COMP1819
Algorithms and Data Structures

Lecture 08: Sets & CW & Revisions

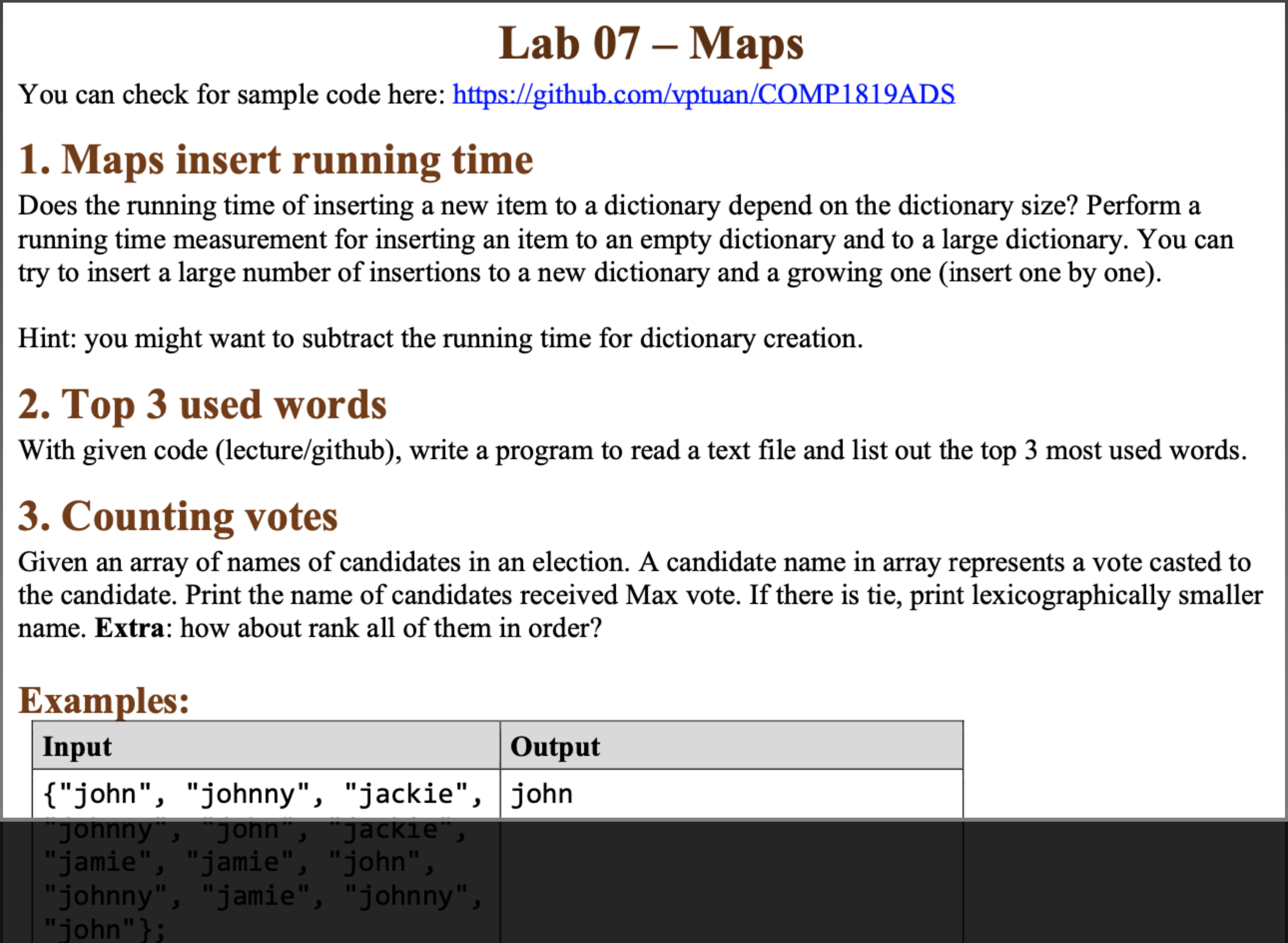
Dr. Tuan Vuong

09/03/2021

Content

- Lab 07 Walk-through
- Python Sets
- Sets ADT
- Revisions Lecture 01-07
- CW Q&A

Walk-through



Lab 07 – Maps

You can check for sample code here: https://github.com/vptuan/COMP1819ADS

1. Maps insert running time

Does the running time of inserting a new item to a dictionary depend on the dictionary size? Perform a running time measurement for inserting an item to an empty dictionary and to a large dictionary. You can try to insert a large number of insertions to a new dictionary and a growing one (insert one by one).

Hint: you might want to subtract the running time for dictionary creation.

2. Top 3 used words

With given code (lecture/github), write a program to read a text file and list out the top 3 most used words.

3. Counting votes

Given an array of names of candidates in an election. A candidate name in array represents a vote casted to the candidate. Print the name of candidates received Max vote. If there is tie, print lexicographically smaller name. **Extra**: how about rank all of them in order?

Examples:

Input	Output
{"john", "johnny", "jackie", "johnny", "john", "jackie", "jamie", "jamie", "john", "johnny", "jamie", "johnny", "john"};	john

