Sets

Table of Contents

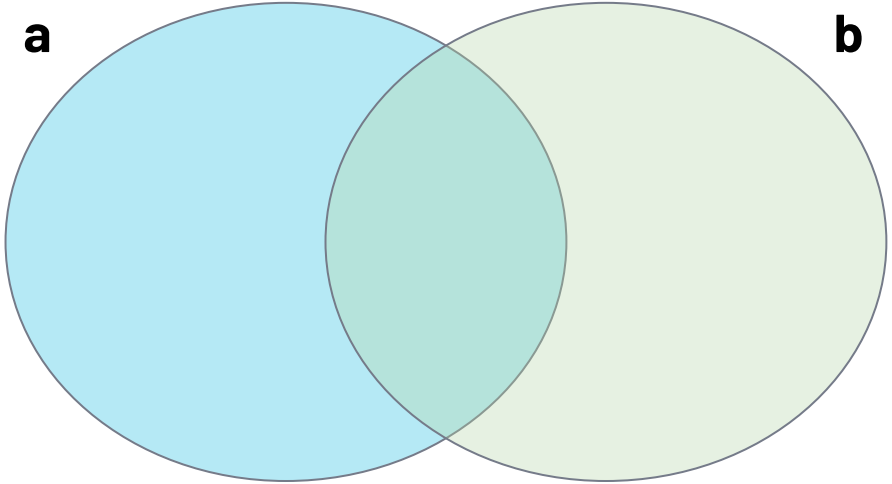
Definitions

```
fruit = {'Apple', 'Orange', 'Banana'}
```

```
set()
```

Definitions

- No repetition
- Math operations
  - union
  - intersection
  - difference
- Unordered elements



Creating a `set`

Creating a `set`

- We have two basic ways to create a set.

- `{}`
- `set()`

```
set_1 = {'red', 'blue', 'pink', 'red'}
colors = 'red', 'blue', 'pink', 'red'
set_2 = set(colors)
print(set_1)
```

```
{'blue', 'pink', 'red'}
```

Creating a `set`

- A `set` can be created by enclosing values, separated by commas, in curly braces 👉 `{}`.
- Another way to create a `set` is to call the `set()` function.

- `{}`
- `set()`

```
set_1 = {'red', 'blue', 'pink', 'red'}
colors = 'red', 'blue', 'pink', 'red'
set_2 = set(colors)
print(set_1)
print(set_2)
```

```
{'red', 'blue', 'pink'}
{'red', 'blue', 'pink'}
```

⚠️ different order from the previous slide

Creating a `set` (review of pre-class)

- Here is an example of creating an empty `set`:

input :

```
empty_set = set()

print(type(empty_set))

```

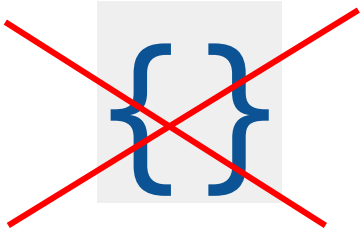
output :

```
<class 'set'>

```

Creating a `set`

- Creating an empty set



To create an empty `set`, you **can not** use 👉 `{}`. The only way to create an empty set is `set()` function.

Creating a `set` (review of pre-class)

```
flower_list = ['rose', 'violet', 'carnation', 'rose', 'orchid', 'rose', 'orchid']
flowerset = set(flower_list)
flowerlist = list(flowerset)

print(flowerset)
print(flowerlist)

```

What is the output? Try to figure out in your mind...

Creating a `set` (review of pre-class)

```
flower_list = ['rose', 'violet', 'carnation', 'rose', 'orchid', 'rose', 'orchid']
flowerset = set(flower_list)
flowerlist = list(flowerset)

print(flowerset)
print(flowerlist)
```

```
{'orchid', 'carnation', 'violet', 'rose'}
['orchid', 'carnation', 'violet', 'rose']
```

Creating a `set` (review of pre-class)

- **Task** :
  - Do these two sets give the same output and why?

```
a = {'carnation', 'orchid', 'rose', 'violet'}
```

👇👆

```
b = {'rose', 'orchid', 'rose', 'violet', 'carnation'}
```

Creating a `set`

- The Answer is : **`True`**

```
{'carnation', 'orchid', 'rose', 'violet'}

{'rose', 'orchid', 'rose', 'violet', 'carnation'}
```

Main Operations with Sets

Main Operations with `sets` (review)

