Python CS-521

Eugene Pinsky
Department of Computer Science
Metropolitan College, Boston University
Boston, MA 02215
email: epinsky@bu.edu

May 17, 2020

Abstract

This course will present an effective approach to help you learn Python. With extensive use of graphical illustrations, we will build understanding of Python and its capabilities by learning through many simple examples and analogies. The class will involve active student participation, discussions, and programming exercises. This approach will help you build a strong foundation in Python that you will be able to effectively apply in real-job situations and future courses.

EXERCISES

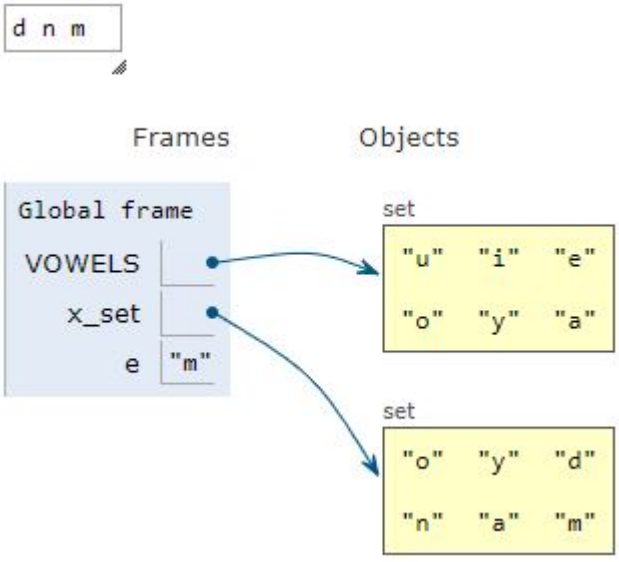
SETS

- 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 = " ")
```

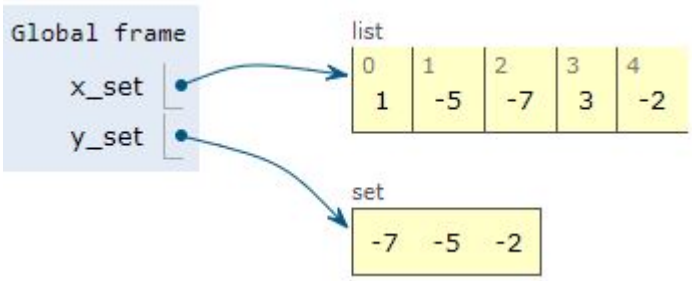


- use set comprehension to construct *y_set* with negative elements from *x_set*:

```
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}
```

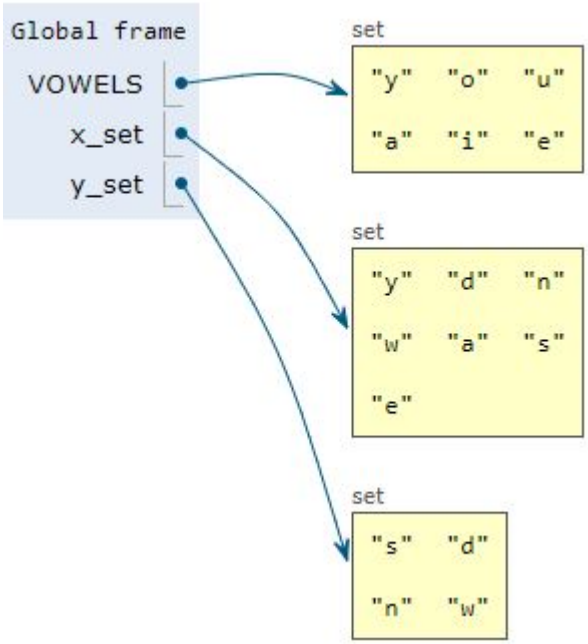


- use set comprehension to construct a list of consonants in *x_set*:

```
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}
```

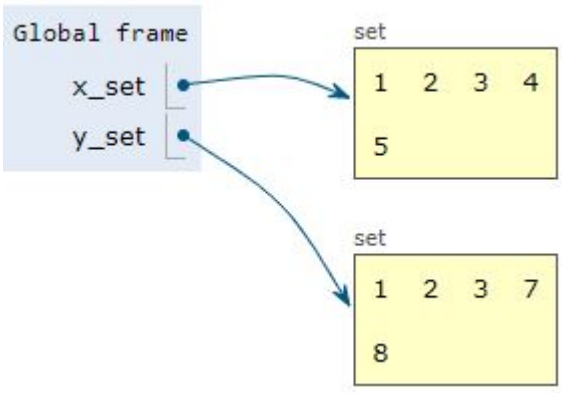

Global frame
VOWELS
x_set
y_set
set
"y" "o" "u"
"a" "i" "e"
set
"y" "d" "n"
"w" "a" "s"
"e"
set
"s" "d"
"n" "w"

- 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})
```



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

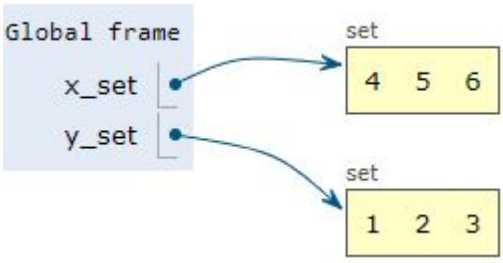
```
x_set = {1, 2, 3}
```

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)
```

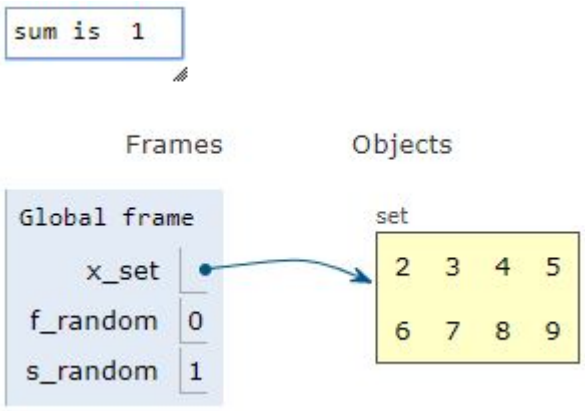

Global frame
x_set
y_set
set
4 5 6
set
1 2 3

- 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)
```



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

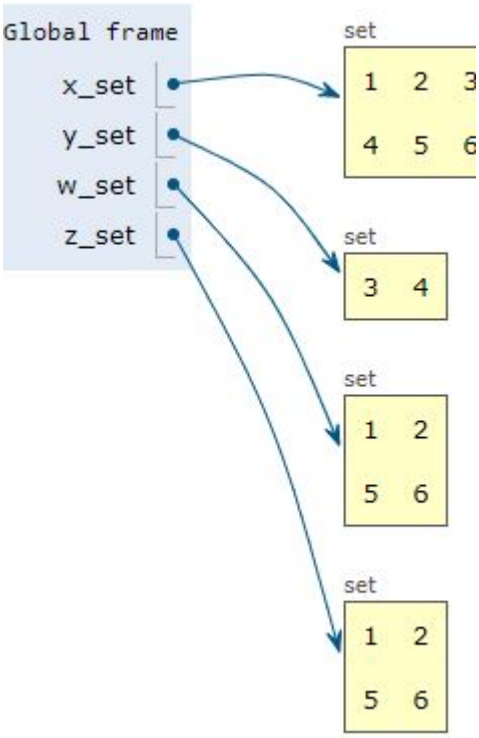
```
x_set = {1, 2, 3, 4, 5, 6}
y_set = {3, 4}
```

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)
```


Global frame
x_set
y_set
w_set
z_set
set
1 2 3
4 5 6
set
3 4
set
1 2
5 6
set
1 2
5 6

- show two different ways to remove even numbers from *x_set*

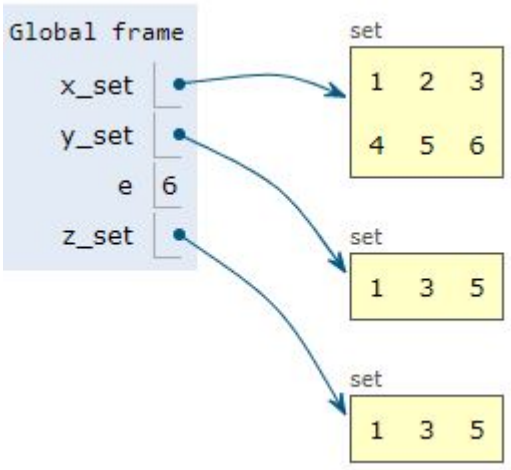
```
x_set = {1, 2, 3, 4, 5, 6}
```

Solution:

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

for e in x_set:
    if e % 2 == 0:
        y_set.discard(e)    # or remove

# second method
z_set = {e for e in x_set if e % 2 == 1}
```

