CITY
UNIVERSITY OF LONDON
EST 1894

STEM Digital Academy

School of Science & Technology

www.city.ac.uk

Set Operations and Processing Sets

Programming and Algorithms

Lecture by

Dr Daniil Osudin

```
n = 3
for i in range(1,n+1):
    print("Hello World!")
```

```
Hello World!
Hello World!
Hello World!
```

What will we Cover?

- Processing sets
- Set operations
- Writing simple programs incorporating the set data structure

Processing Sets I

Use an IN keyword to check if the value exists in a set

```
names = {"Sam", "Jo", "Adam", "Aaron", "Tamar"}
if "Jill" in names:
    print("Jill exists in names")
else:
    print("name not found")
```

```
name not found
```

Processing Sets II

Use a FOR loop to iterate over a set and print each element

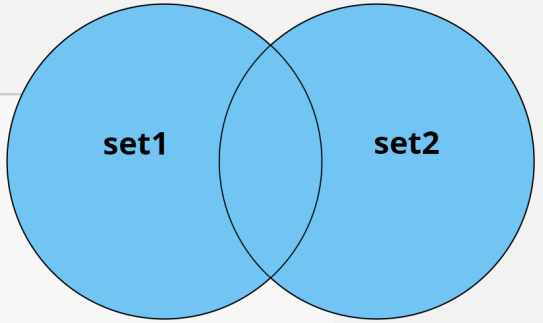
```
names = {"Sam", "Jo", "Adam", "Aaron", "Tamar"}
for n in names:
    print(n)
```

```
Sam
Adam
Tamar
Aaron
Jo
```

Set Operations I

`union()` – used to find all values from the sets, without duplicates

Alternatively, | operator can be used




```
set1 = {"red", "orange", "green", "blue", "purple"}
set2 = {"white", "green", "red", "black", "yellow"}
print(set1.union(set2))
print(set1 | set2)
```

```
{'white', 'black', 'orange', 'blue', 'yellow', 'purple', 'green', 'red'}
{'white', 'black', 'orange', 'blue', 'yellow', 'purple', 'green', 'red'}
```

CITY UNIVERSITY OF LONDON EST 1894

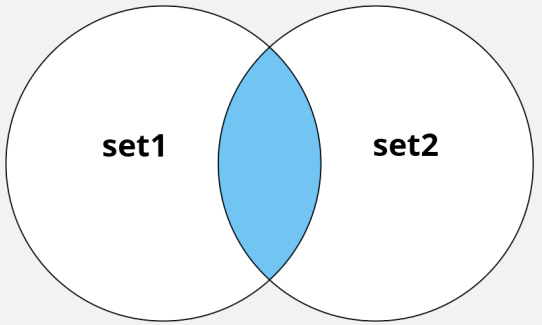
Set Operations II

`intersection()` – used to find all values common to both sets, without duplicates

Alternatively, `&` operator can be used

```
set1 = {"red", "orange", "green", "blue", "purple"}
set2 = {"white", "green", "red", "black", "yellow"}
print(set1.intersection(set2))
print(set1 & set2)
```

```
{'green', 'red'}
{'green', 'red'}
```



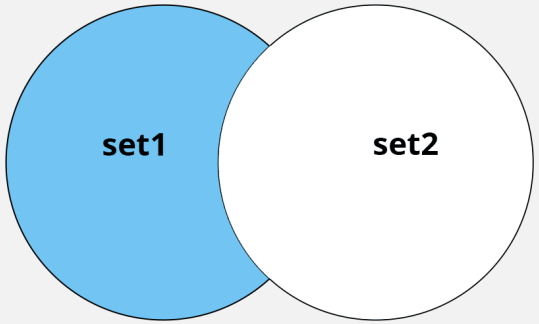
Set Operations III

`difference()` – used to find all values found in one set but not the other

Alternatively, − operator can be used

```
set1 = {"red", "orange", "green", "blue", "purple"}
set2 = {"white", "green", "red", "black", "yellow"}
print(set1.difference(set2))
print(set1 - set2)
```

```
{'orange', 'blue', 'purple'}
{'orange', 'blue', 'purple'}
```



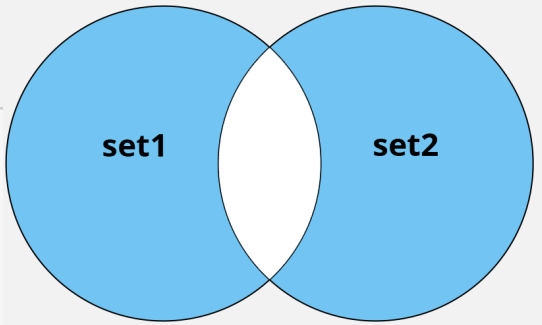
Set Operations IV

`symmetric_difference()` – used to find all values unique to either set

Alternatively, ^ operator can be used

```
set1 = {"red", "orange", "green", "blue", "purple"}
set2 = {"white", "green", "red", "black", "yellow"}
print(set1.symmetric_difference(set2))
print(set1 ^ set2)
```

```
{'white', 'orange', 'black', 'yellow', 'blue', 'purple'}
{'white', 'orange', 'black', 'yellow', 'blue', 'purple'}
```



CITY UNIVERSITY OF LONDON EST 1894

Try It Yourself

```
customer1 = {"oil paints", "lamp", "watch",
"frame", "kit-kat", "desk"}
customer2 = {"lamp", "monitor", "home-hub",
"desk", "canvas"}
```

Using the two sets given above, write a program which determines the following :

- Items bought by both `customer1` and `customer2`
- All items bought by `customer1` and `customer2`
- Items bought by `customer1` or `customer2` but not bought by both
- Items that `customer2` bought but not `customer1`
- Remaining items the `customer1` had after they returned the items matching with `customer2`