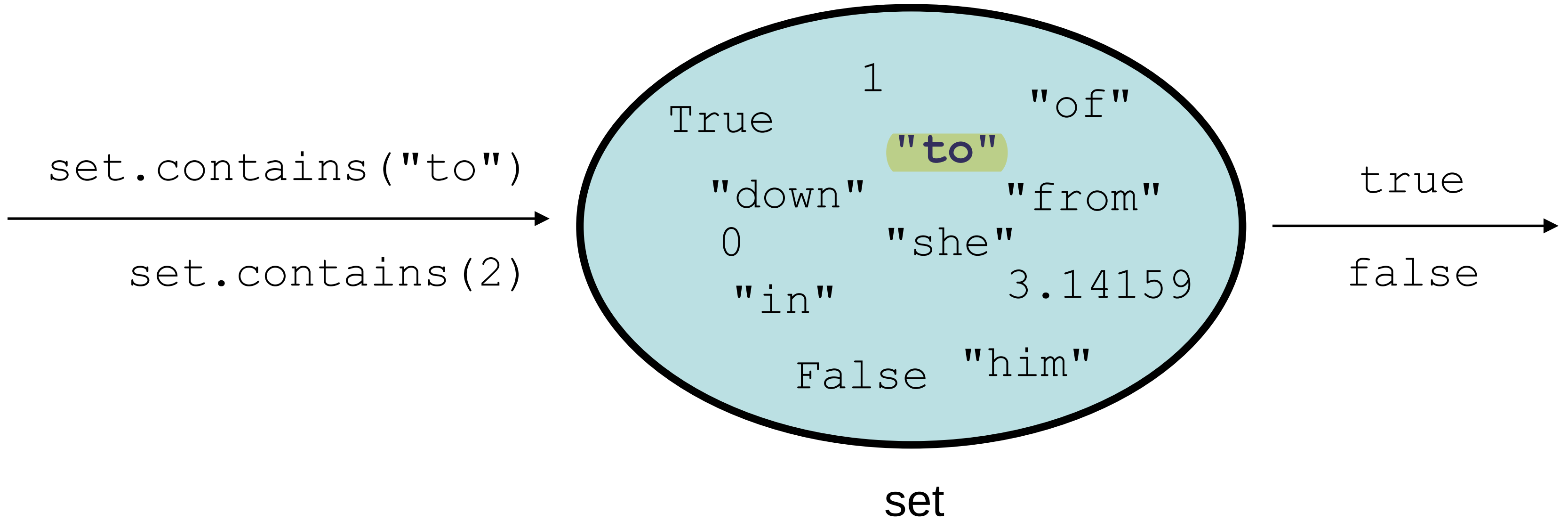
Today

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

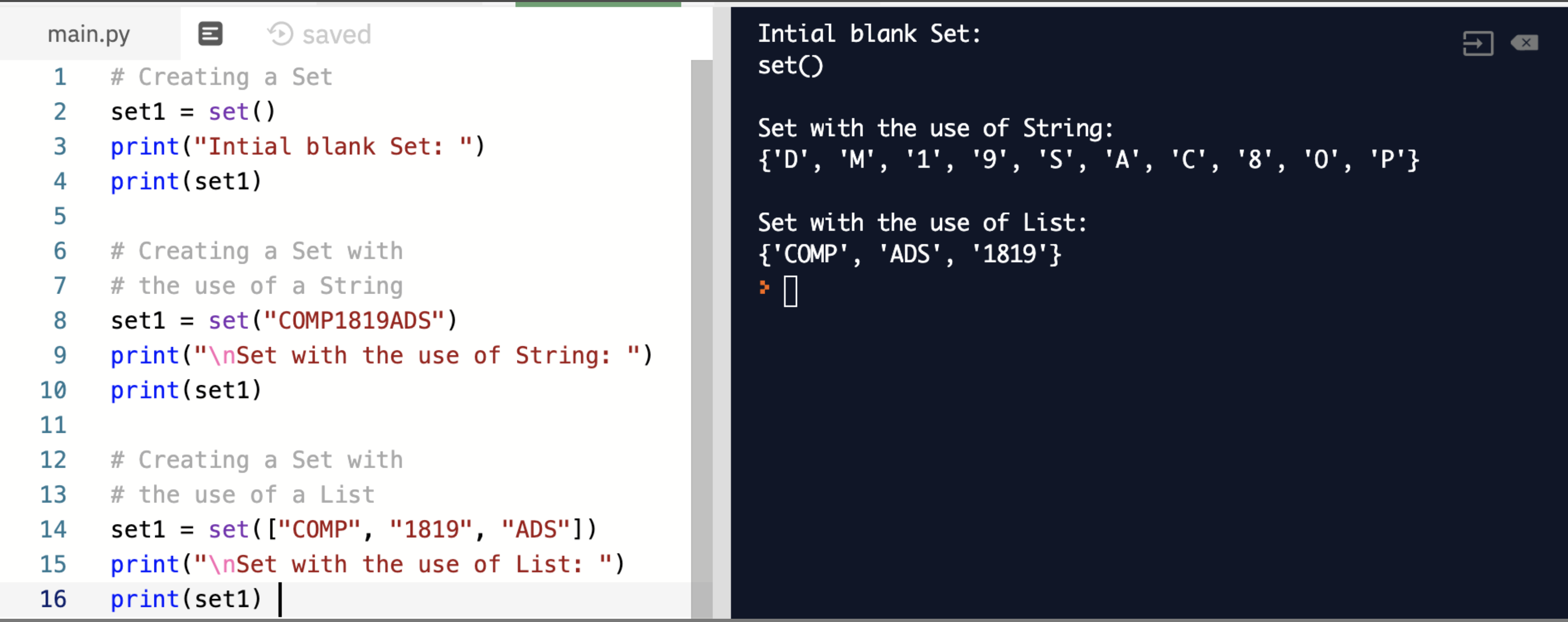
https://www.thegeekstuff.com/

Definitions

- A **set** is an unordered collection of elements, without duplicates that typically supports efficient membership tests.



Operations: create

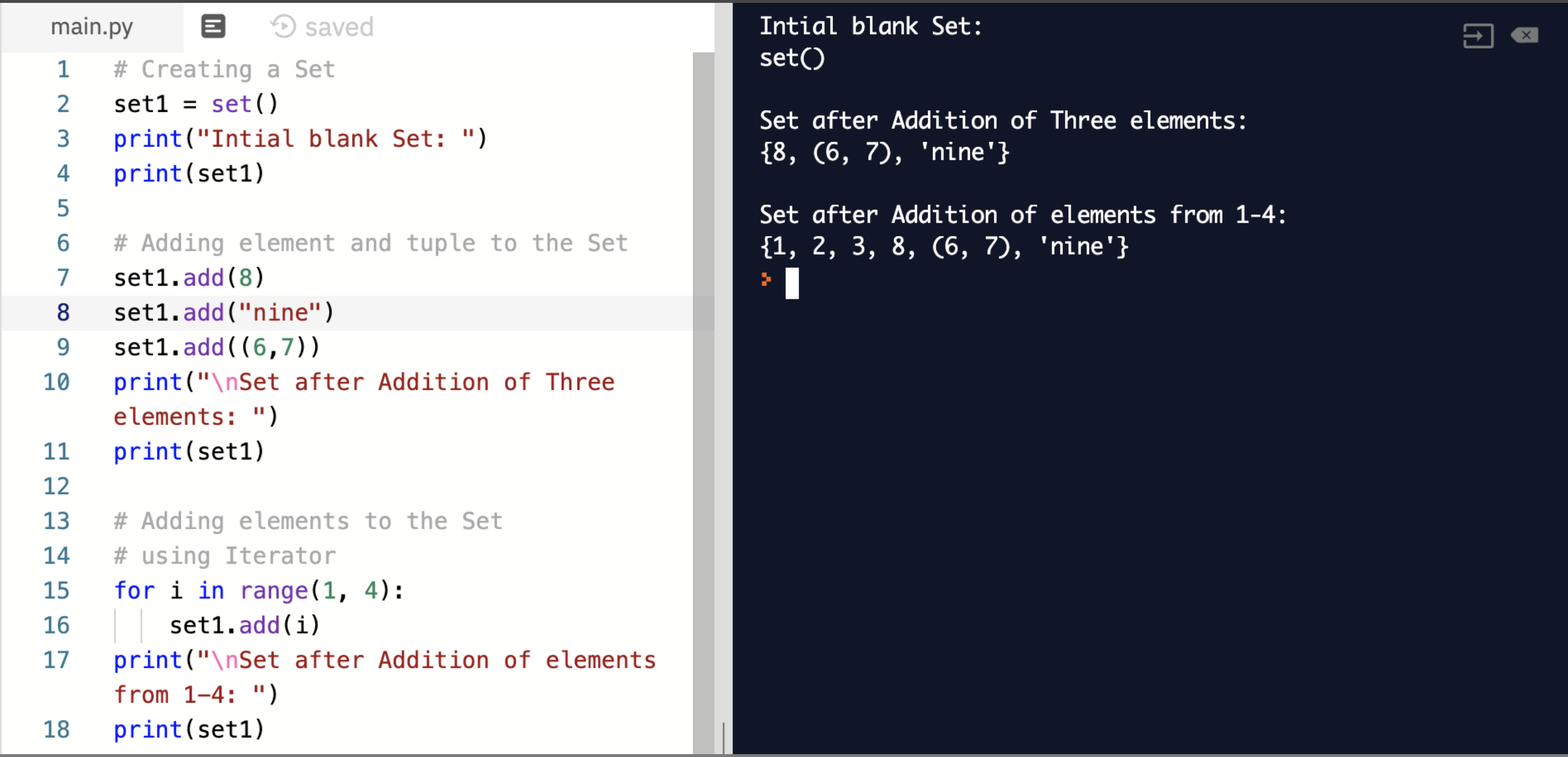


main.py saved

```python
# Creating a Set
set1 = set()
print("Intial blank Set: ")
print(set1)

# Creating a Set with
# the use of a String
set1 = set("COMP1819ADS")
print("\nSet with the use of String: ")
print(set1)

# Creating a Set with
# the use of a List
set1 = set(["COMP", "1819", "ADS"])
print("\nSet with the use of List: ")
print(set1)
```

```
Intial blank Set:
set()

Set with the use of String:
{'D', 'M', '1', '9', 'S', 'A', 'C', '8', 'O', 'P'}

Set with the use of List:
{'COMP', 'ADS', '1819'}
```

