

# Sets

C-START Python PD Workshop

# Review: What Data Types Do We Know?

- Integers (e.g., 19)

# Review: What Data Types Do We Know?

- Integers (e.g., 19)
- Floating point numbers (e.g., 19.5)

# Review: What Data Types Do We Know?

- Integers (e.g., 19)
- Floating point numbers (e.g., 19.5)
- Booleans (`True` and `False`)

# Review: What Data Types Do We Know?

- Integers (e.g., 19)
- Floating point numbers (e.g., 19.5)
- Booleans (`True` and `False`)
- Strings (e.g., `"Hello"`)

# Review: What Data Types Do We Know?

- Integers (e.g., 19)
- Floating point numbers (e.g., 19.5)
- Booleans (`True` and `False`)
- Strings (e.g., `"Hello"`)
- Lists (e.g., `[12, 13, ["Hello", "There"]]`)  
**Ordered** and **potentially non-unique** collections of data

Sets are like lists in that they are a container for multiple objects, however, they are **unordered** and cannot contain multiple copies of the same object. We can specify set literals in Python using squirly braces.

```
my_set = {"a", 12, "computers", "cheese"}
```

Sets are like lists in that they are a container for multiple objects, however, they are **unordered** and cannot contain multiple copies of the same object. We can specify set literals in Python using squirly braces.

```
my_set = {"a", 12, "computers", "cheese"}
```

We can check if an item exists in a set using the **in** operator.

- What will `"cheese" in my_set` evaluate to?
- What will `42 in my_set` evaluate to?



## Example of Using a Set

`s.add(item)` will add `item` to the set `s` if `item` is not already in `s`.

## Example of Using a Set

`s.add(item)` will add `item` to the set `s` if `item` is not already in `s`.

```
food = set()           # this creates an empty set
while True:
    line = input('Give a tasty food, blank to end: ')
    if line == '':
        break          # exits the while loop
    food.add(line)
print('You think', len(food), 'foods are tasty')
```

**Note:** Code is on website for easy copy-paste.

# Investigate: What types of data can a set store?

Sets may only store **hashable** data types.

Try to answer each of the following questions by attempting to create an example in the interactive interpreter and seeing if you get an error.

- 1 Can a set contain floating point numbers?
- 2 Can a set contain booleans?
- 3 Can a set contain lists?
- 4 Can a list contain sets?
- 5 Can a set contain sets?

**When should you choose a set over a list?**

## When should you choose a set over a list?

- You should choose a set if you have data in which **order does not matter**, and **you require unique items in the structure**.

## When should you choose a set over a list?

- You should choose a set if you have data in which **order does not matter**, and **you require unique items in the structure**.
- You should choose a list if you have data in which **order matters**, and **you can have multiple copies of items in the structure**.

## When should you choose a set over a list?

- You should choose a set if you have data in which **order does not matter**, and **you require unique items in the structure**.
- You should choose a list if you have data in which **order matters**, and **you can have multiple copies of items in the structure**.

There are certainly plenty of exceptions to the rules presented above, however, for the scope of this workshop, you should be fine following these rules.

Suppose that  $a$  and  $b$  are sets. Then,

- $a - b$  is the **set difference** of  $a$  and  $b$ . This is the set of elements in  $a$  that are not in  $b$ .



Suppose that  $a$  and  $b$  are sets. Then,

- $a - b$  is the **set difference** of  $a$  and  $b$ . This is the set of elements in  $a$  that are not in  $b$ .
- $a \& b$  is the **set intersection** of  $a$  and  $b$ . This is the set of elements that both  $a$  and  $b$  have in common.

Suppose that  $a$  and  $b$  are sets. Then,

- $a - b$  is the **set difference** of  $a$  and  $b$ . This is the set of elements in  $a$  that are not in  $b$ .
- $a \& b$  is the **set intersection** of  $a$  and  $b$ . This is the set of elements that both  $a$  and  $b$  have in common.
- $a | b$  is the **set union** of  $a$  and  $b$ . This is the set of elements that either  $a$  or  $b$  or both has.

# Set Operations Example

```
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}
diff1 = a - b    # {1, 2}
diff2 = b - a    # {5, 6}
diff3 = a - a    # empty set
inter = a & b     # {3, 4}
union = a | b     # {1, 2, 3, 4, 5, 6}
```

## Practice: Set Operations

Suppose you have three children, all of them very picky eaters.

- Charlie says he won't eat fish or broccoli.
- Alice says she will only eat pasta, hot dogs, or fish.
- Mary says she won't eat pasta or salad.

Define the Python sets `charlie_wont`, `alice_will`, and `mary_wont`. See which of the following operations will compute what you can have for dinner tonight (`{"hot dogs"}`):

- `alice_will - charlie_wont - mary_wont`
- `alice_will - (charlie_wont & mary_wont)`
- `alice_will - charlie_wont | mary_wont`
- `alice_will - (charlie_wont | mary_wont)`

Lastly, see if you can come up with more of your own statements which compute what you can have for dinner tonight.