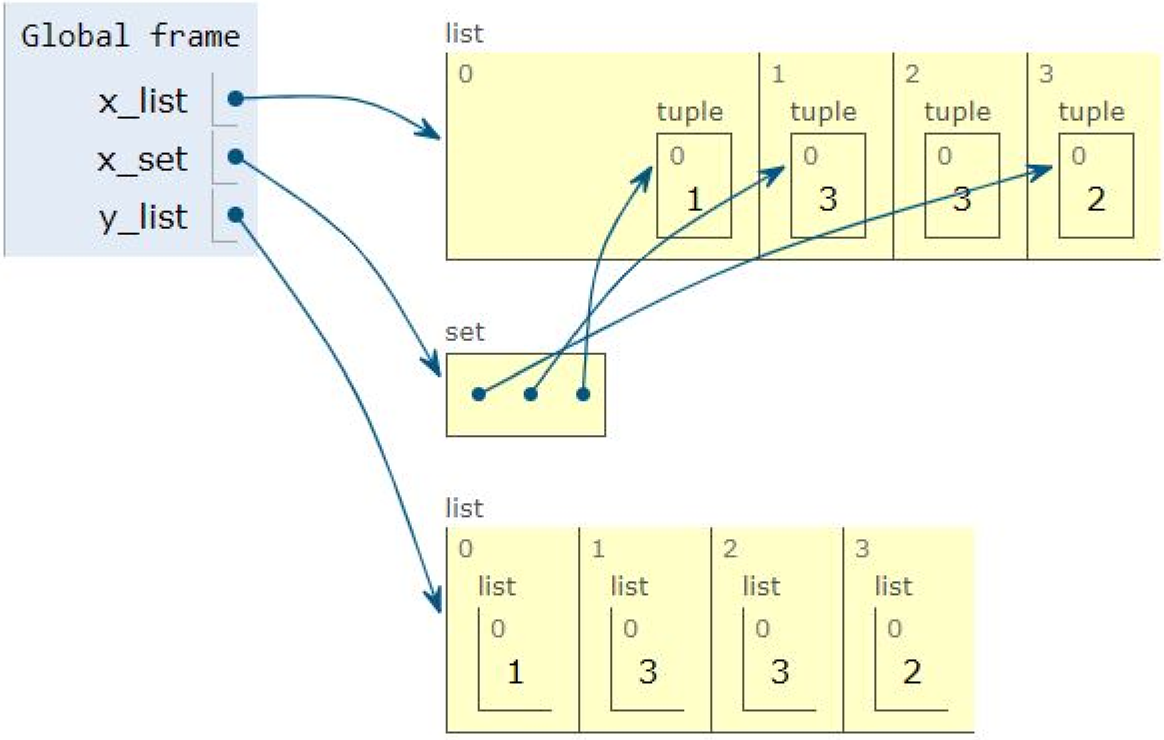
Sets from Ranges

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



Sets from Lists

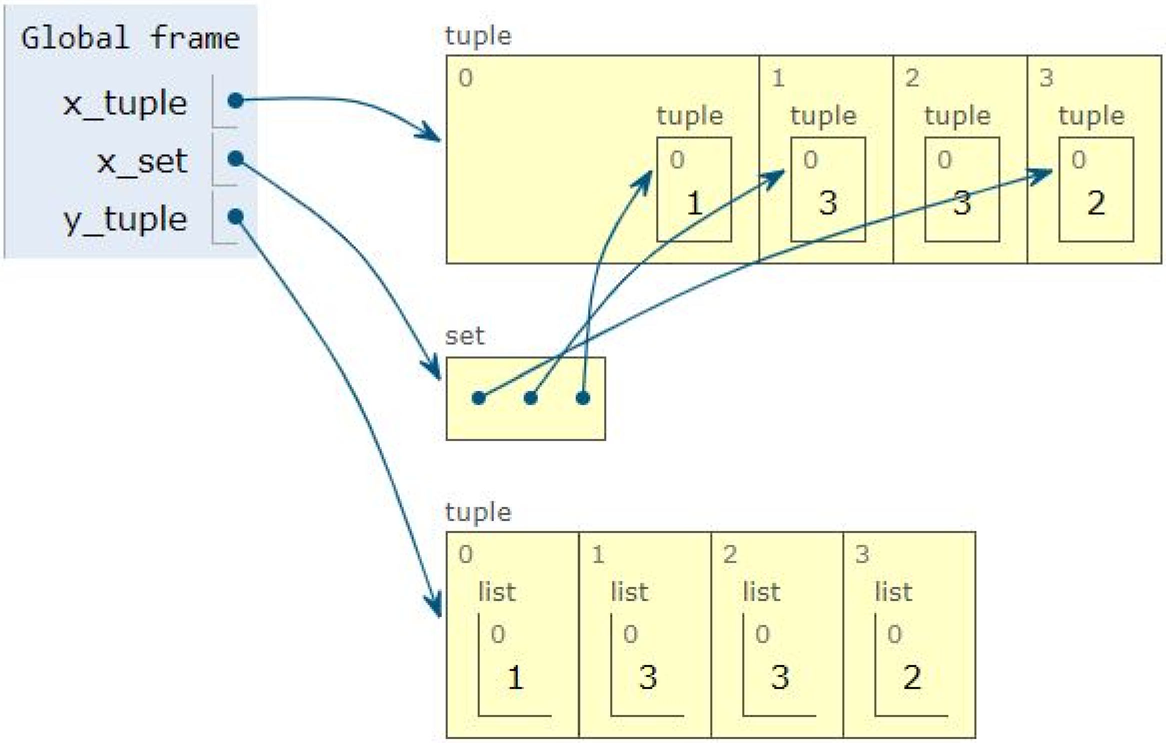
```
x_list = [ (1,), (3,), (3,), (2,)]
x_set  = set(x_list)
y_list = [  [1],   [3], [3], [2] ]
y_set  = set(y_list)     # illegal (unhashable)
```