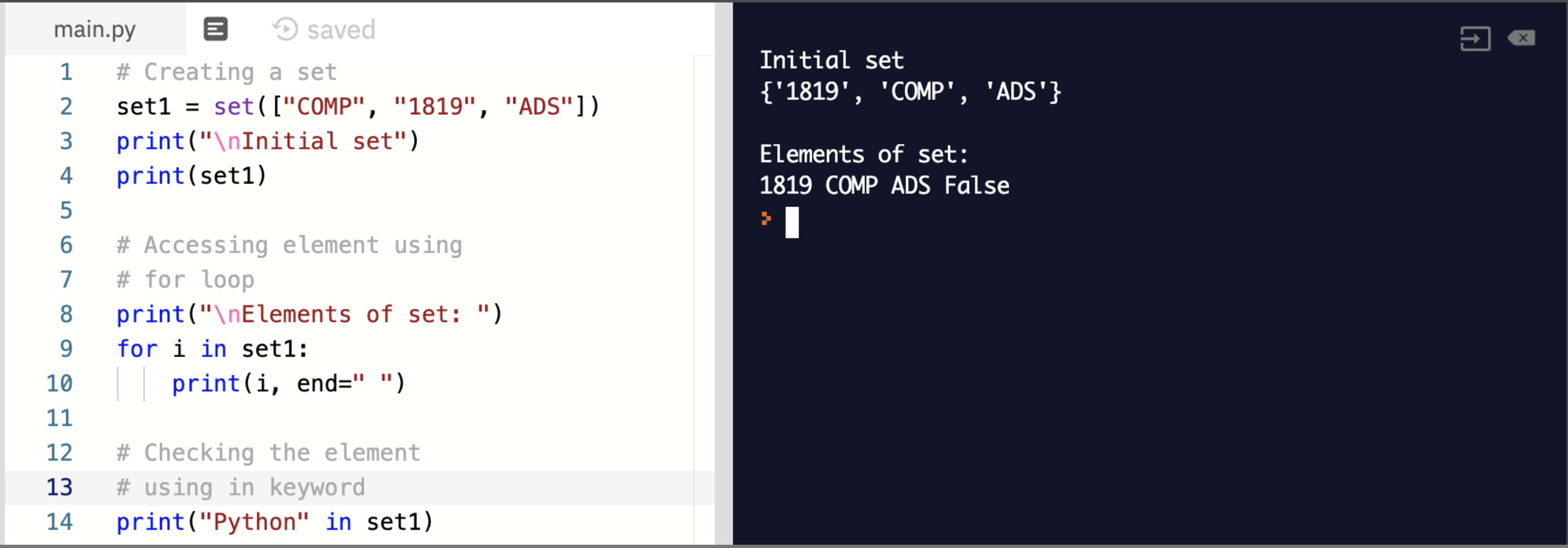
Operations: add



```python
# Creating a Set
set1 = set()
print("Intial blank Set: ")
print(set1)

# Adding element and tuple to the Set
set1.add(8)
set1.add("nine")
set1.add((6,7))
print("\nSet after Addition of Three elements: ")
print(set1)

# Adding elements to the Set
# using Iterator
for i in range(1, 4):
    set1.add(i)
print("\nSet after Addition of elements from 1-4: ")
print(set1)
```

```
Intial blank Set:
set()

Set after Addition of Three elements:
{8, (6, 7), 'nine'}

Set after Addition of elements from 1-4:
{1, 2, 3, 8, (6, 7), 'nine'}
>
```

Operations: accessing

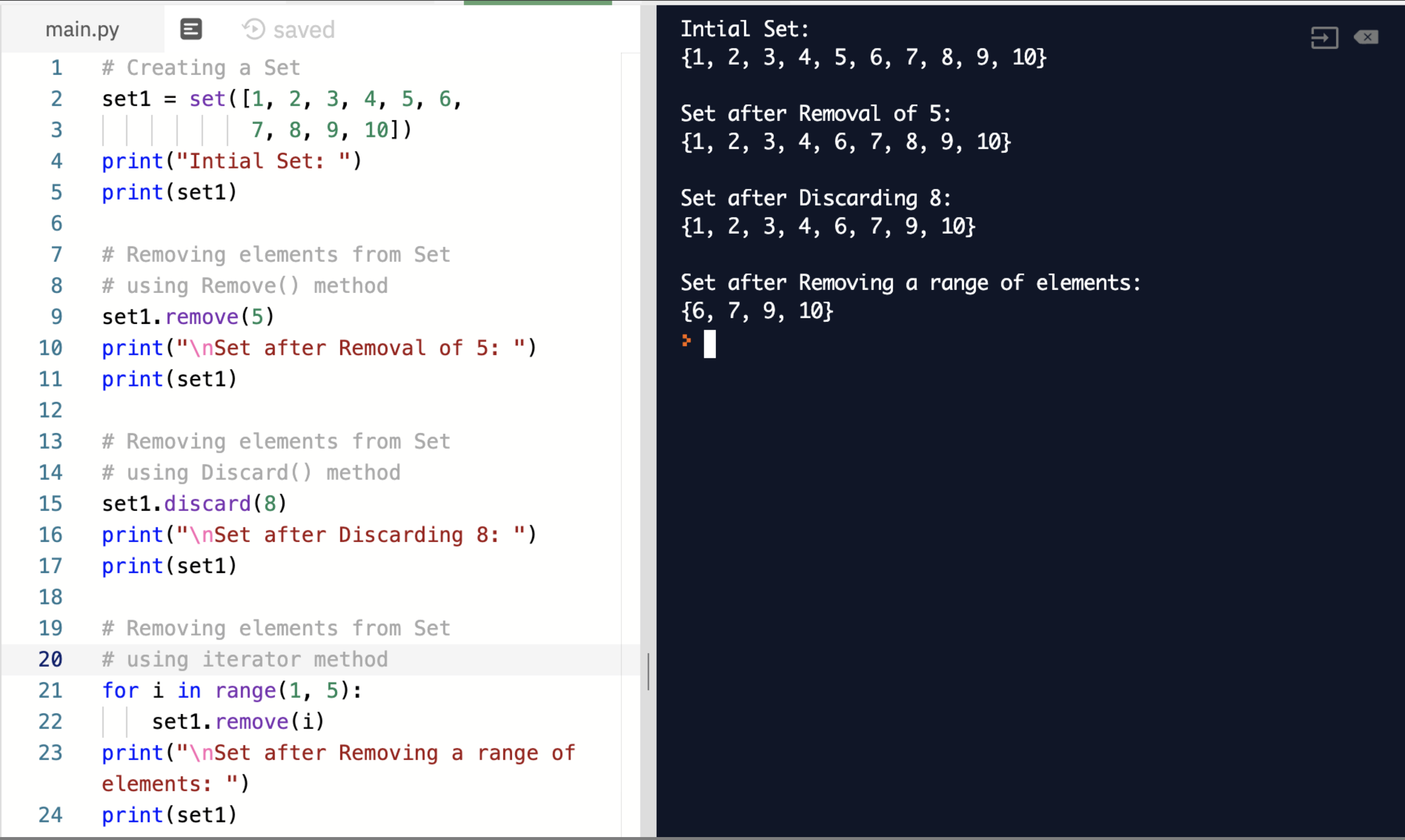


```python
# Creating a set
set1 = set(["COMP", "1819", "ADS"])
print("\nInitial set")
print(set1)

# Accessing element using
# for loop
print("\nElements of set: ")
for i in set1:
    print(i, end=" ")

# Checking the element
# using in keyword
print("Python" in set1)
```

```
Initial set
{'1819', 'COMP', 'ADS'}

Elements of set:
1819 COMP ADS False
```