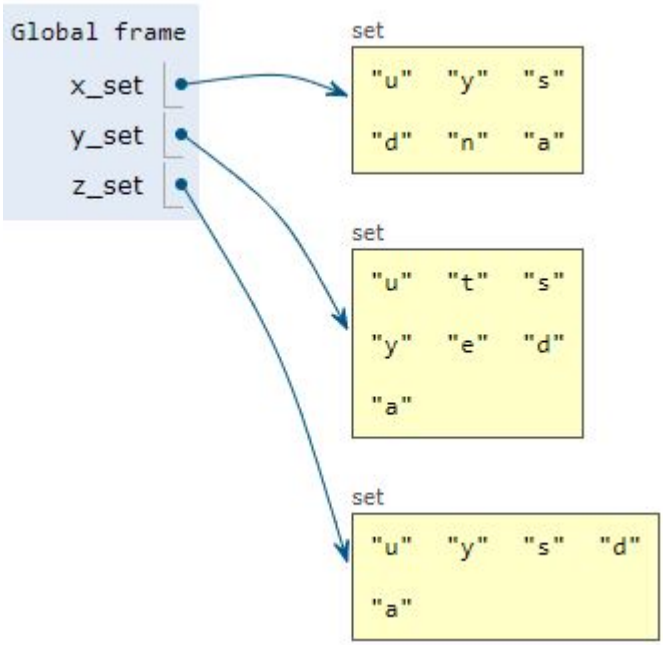

Global frame
x_set
y_set
e
6
z_set
set
1 2 3
4 5 6
set
1 3 5
set
1 3 5

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

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

Solution:

```
x_set = set("sunday")
y_set = set("tuesday")
z_set = x_set.intersection(y_set)
```

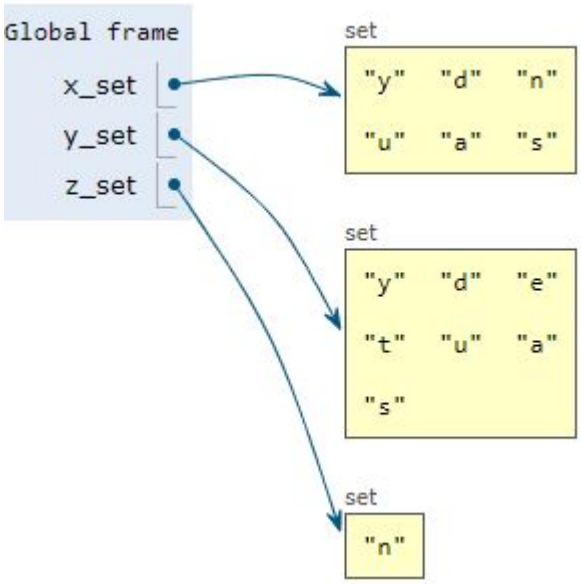

Global frame
x_set
y_set
z_set
set
"u" "y" "s"
"d" "n" "a"
set
"u" "t" "s"
"y" "e" "d"
"a"
set
"u" "y" "s" "d"
"a"

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

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

Solution:

```
x_set = set("sunday")
y_set = set("tuesday")
z_set = x_set.difference(y_set)
```