- hashable elements only!

Sets from Tuples

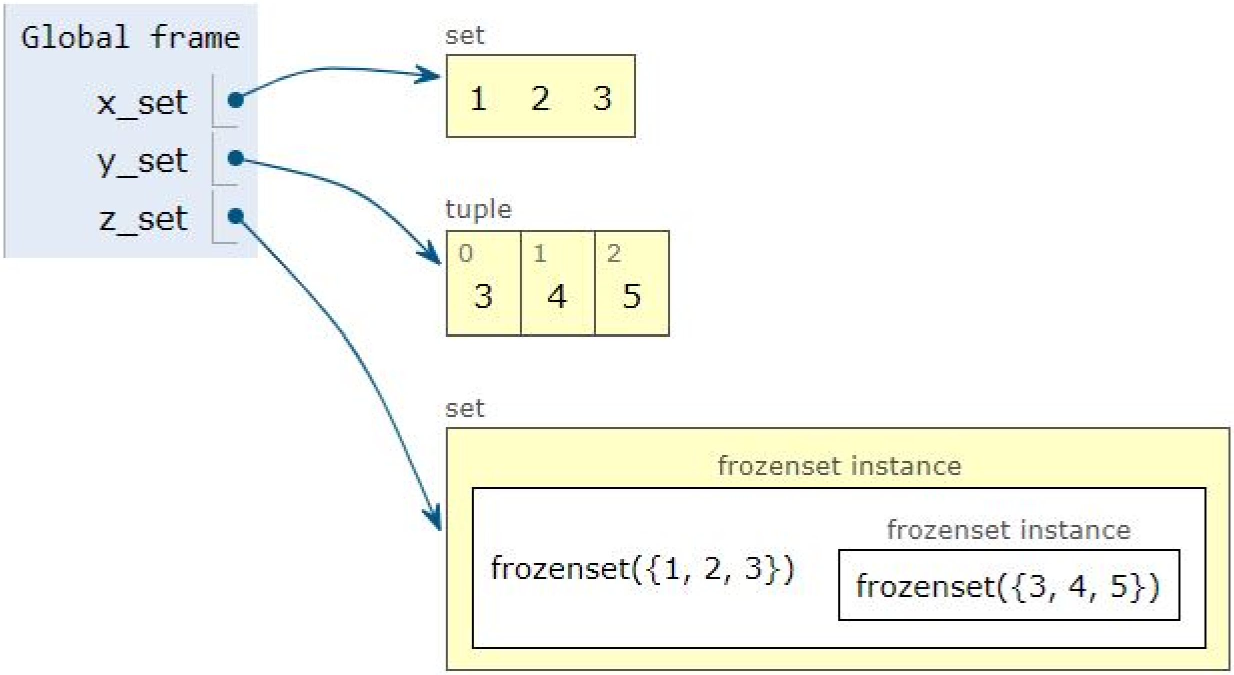
```
x_tuple = ( (1,), (3,), (3,), (2,))
x_set   = set(x_tuple)
y_tuple = (  [1],   [3], [3], [2] ) # illegal
```



- hashable elements only!