Operation “in” and ”not in” for checking membership

Operations: remove



main.py saved

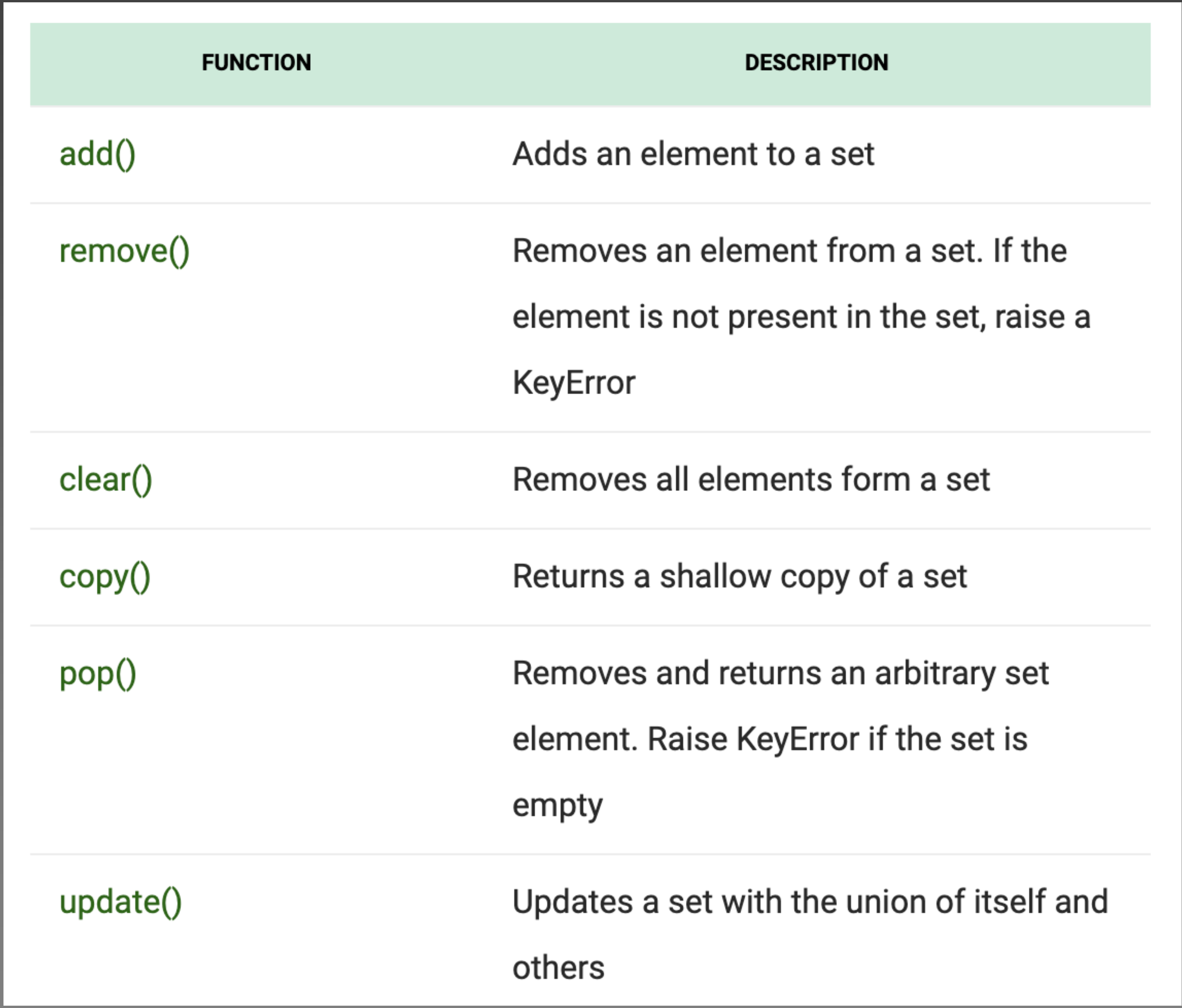
```python
# Creating a Set
set1 = set([1, 2, 3, 4, 5, 6,
            7, 8, 9, 10])
print("Intial Set: ")
print(set1)

# Removing elements from Set
# using Remove() method
set1.remove(5)
print("\nSet after Removal of 5: ")
print(set1)

# Removing elements from Set
# using Discard() method
set1.discard(8)
print("\nSet after Discarding 8: ")
print(set1)

# Removing elements from Set
# using iterator method
for i in range(1, 5):
    set1.remove(i)
print("\nSet after Removing a range of elements: ")
print(set1)
```

```
Intial Set:
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Set after Removal of 5:
{1, 2, 3, 4, 6, 7, 8, 9, 10}

Set after Discarding 8:
{1, 2, 3, 4, 6, 7, 9, 10}

Set after Removing a range of elements:
{6, 7, 9, 10}
›
```

How about removing all items?

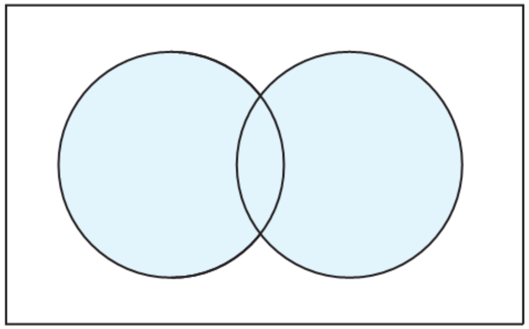
Operations

Missing?

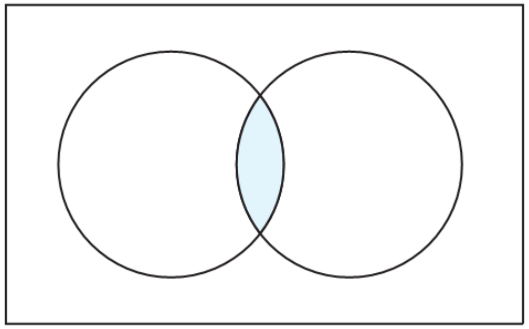


FUNCTION	DESCRIPTION
add()	Adds an element to a set
remove()	Removes an element from a set. If the element is not present in the set, raise a KeyError
clear()	Removes all elements form a set
copy()	Returns a shallow copy of a set
pop()	Removes and returns an arbitrary set element. Raise KeyError if the set is empty
update()	Updates a set with the union of itself and others