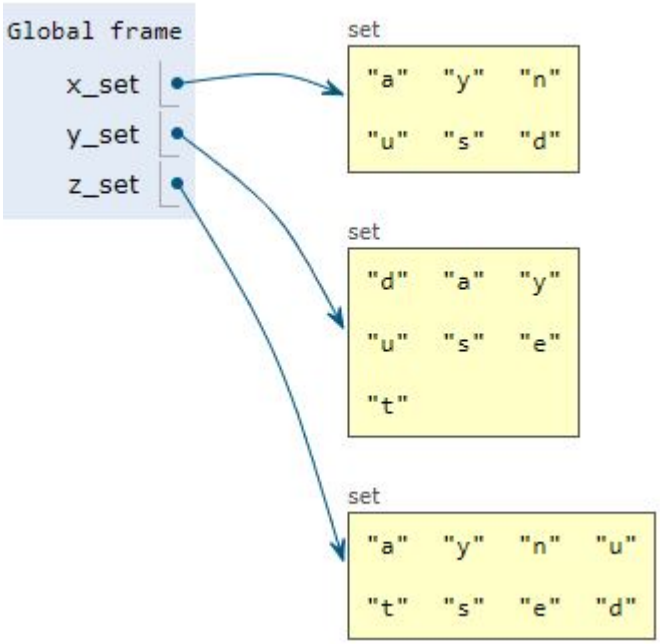

Global frame
x_set
y_set
z_set
set
"y" "d" "n"
"u" "a" "s"
set
"y" "d" "e"
"t" "u" "a"
"s"
set
"n"

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

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

Solution:

```
x_set = set("sunday")
y_set = set("tuesday")
z_set = x_set.union(y_set)
```


Global frame
x_set
y_set
z_set
set
"a" "y" "n"
"u" "s" "d"
set
"d" "a" "y"
"u" "s" "e"
"t"
set
"a" "y" "n" "u"
"t" "s" "e" "d"

- 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")
```

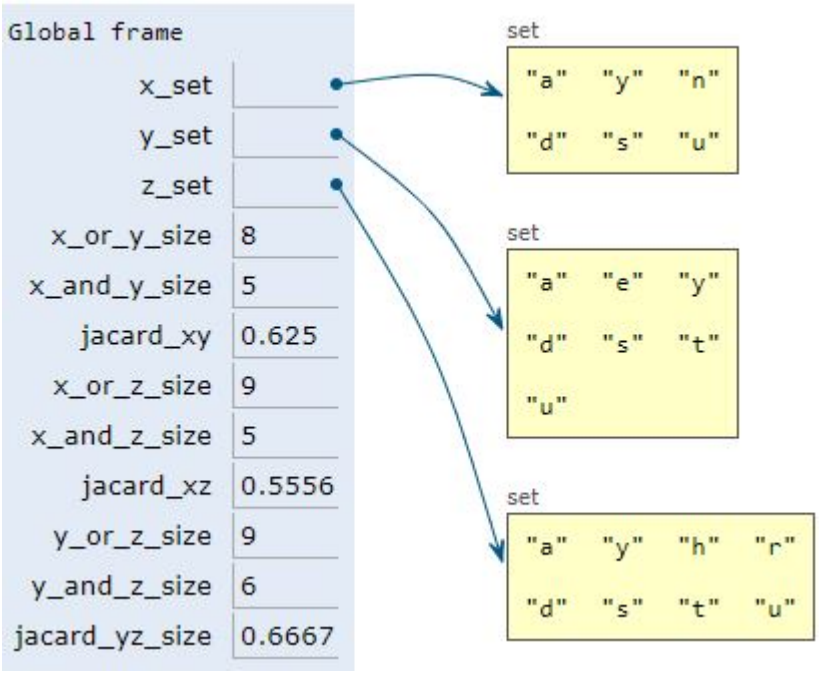
Solution:

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

x_or_y_size = len(x_set.union(y_set))
x_and_y_size   = len(x_set.intersection(y_set))
jacard_xy = x_and_y_size / x_or_y_size

x_or_z_size    = len(x_set.union(z_set))
x_and_z_size   = len(x_set.intersection(z_set))
jacard_xz = x_and_z_size / x_or_z_size

y_or_z_size    = len(y_set.union(z_set))
y_and_z_size   = len(y_set.intersection(z_set))
jacard_yz_size = y_and_z_size / y_or_z_size
```


Global frame
x_set
y_set
z_set
x_or_y_size 8
x_and_y_size 5
jacard_xy 0.625
x_or_z_size 9
x_and_z_size 5
jacard_xz 0.5556
y_or_z_size 9
y_and_z_size 6
jacard_yz_size 0.6667
set
"a" "y" "n"
"d" "s" "u"
set
"a" "e" "y"
"d" "s" "t"
"u"
set
"a" "y" "h" "r"
"d" "s" "t" "u"

- which two sets are most similar?

Solution:

- most similar: *y_set* and *z_set*
- they have highest Jaccard's similarity