Sets from Frozen Sets

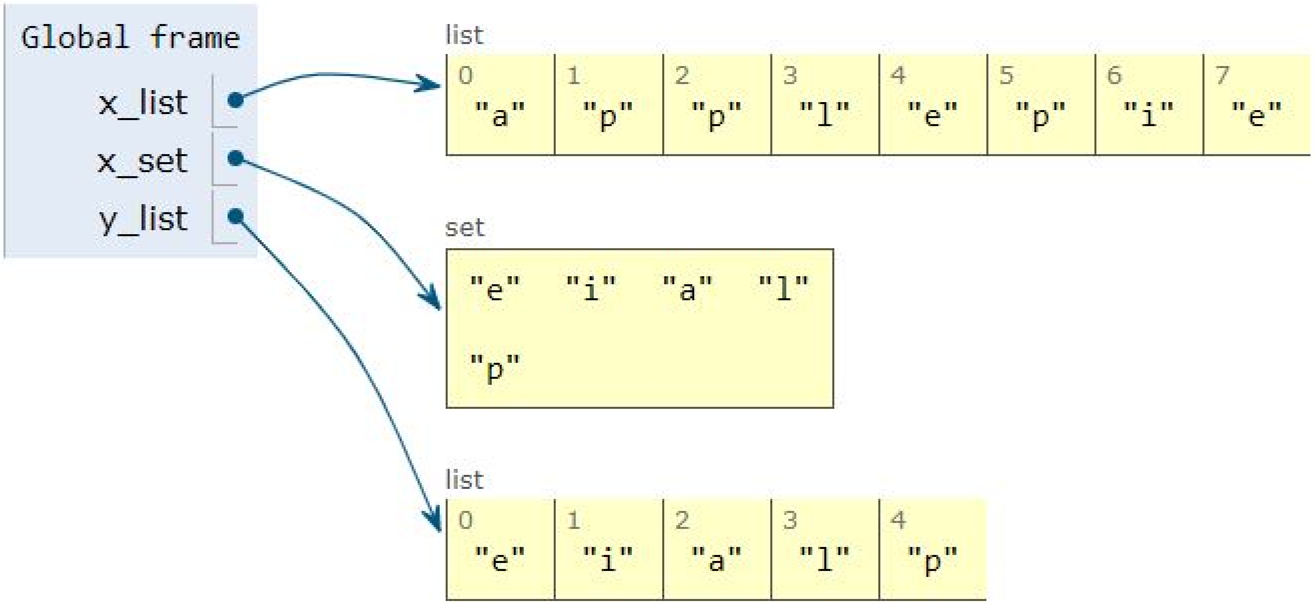
```
x_set = {1,2,3}
y_set = (3,4,5)

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)
```



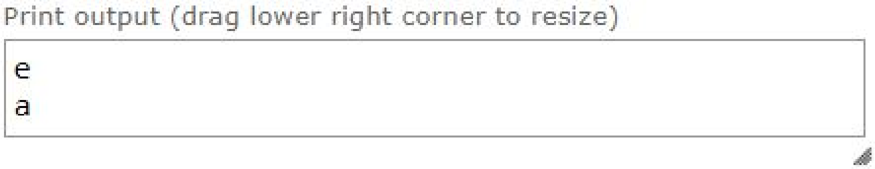
0	1	2	3	4	5	6	7
"a"	"p"	"p"	"l"	"e"	"p"	"i"	"e"

0	1	2	3	4
"e"	"i"	"a"	"l"	"p"