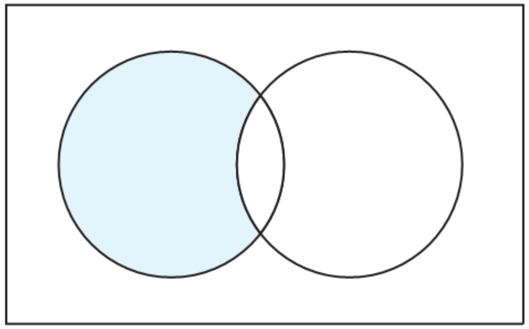
$A \cup B$ Union

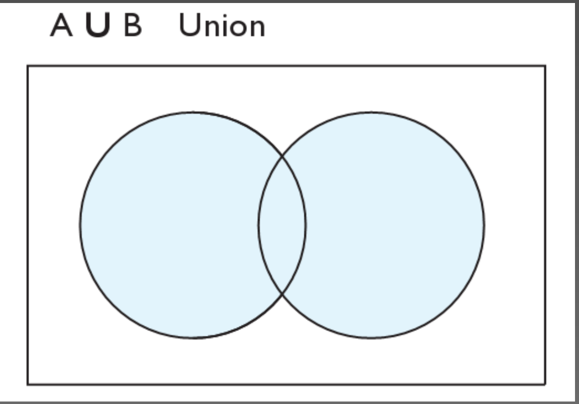


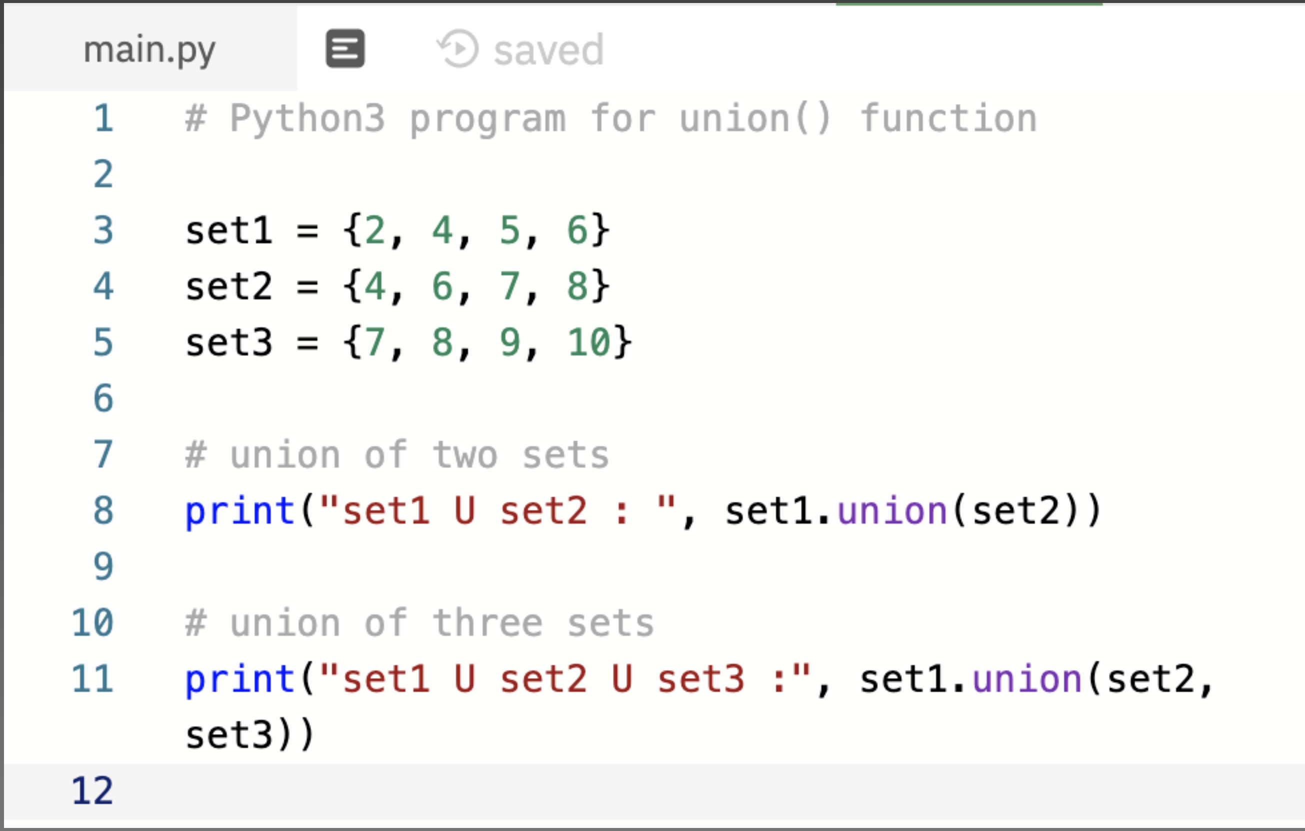
$A \cap B$ Intersection



$A - B$ Difference

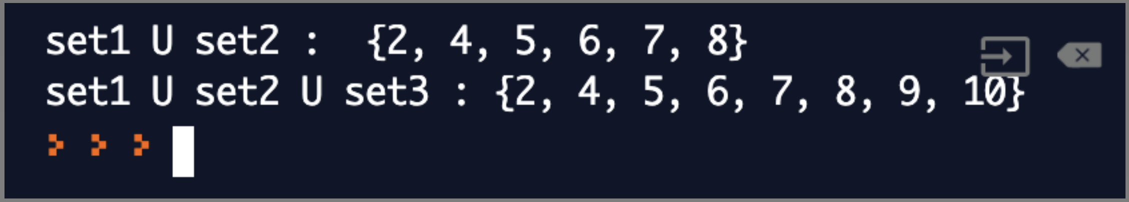




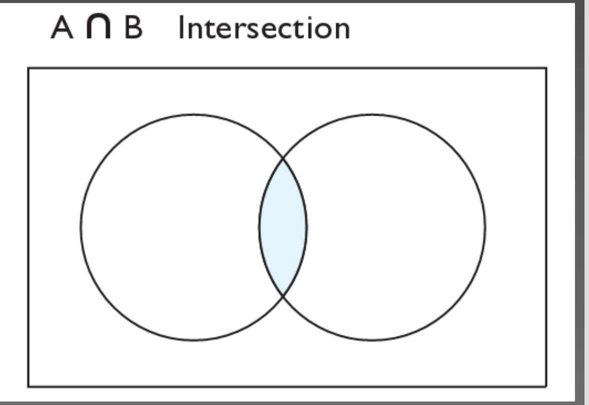


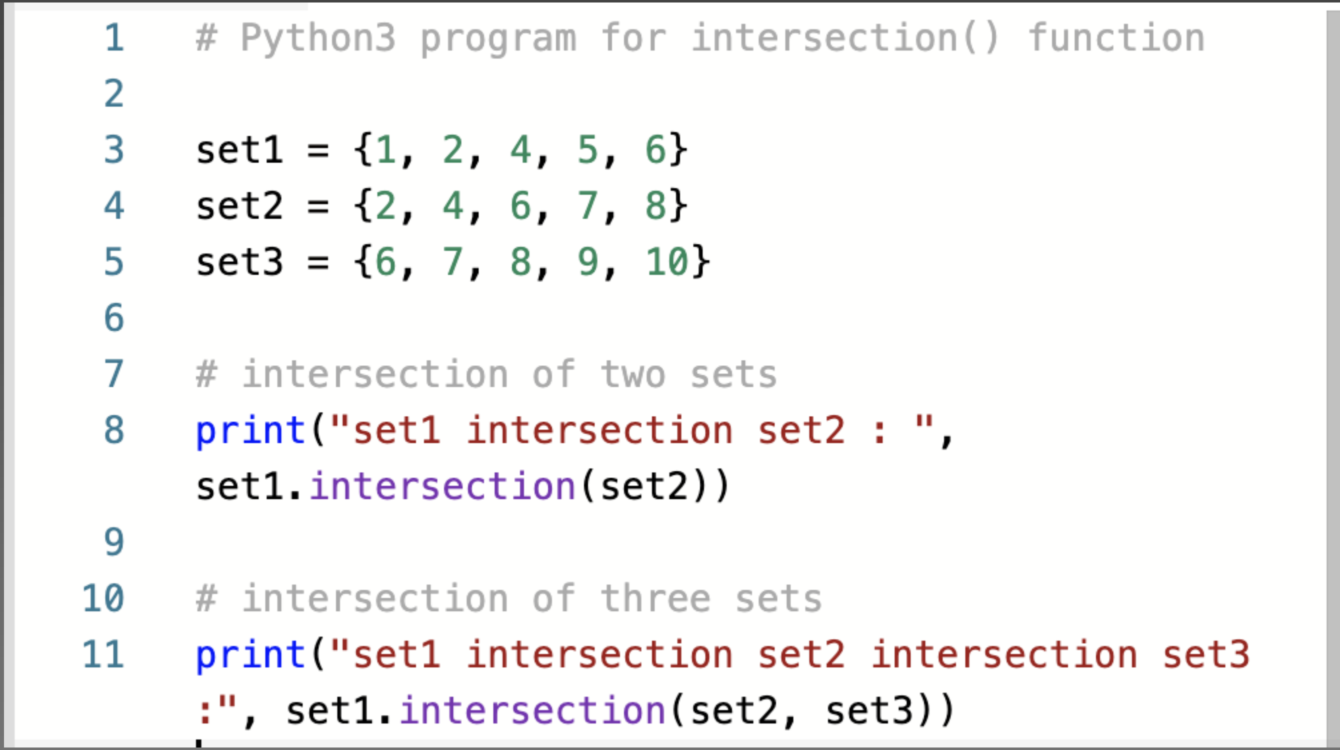
```python
# Python3 program for union() function

set1 = {2, 4, 5, 6}
set2 = {4, 6, 7, 8}
set3 = {7, 8, 9, 10}

# union of two sets
print("set1 U set2 : ", set1.union(set2))

# union of three sets
print("set1 U set2 U set3 :", set1.union(set2,
set3))

```

What is the output?