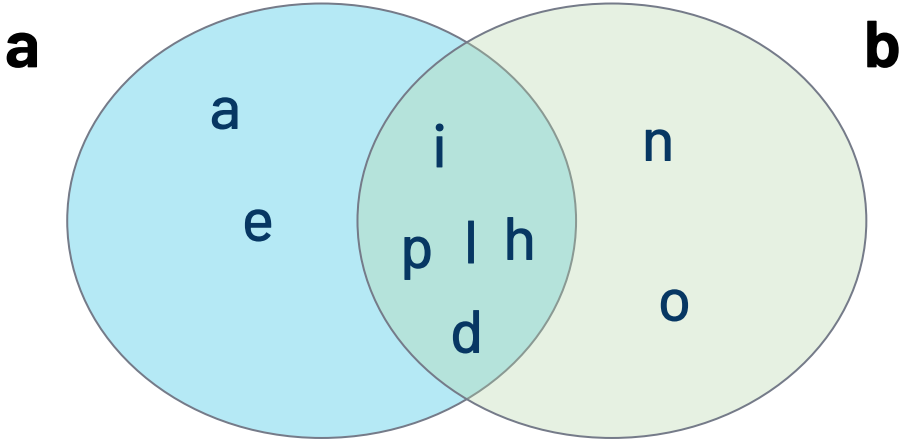
- The methods that can be used with `sets` :

- `.add()` : Adds a new item to the set.
- `.remove()` : Allows us to delete an item.
- `.intersection()` : Returns the intersection of two sets.
- `.union()` : Returns the unification of two sets.
- `.difference()` : Gets the difference of two sets.

Main Operations with sets

- Let's take a look these two sets below :

```
a = set('philadelphia')
b = set('dolphin')
```

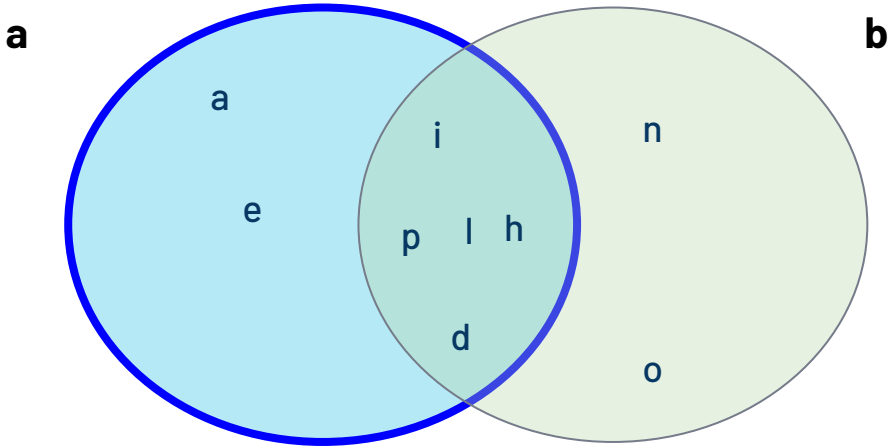


Main Operations with `sets`

- Let's take a look these two sets below :

```
a = set('philadelphia')
print(a)
```

```
{'a', 'e', 'i', 'd', 'l', 'p', 'h'}
```

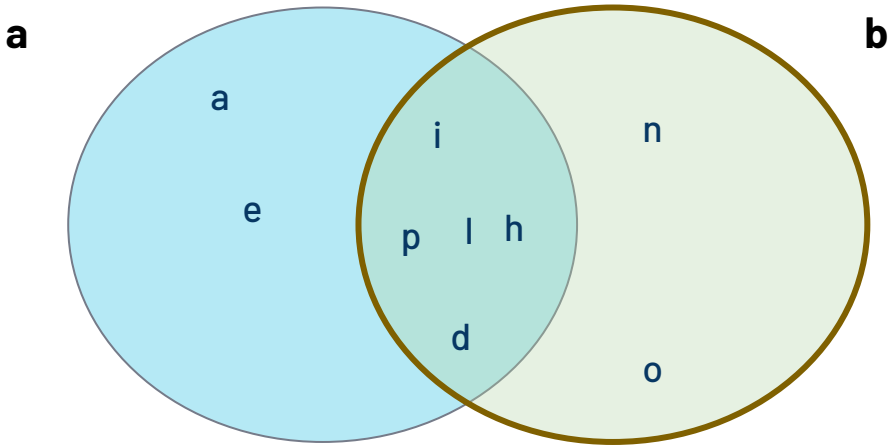


Main Operations with sets

- Let's take a look these two sets below :

```
b = set('dolphin')
print(b)
```

```
{'d', 'l', 'o', 'p', 'n', 'i', 'h'}
```