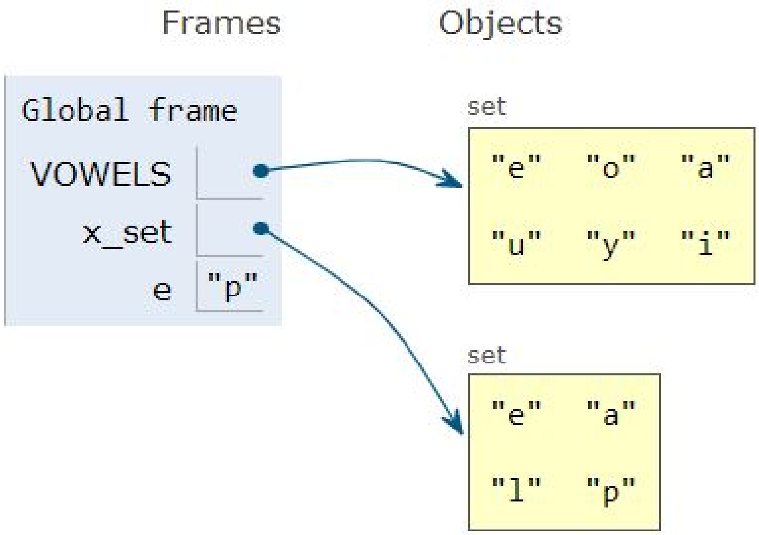
- no guarantee for ordering

Membership & Iteration

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



```
e
a
```



- iterable but not indexed

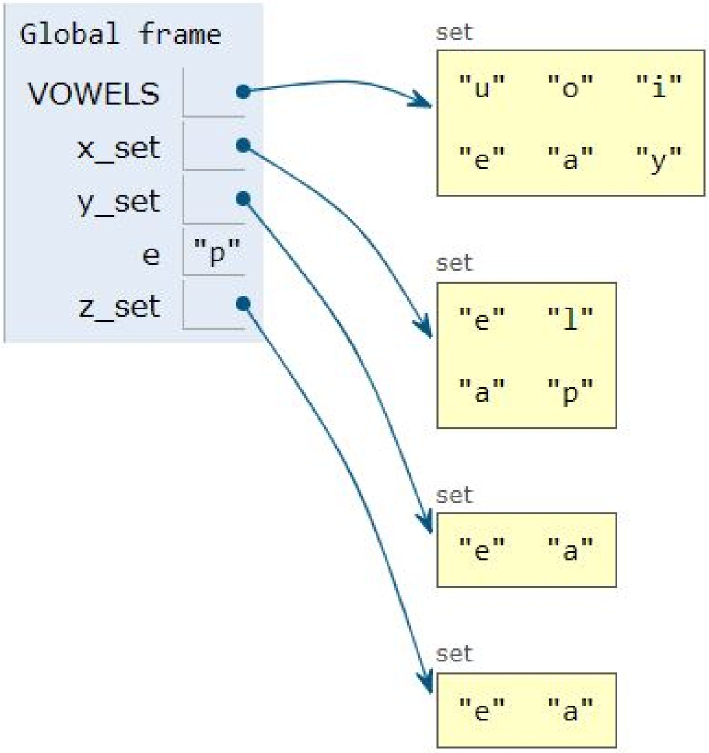
Exercise(s):

- print consonants in *x_set*:

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

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 }
```



Exercise(s):

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

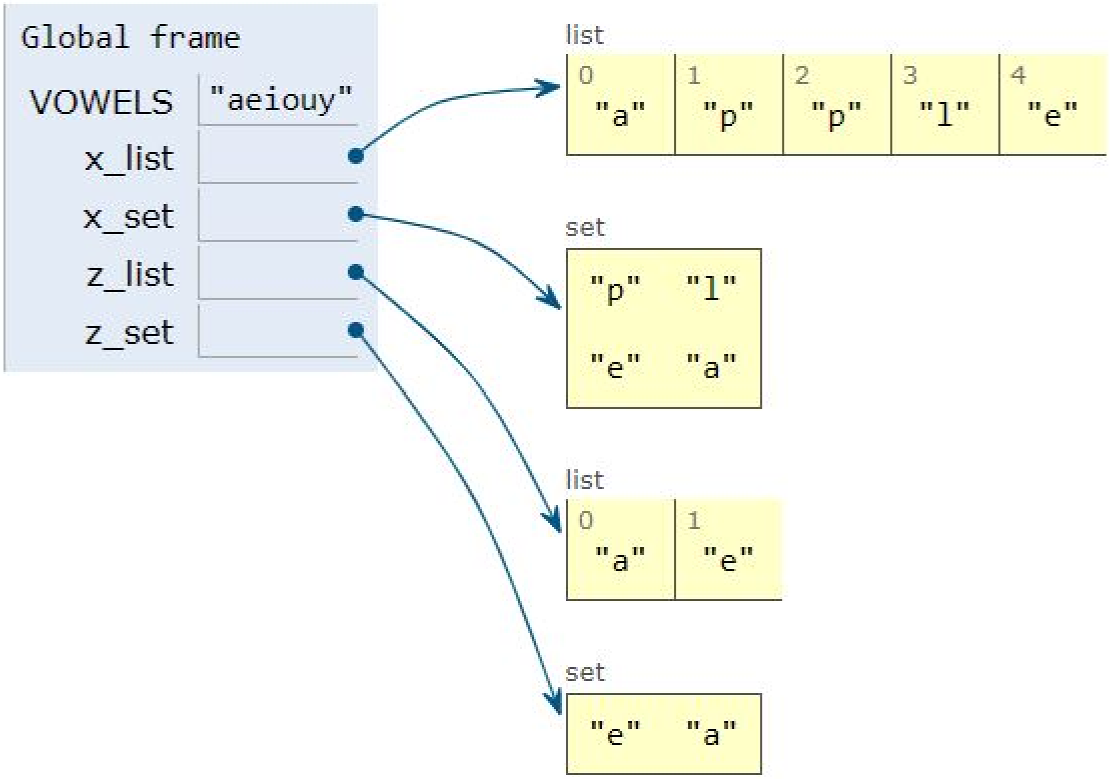
  ```
  x_set = [1,-5,-7, 3,-2]
  y_set = [-5,-7,-2]
  ```

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

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

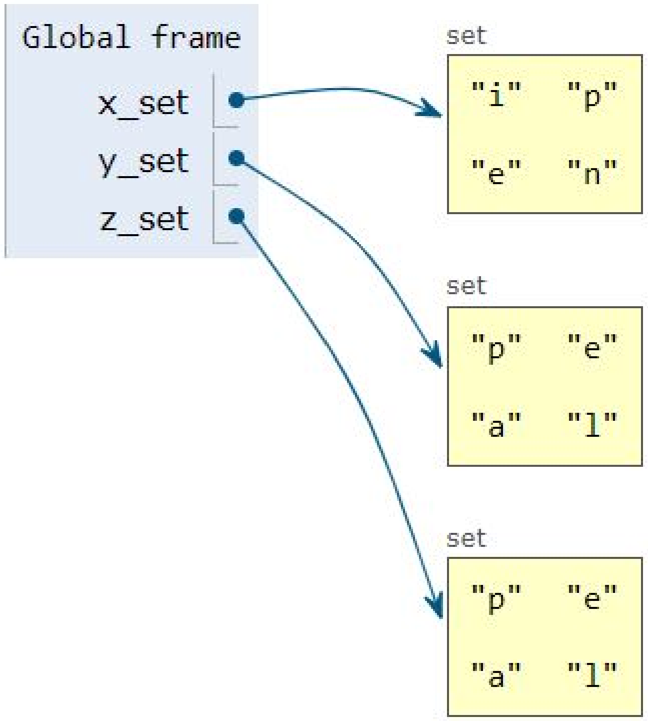
Set/List *comprehension*