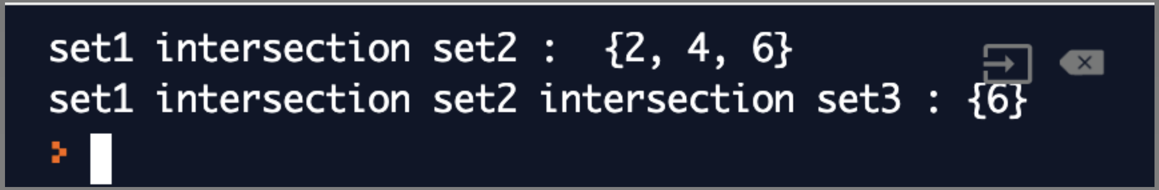
```
set1 U set2 :  {2, 4, 5, 6, 7, 8}
set1 U set2 U set3 : {2, 4, 5, 6, 7, 8, 9, 10}
> > >
```



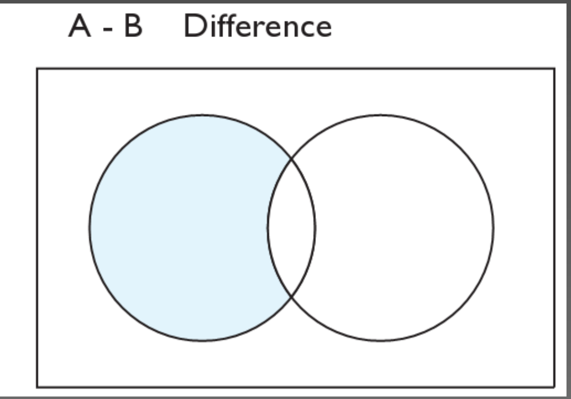


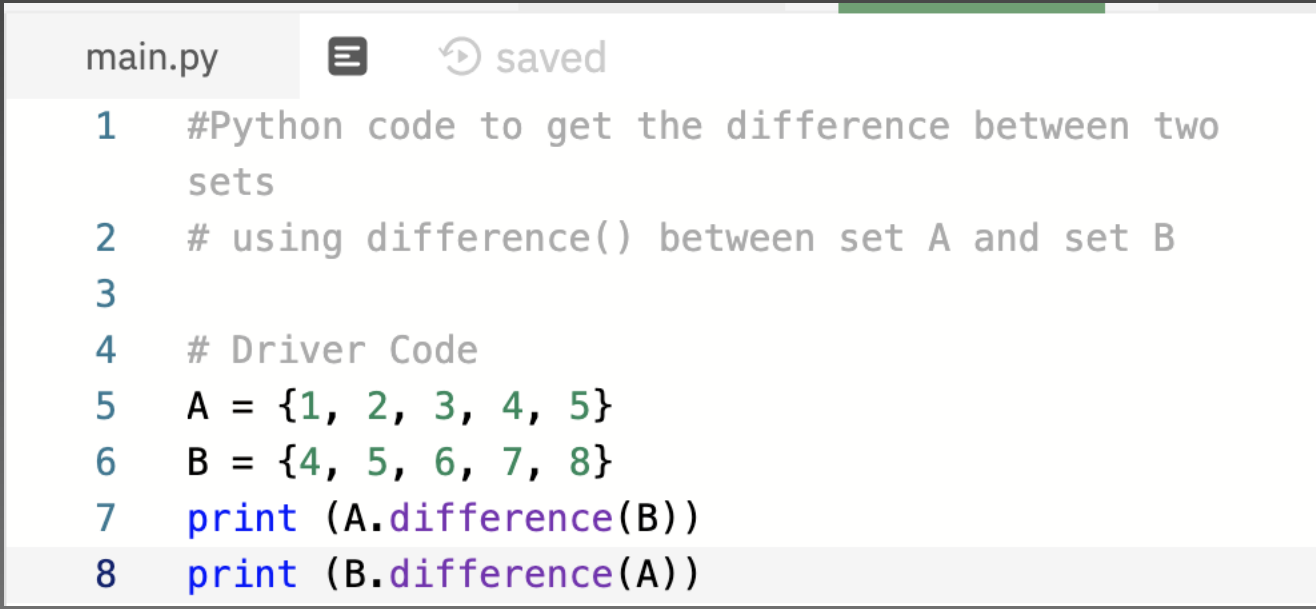
```python
# Python3 program for intersection() function

set1 = {1, 2, 4, 5, 6}
set2 = {2, 4, 6, 7, 8}
set3 = {6, 7, 8, 9, 10}

# intersection of two sets
print("set1 intersection set2 : ", set1.intersection(set2))

# intersection of three sets
print("set1 intersection set2 intersection set3 :", set1.intersection(set2, set3))
```

What is the output?

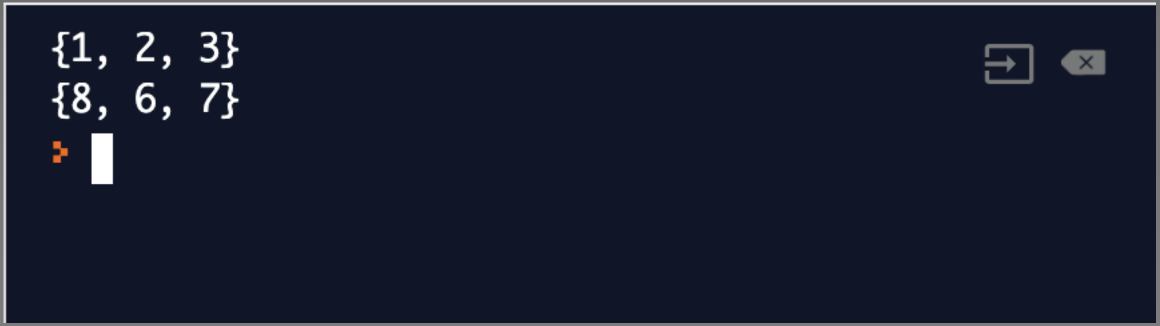


```
set1 intersection set2 :  {2, 4, 6}
set1 intersection set2 intersection set3 : {6}
>
```




main.py saved

```python
#Python code to get the difference between two sets
# using difference() between set A and set B

# Driver Code
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
print (A.difference(B))
print (B.difference(A))
```



```
{1, 2, 3}
{8, 6, 7}
>
```

What is the output?

Please read more different sets methods: isdisjoint(), issubset(), … from help(set)

Set ADT

S.add(e):	Add element e to the set. This has no effect if the set already contains e.
S.discard(e):	Remove element e from the set, if present. This has no effect if the set does not contain e.
e in S:	Return True if the set contains element e. In Python, this is implemented with the special __contains__ method.
len(S):	Return the number of elements in set S. In Python, this is implemented with the special method __len__.
iter(S):	Generate an iteration of all elements of the set. In Python, this is implemented with the special method __iter__.
S.remove(e):	Remove element e from the set. If the set does not contain e, raise a KeyError.
S.pop():	Remove and return an arbitrary element from the set. If the set is empty, raise a KeyError.
S.clear():	Remove all elements from the set.

