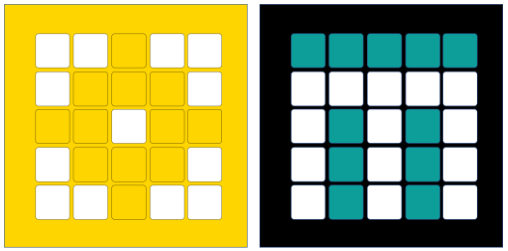
PRIME LESSONS

By the Makers of EV3Lessons



DICTIONARIES AND SETS

BY SANJAY AND ARVIND SESHAN

LESSON OBJECTIVES

- Learn to create and use dictionaries and sets

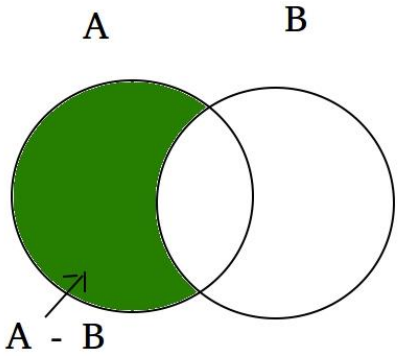
SETS

- Similar to lists
- Stores a set of items
- All items are unique and unordered
  - You can only place one of any item in a set
  - There is no order to a set (even if you entered in the items in a certain order)
- Sets are items in `{a, b,...}` brackets
- You can add to a set using the add method

```
>> s1 = set() # do not use {} to initialize empty set
>> s1.add(5) # add to a set
>> s1.add(2)
>> print(s1)
{2, 5}
>> s2 = {1, 2, 4, 4, "hello"} # define set
>> print(s2) # note only one 4 is below
{1, 2, 4, 'hello'}
```

MORE ON SETS

- You can find the difference, intersection, union, etc. between sets
- If you try to add a list to a set, or any other mutable type, the program will crash
- In general, it is much faster to do lookups on set than on a list due to something called hashing




```
>> print(s2.difference(s1))
{'hello'}
>> print(s2.intersection(s1))
{1, 2, 4}
>> s2.add([4,1])
TypeError: unhashable type: 'list'
```

SET METHODS

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)

Highlighted ones are most important

SET METHODS CONT.

isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

COPYING SETS

- Just like 1d lists, use the copy method

```
s1 = {1, 2, 3}

s2 = s1.copy()
```

- Sets are also mutable, like lists, so you need to be careful when doing something like s1=s2

DICTIONARIES

- Think like an English dictionary
  - Matches something to a definition
- Defined using {} braces and : colons
  - Format for each element is `item:definition` → typically called `key:value`
  - `d = {"hello":"a greeting", "red":"a color"}`
- The keys/items (e.g. "hello") must be unique, but many keys can have the same definition
- Keys can be any immutable data type (e.g. int, str)
- Values/definitions can be anything, (e.g. int, list, None)
- Use `d2 = d.copy()` to copy a dictionary (dicts are mutable)

DICTIONARY METHODS

The highlighted ones are the most important

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

GETTING A VALUE

- Sets:

```
s = {1, 2, 3}
3 in s # True
"hi" in s # False
```

- Dicts:

```
d = {"a":1,"b":2}
d["a"] == 1 # True
d["b"] == 2 # True
```

  - Addressed similar to a list → use [] brackets next to dict variable containing a key to get the "value"

CHALLENGE

- Translate a handful of Spanish words to English and print the result
- Hola → hello
- Rojo → red
- Naranja → orange
- Verde → green

CHALLENGE SOLUTION

```
d = {"hola":"hello", "rojo":"red", "naranja":"orange", "verde":"green"}

data = "hola"
print(d[data]) # hello

data = "naranja"
print(d[data]) # orange
```

REVIEW

- List → stores values `[1, 2, 2, 3, "hello"]` (mutable)
- Tuple → stores values `(1, 2, 2, 3, "hello")` (immutable)
- Set → stores unique values `{1, 2, 3, "hello"}` (mutable, but elements must be immutable)
- Dictionary → stores values that can be indexed with a key `{1:"a", 2:"b"}` (mutable, but keys must be immutable)

CREDITS

- This lesson was created by Sanjay and Arvind Seshan for Prime Lessons
- More lessons are available at www.primelessons.org