```
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
```

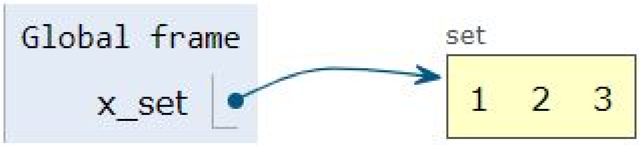


Exercise(s):

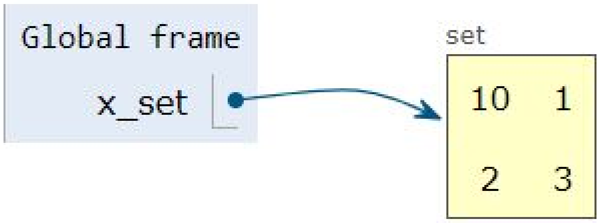
- use *update*() to transform *x_set* into *y_set*

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

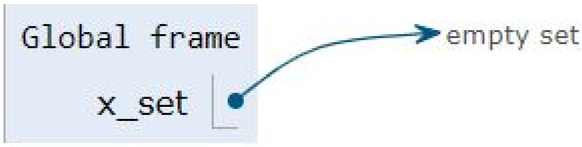
add() & *clear*() Methods



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



```
x_set.clear()
```



Exercise(s):

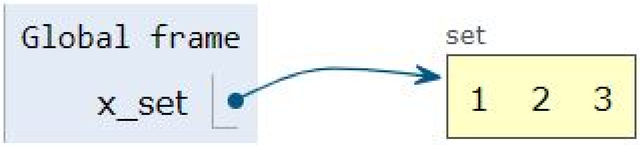
- change (in-place) the contents of *x_set* from

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

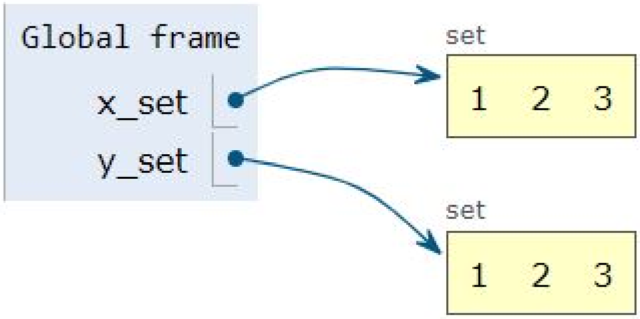
  to:

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

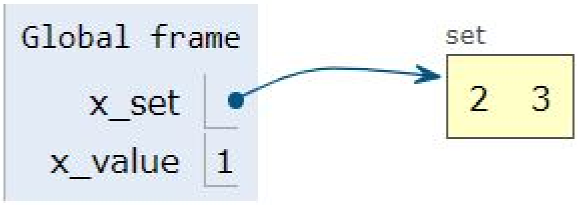
copy() & *pop*() Methods