Boolean Set Operations

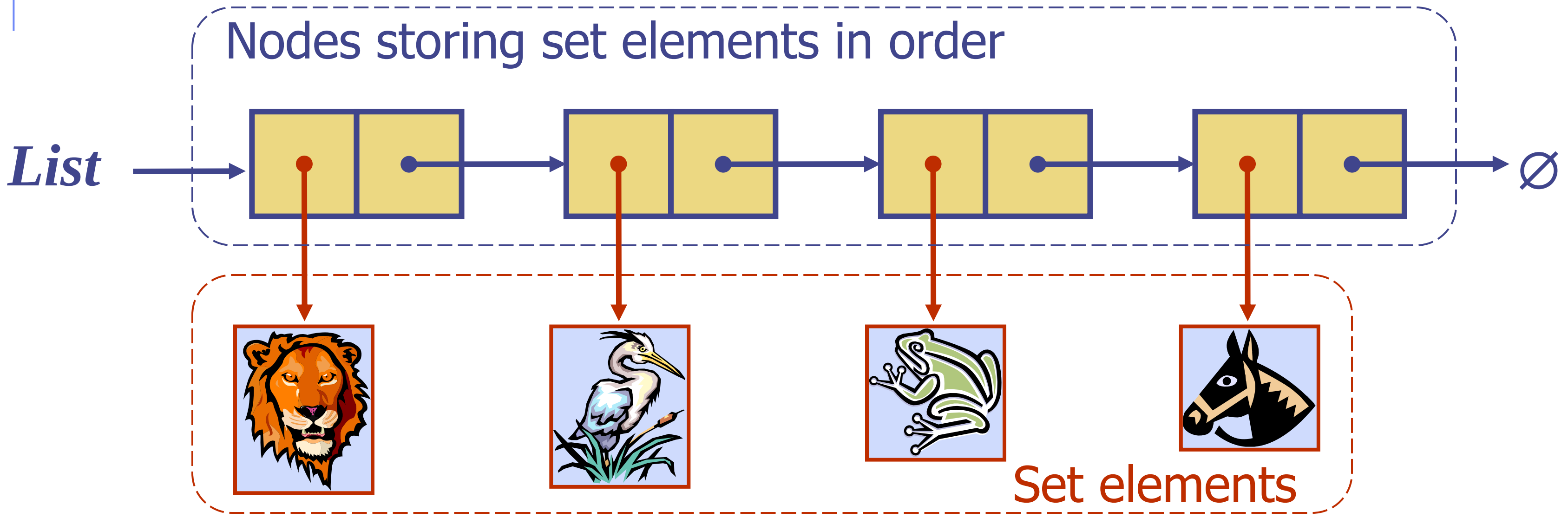
S == T:	Return True if sets S and T have identical contents.
S != T:	Return True if sets S and T are not equivalent.
S <= T:	Return True if set S is a subset of set T.
S < T:	Return True if set S is a *proper* subset of set T.
S >= T:	Return True if set S is a superset of set T.
S > T:	Return True if set S is a *proper* superset of set T.
S.isdisjoint(T):	Return True if sets S and T have no common elements.

Set Update Operations

S | T:	Return a new set representing the union of sets S and T.
S |= T:	Update set S to be the union of S and set T.
S & T:	Return a new set representing the intersection of sets S and T.
S &= T:	Update set S to be the intersection of S and set T.
S ^ T:	Return a new set representing the symmetric difference of sets S and T, that is, a set of elements that are in precisely one of S or T.
S ^= T:	Update set S to become the symmetric difference of itself and set T.
S − T:	Return a new set containing elements in S but not T.
S −= T:	Update set S to remove all common elements with set T.

Storing a Set in a List

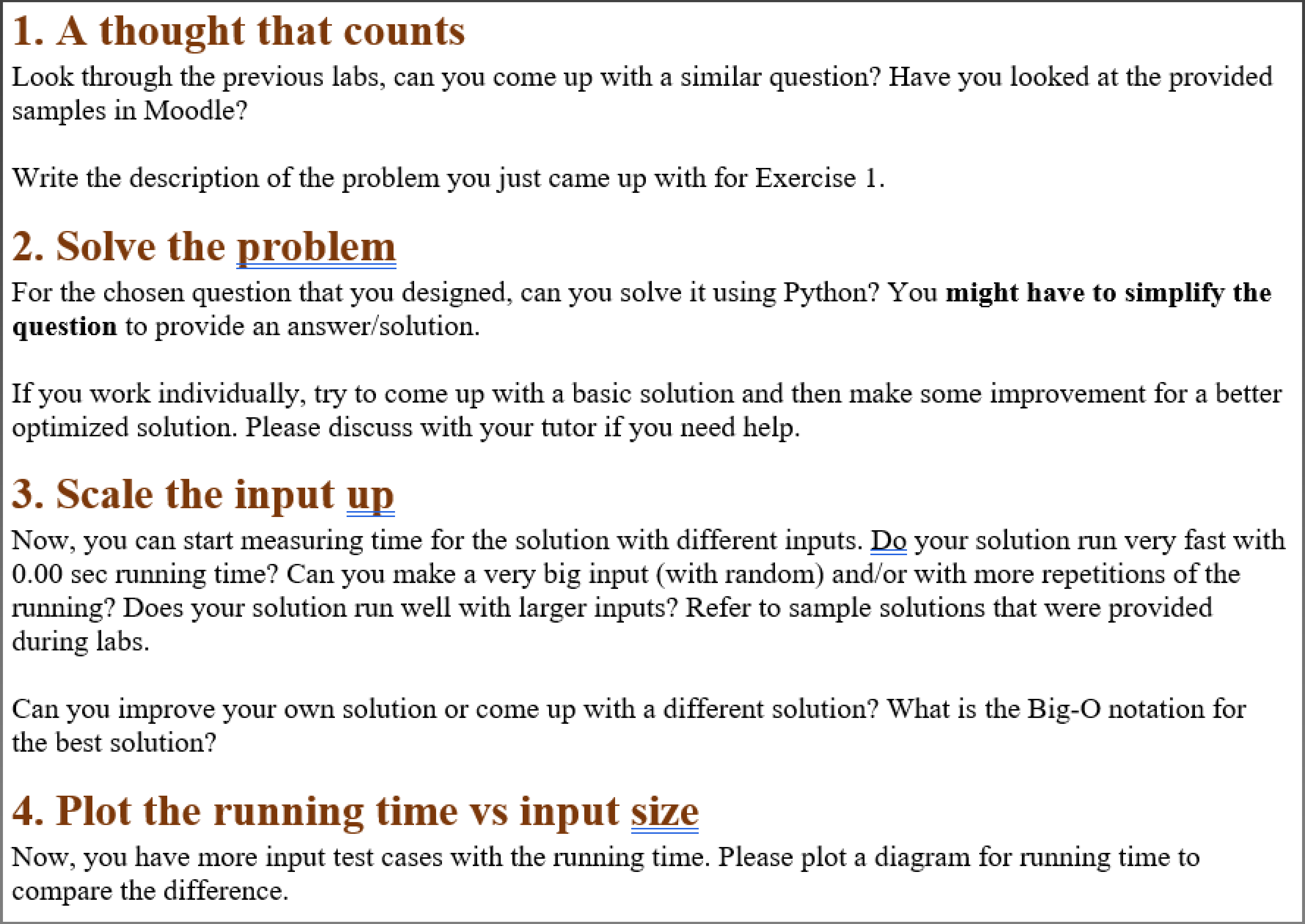
- We can implement a set with a list
- Elements are stored sorted according to some canonical ordering
- The space used is $O(n)$



Complexity Analysis of the List Implementations of Sets

- The list implementations of sets require little programmer effort
  - Unfortunately, they do not perform well
- Basic accessing methods must perform a linear search of the underlying list
  - Each basic accessing method is O(*n*)

Quick revisions: Lecture 01-07



1. A thought that counts

Look through the previous labs, can you come up with a similar question? Have you looked at the provided samples in Moodle?

Write the description of the problem you just came up with for Exercise 1.

2. Solve the problem

For the chosen question that you designed, can you solve it using Python? You **might have to simplify the question** to provide an answer/solution.

