

Sets

Date: Oct 14, 2024

Sets

Sets

Unordered collection of immutable (integer, float, boolean, string) values

Same value can only be stored once

Sets are an **unordered** collection of **immutable** values.

- no duplicates
- mutable

Note (PRO): Sets are more efficient than lists because they are only allowed to contain immutable values.

This is because sets store items in memory using hashing and storing immutable values is much more efficient for look up.

Note (CON): Sets are unordered so therefore you cannot index into them or slice them based on index start and end positions.

What operations can we do on sets that are the same as lists?

- `len()`
- `in` operator
- iterate/traverse
- accumulator pattern

Syntax: Option 1

Curly braces {}

Example:

```
years = {1995, 2020, 1992, 1920, 1776}
years = {1995, 2020, 1995, 1992, 1920, 2020, 1776}
=> years will be {1995, 2020, 1992, 1920, 1776}
```

Duplicated entries get removed

Syntax: Option 2

`set(a)` where *a* is a data type that can be iterated over

Example:

```
word = set('helloworld')
=> word will be {'h', 'e', 'l', 'o', 'w', 'r', 'd'}
items = set(['couch', 'pillow', 'couch', 'lamp', 'lamp', 'clock'])
=> items will be {'couch', 'pillow', 'lamp', 'clock'}
mapping = set({'a': 2, 'b': 5, 'c': 6, 'd': 7})
=> mapping will be {'a', 'b', 'c', 'd'}
```

Duplicated entries get removed

To create a set, one option is to use `{}` and put the values you want in the set inside the `{}` separated by commas.

Note: If you try to add a value multiple times, the duplicate values will be removed because sets can only have unique values!

Another way you create a set is using the built-in function `set()` which takes in an iterable data type as an argument.

Examples of iterable data types to pass into the `set()` function to create a set:

- string -> set of unique characters
- list -> set of unique values
- dictionary -> set of (unique) keys

Note: Dictionary keys are already unique, so calling `set()` on a dictionary just gives you a set of the keys in the dictionary!

Empty Set

Has to be:

```
new_set = set()
```

Cannot be:

```
new_set = {}
```

`{}` creates a new dictionary, not a set

Keep in mind that if you were to create an empty set and tried to do `new_set = {}` the computer is going to think that you are creating an empty dictionary in Python!

SO in order to create an empty set in Python, do `new_set = set()`!

Mutation: **Add an item**

`set.add(x)`

- Add element `x` to set
- Changes set but does NOT return set (returns `None`)

```
years = {1995, 2020, 1992, 1920, 1776}
years.add(2023)
=> returns NONE, not the set
print(years)
=> {1995, 2020, 1992, 2023, 1920, 1776}
```

```
new_set = years.add(2023)
print(new_set)
=> None
```

To add an item to a set, use `.add()`!

Note that this function returns `None` because the job of this function is to mutate the set, so it doesn't really need to return anything!

Mutation: **Delete an item**

`set.remove(x)`

- `x` is an item in set
- If `x` does not exist in set, will raise `KeyError`
- Changes set but does NOT return set (returns `None`)

```
years = {1995, 2020, 1992, 1920, 1776}
years.remove(2020)
=> returns NONE, not the set
print(years)
=> {1995, 1992, 1920, 1776}
```

To remove an item from a set, use `.remove()`!

Note that this function returns `None` because the job of this function is to mutate the set, so it doesn't really need to return anything!

Mutation: **Delete a random item**

`set.pop()`

- Removes arbitrary item from set, returns item
- If set is empty, will raise `KeyError`

Why?

Could be useful if you have only one item left in set!

or

If you want to use every item in the set one at a time (like drawing from a hat).

To remove an item at random from a set use `.pop()`!

Because there are cases we probably want that element that gets removed, it makes sense that this function returns that item!

Calculate: **Set Math**

`s1.union(s2)`

- Combines two sets

`s1.intersection(s2)`

- Finds overlap between two sets

`s1.difference(s2)`

- Removes all items in s2 from s1

Note: none of these functions mutate the set

If we want to get information about how two different sets are related to each other, we can use these built-in methods that allow us to do “math” on these sets!

- `union()`
- `intersection()`
- `difference()`

You’ve probably seen these words before like union, intersection, and difference when it comes to VENN DIAGRAMS!

Review: **Sets**

What are they?

Unordered collection of immutable (integer, float, boolean, string) values

Same value can only be stored once

Can find `len()`, use `in` operator, `iterate`, `accumulate`

Set functions: `add()`, `remove()`, `pop()`

Why are they important?

Contain data that can **only have one entry** per value

Identify / remove duplicates

Sets are an unordered collection of unique immutable values!

To mutate the set, we can use built-in set methods like `add()`, `remove()`, `pop()`!

If we want to store data that is unique (and order does not matter), we should use a set!

Using the `set()` method helps us to easily identify and remove duplicates in iterable data types!

TODOs

- ☐ Project 1 (EXTRA CREDIT) - Multiplayer Blackjack!
 - Due Wednesday at 4:00pm (end of my OH that day)!
- ☐ Project 2 (pt. 1) - Search Engine
 - Due next Monday at 11:59pm!
- ☐ HW 7 - 2D Lists
 - Due Friday!
- ☐ Quiz 7 - Dictionaries, Sets
 - This Friday!

- Midterm Exam 2 - less than 2 weeks from today!
 - It will cover EVERYTHING we have learned so far.

- STEP Internship @ Google
 - Applications open Sep 30, 2024 and close on Oct 25, 2024 !!
 - For 1st year and 2nd year students, applications will be available at g.co/jobs/step!
 - All of the available intern opportunities can be found at google.com/students!