```
w_set = x_set.copy()
```



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

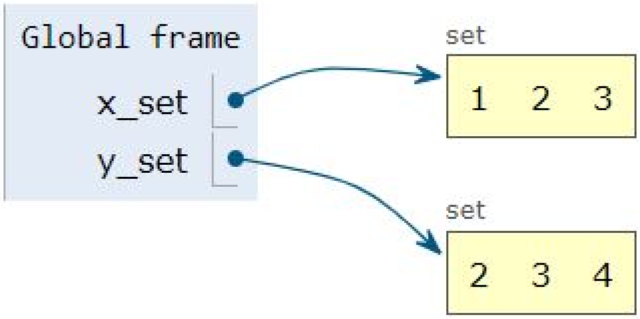


Exercise(s):

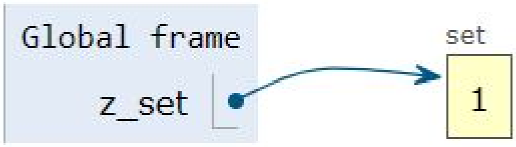
- compute the sum of two elements from *x_set* chosen at random

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

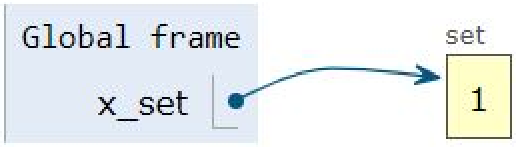
difference() Method(s)



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



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

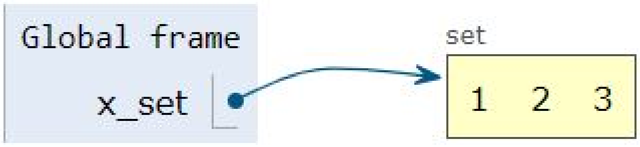


Exercise(s):

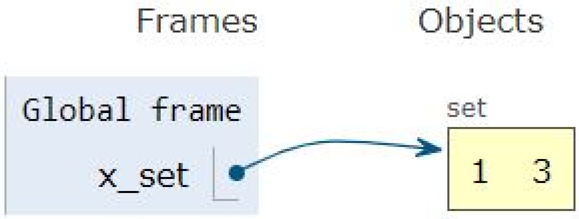
- show two ways to construct a set containing elements from *x_set* but not from *y_set*:

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

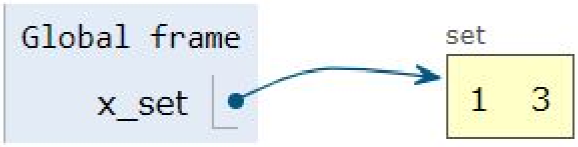
discard() & *remove*()



```
x_set.discard(2)
x_set.discard(10)          # no error if missing
```



```
x_set.remove(2)
x_set.remove(10)          # error if missing
```

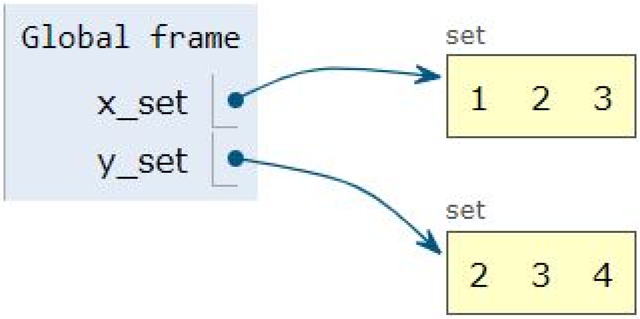


Exercise(s):

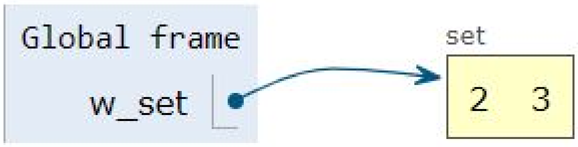
- show two different ways to remove even numbers from x_set

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

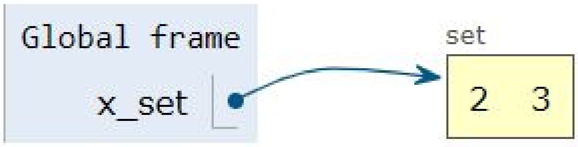
intersection() **Method(s)**