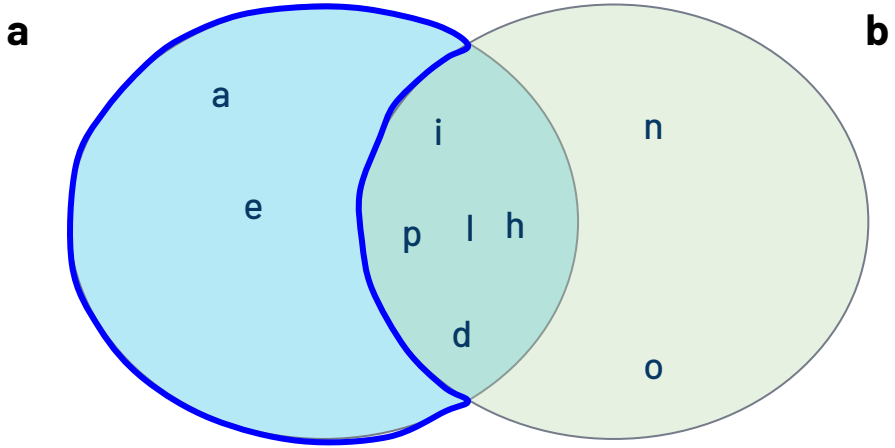
Main Operations with `sets`

- Basic `set` operations :

`.difference(arg)`

```
print(a - b)
print(a.difference(b))
```

```
{'a', 'e'}
```



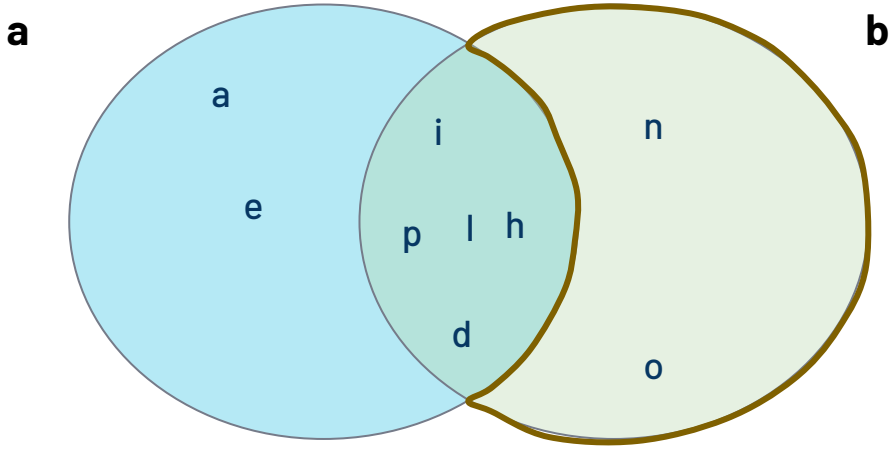
Main Operations with sets

- Basic `set` operations :

`.difference(arg)`

```
print(b - a)
print(b.difference(a))
```

```
{'n', 'o'}
```



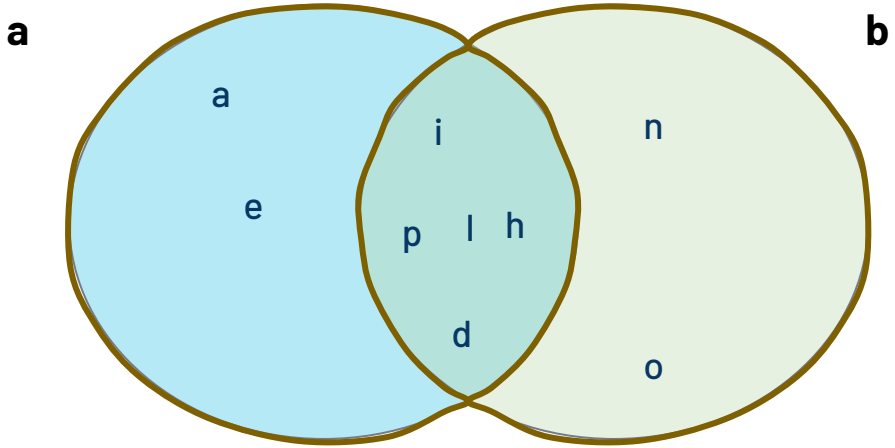
Main Operations with sets

- Basic set operations :

.union(arg)

```
print(a | b)
print(a.union(b))
```

```
{'p', 'h', 'i', 'l', 'd', 'o', 'n', 'a', 'e'}
```