If you work individually, try to come up with a basic solution and then make some improvement for a better optimized solution. Please discuss with your tutor if you need help.

3. Scale the input up

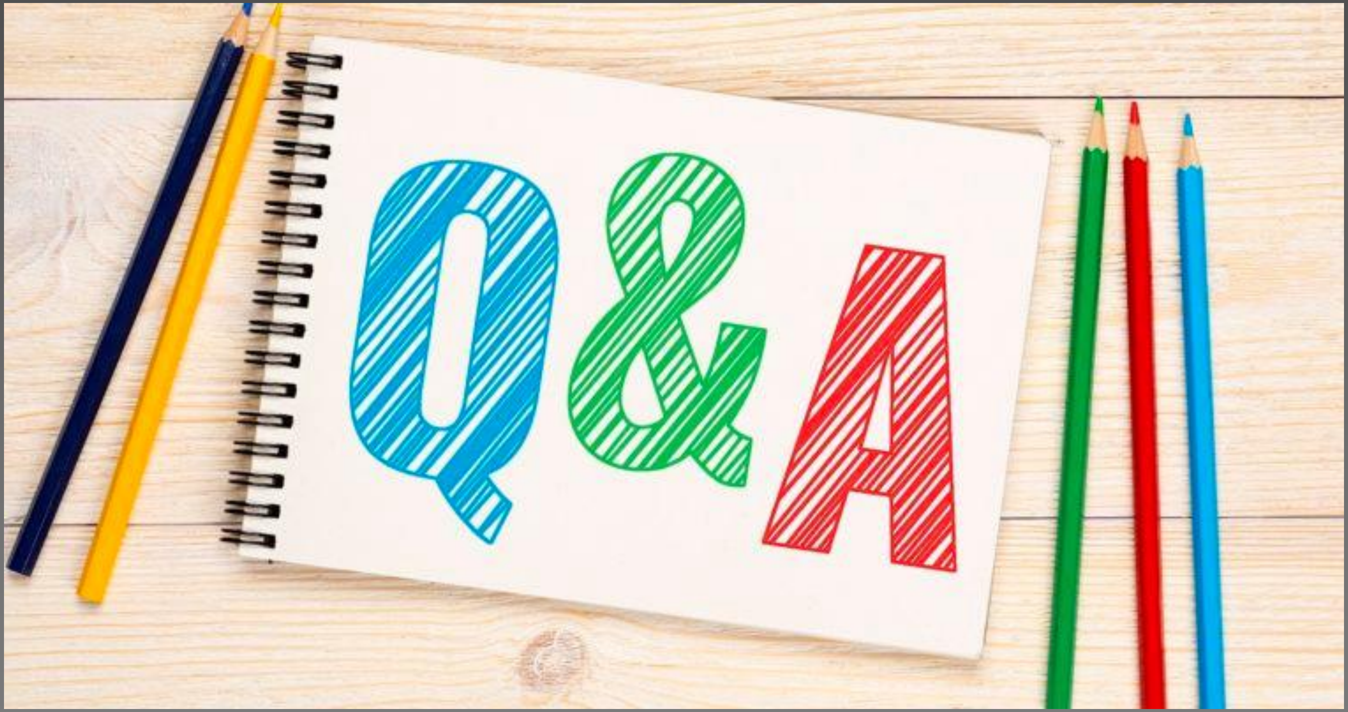
Now, you can start measuring time for the solution with different inputs. Do your solution run very fast with 0.00 sec running time? Can you make a very big input (with random) and/or with more repetitions of the running? Does your solution run well with larger inputs? Refer to sample solutions that were provided during labs.

Can you improve your own solution or come up with a different solution? What is the Big-O notation for the best solution?

4. Plot the running time vs input size

Now, you have more input test cases with the running time. Please plot a diagram for running time to compare the difference.

CW Q&A

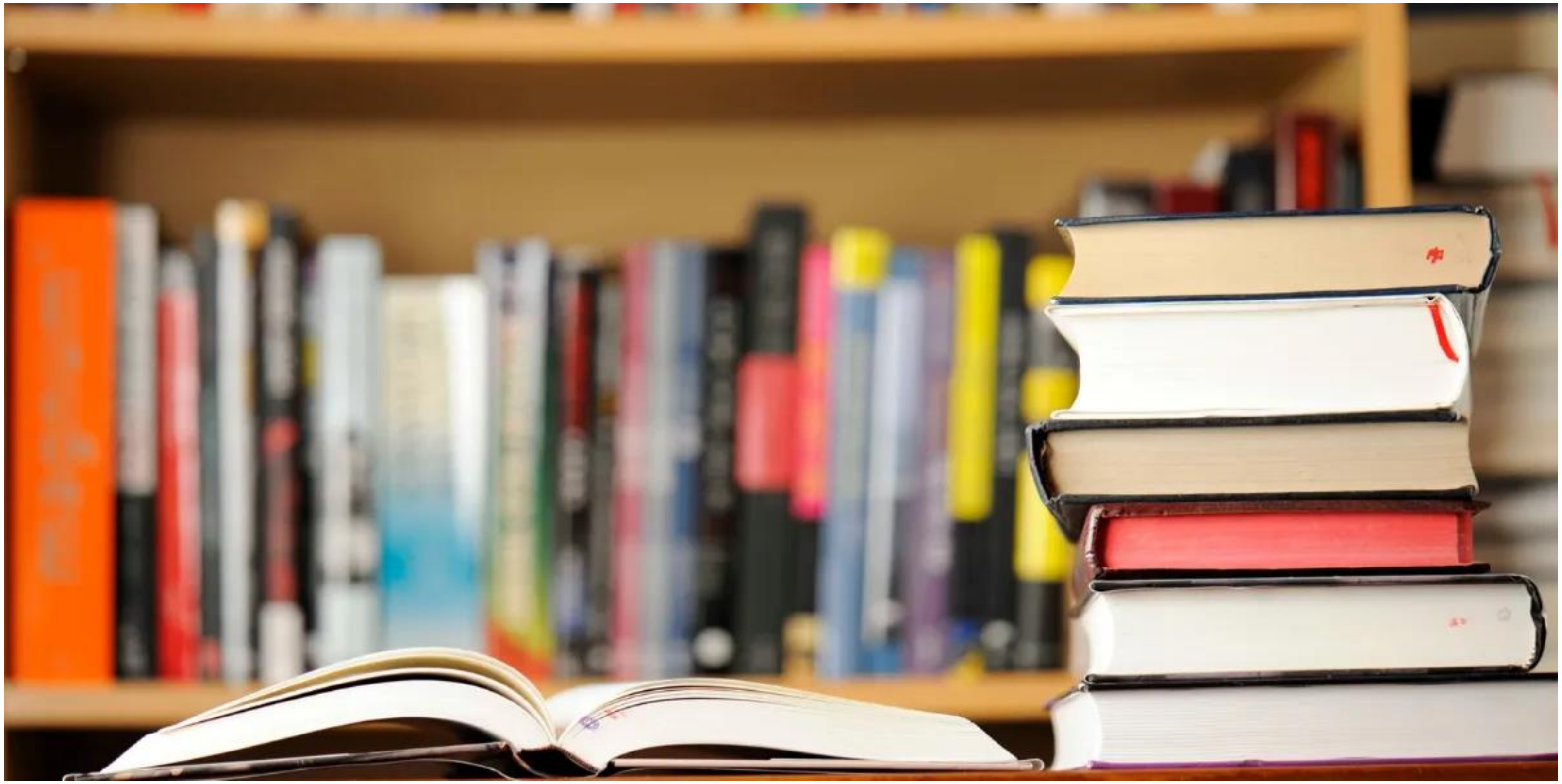


Quick overview

- Python Sets
- Sets ADT

Extra reading

- Hash Tables
- Skip lists



Revision for CW