```
w_set = x_set.intersection(set_y)
```



```
x_set.intersection_update(y_set)
```

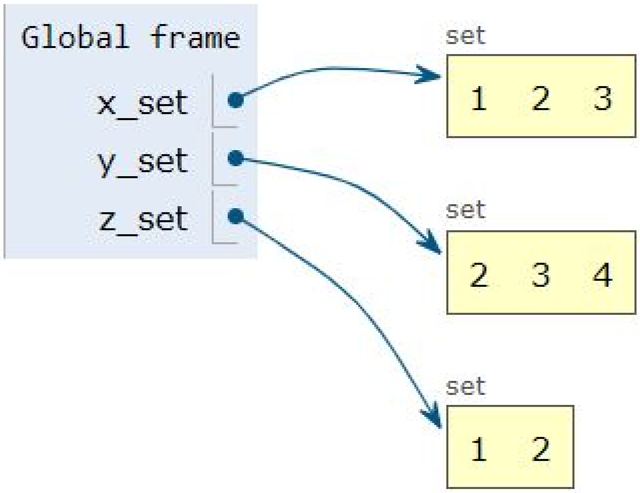


Exercise(s):

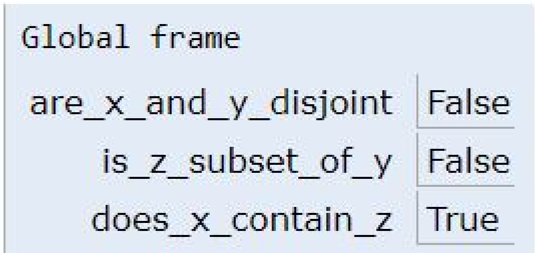
- compute characters that are both in *x_set* and *y_set*

```
x_set = set("sunday")
y_set = set("tuesday")
```

subset() Method(s)



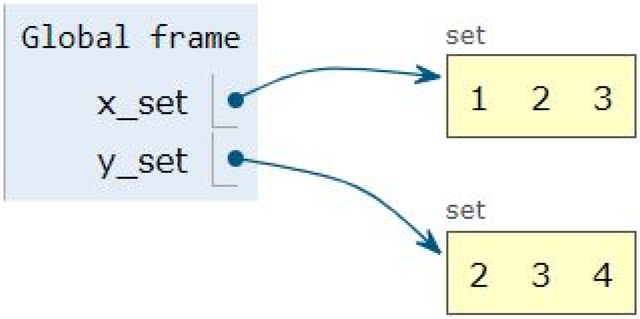
```
are_x_and_y_disjoint = x_set.isdisjoint(y_set)
is_z_subset_of_y     = z_set.issubset(y_set)
does_x_contain_z     = x_set.issuperset(z_set)
```



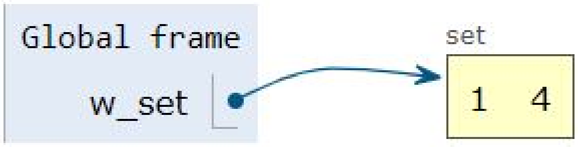
Global frame

are_x_and_y_disjoint	False
is_z_subset_of_y	False
does_x_contain_z	True

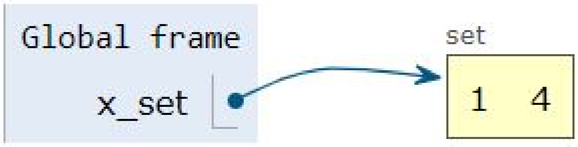
symmetric_difference()



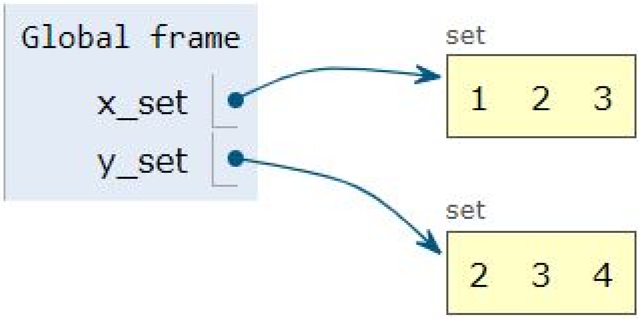
```
w_set = x_set.symmetric_difference(set_y)
```



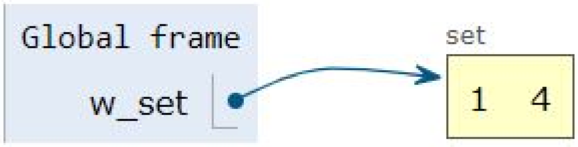
```
x_set.symmetric_difference_update(y_set)
```



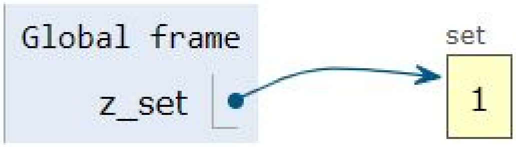
Symmetric Difference vs. Difference