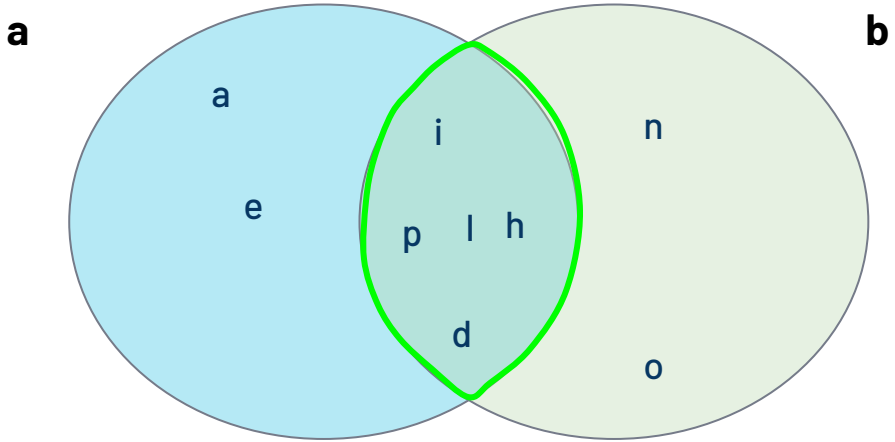
Main Operations with sets

- Basic set operations :

```
.intersection(arg)
```

```
print(a & b)
print(a.intersection(b))
```

```
{'p', 'h', 'i', 'l', 'd'}
```



Creating a `set`

- **Task** :

  Given a `list`, create a `set` to select and print the **unique** elements of the it.

```
given_list = [1, 2, 3, 3, 3, 3, 4, 4, 5, 5]
```

Creating a `set`

- **The code might be like** :

```
given_list = [1, 2, 3, 3, 3, 3, 4, 4, 5, 5]

unique = set(given_list)

print(unique)
```

```
{1, 2, 3, 4, 5}
```

Discuss in-class! Could you do the same thing using only curly braces `{}` instead of `set()` function?

Creating a `set`

- **Task** :

  -Create two `sets` of string data from the capitals of the **USA** and **New Zealand**. (**e.g**: `'Madrid'` → `convert into a set`)

  -Perform all set operations.

  - Intersection
  - Union
  - Difference

Creating a `set`

- **The code might be like** :

```
usa_capt = set('Washington')
nz_capt = set('Wellington')

print(usa_capt)
print(nz_capt)
```

```
{'h', 'W', 'a', 'o', 's', 'n', 'g', 'i', 't'}
{'W', 'o', 'l', 'e', 'n', 'g', 'i', 't'}
```

Creating a `set`

- **The code might be like** :

```
usa_capt = set('Washington')
nz_capt = set('Wellington')

print(usa_capt - nz_capt)
print(usa_capt.difference(nz_capt))
```

```
{'s', 'h', 'a'}
{'s', 'h', 'a'}
```

Creating a `set`

- **The code might be like** :

```
usa_capt = set('Washington')
nz_capt = set('Wellington')

print(nz_capt - usa_capt)
print(nz_capt.difference(usa_capt))
```

```
{'l', 'e'}
{'l', 'e'}
```

Creating a `set`

- **The code might be like** :

```
usa_capt = set('Washington')
nz_capt = set('Wellington')

print(nz_capt & usa_capt)
print(nz_capt.intersection(usa_capt))
```

```
{'i', 'o', 'g', 'n', 't', 'W'}
{'i', 'o', 'g', 'n', 't', 'W'}
```

frozenset()

- **Frozen set is just an immutable version of a Python set:**

```
usa_capt = set('Washington')
dondur_capt = frozenset(usa_capt) #elements cannot be changed
print(dondur_capt)
dondur_capt.add('z')   ## ?
print(dondur_capt)
```

```
frozenset({'g', 's', 'i', 'o', 't', 'n', 'a', 'h', 'W'})

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-68-9e416fb52e55> in <module>()
      2 dondur_capt = frozenset(usa_capt) #elemanları değiştirlemeyen bir küme
      3 print(dondur_capt)
----> 4 dondur_capt.add('z')
      5 print(dondur_capt)

AttributeError: 'frozenset' object has no attribute 'add'
```

List	Tuple	Dict	Set
'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse' 'sort'	'count', 'index'	'clear', 'copy', 'fromkeys' 'get', 'items', 'keys', 'pop', 'popitem' 'setdefault 'update', 'values'	'add', 'clear', 'copy', 'difference''differenc e_update', 'discard', 'intersection''interse ction_update', 'isdisjoint', 'issubset', 'issuperset', 'pop', 'remove', 'symmetric_difference' , 'symmetric_difference_ update'union', 'update'

List	Tuple	Dict	Set
mutable	immutable	mutable	**mutable**
ordered	ordered	unordered	unordered
unhashable	hashable	unhashable	unhashable
iterable	iterable	no duplicate	iterable
duplicate	duplicate		no duplicate