```
w_set = x_set.symmetric_difference(set_y)
```



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

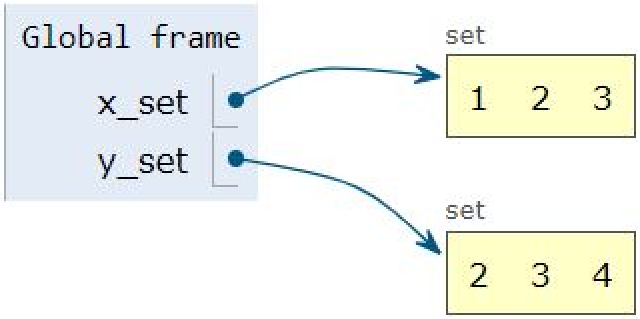


Exercise(s):

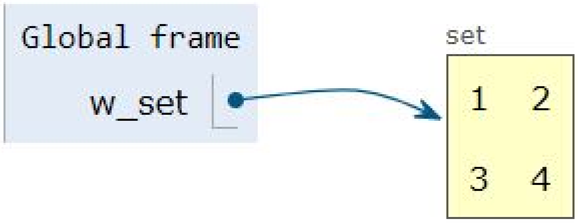
- find a set of characters that are in *x_set* but not in *y_set*

```
x_set = set("sunday")
y_set = set("tuesday")
```

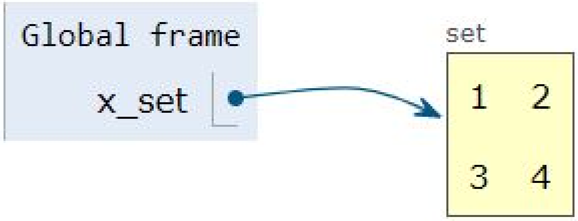
union() vs. *update*()



```
w_set = x_set.union(set_y)
```



```
x_set.update(y_set)
```



Exercise(s):

- find a set of characters that are either in *x_set* or *y_set*

```
x_set = set("sunday")
y_set = set("tuesday")
```

Jacard's Similarity

- similarity metric for sets

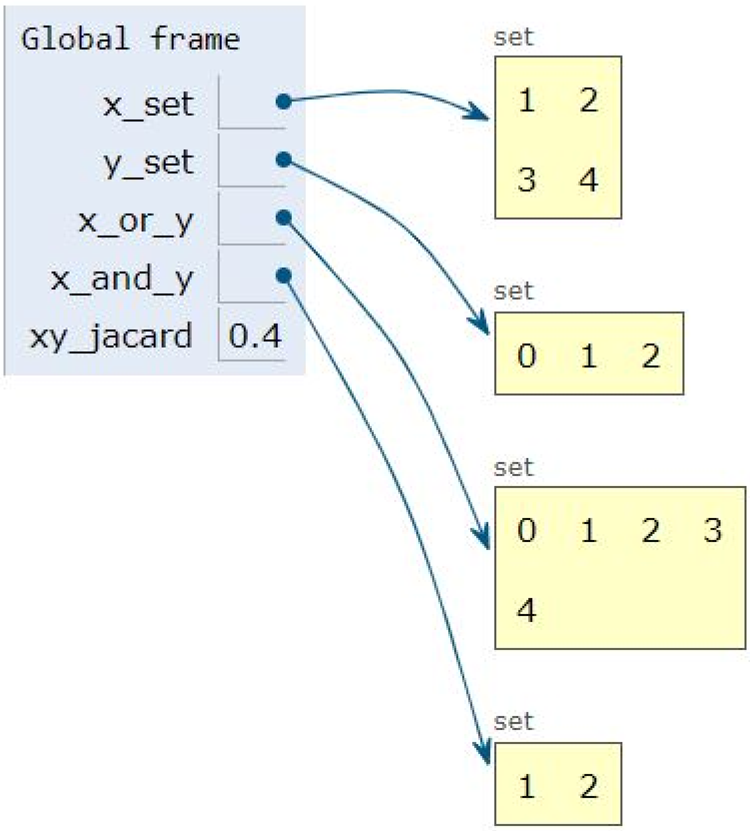
$$\text{Jacard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- widely used in data science
- minimum 0 (A, B disjoint)
- maximum 1 (A, B identical)

```
x_set     = {1, 2, 3, 4}
y_set     = {0, 1, 2}
x_or_y    = x_set.union(y_set)
x_and_y   = x_set.intersection(y_set)
xy_jacard = len(x_and_y) / len(x_or_y)
```

Jacard's Similarity

$$\text{Jacard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$



Exercise(s):

- compute Jacard's similarity for each pair of sets from *x_set*, *y_set*, *z_set*:

  ```
  x_set = set("sunday")
  y_set = set("tuesday")
  z_set = set("thursday")
  ```

- which two sets are most similar?

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

Summary:

- unordered collection
- iterable and mutable
- unique, immutable and hashable elements
- many methods