enactus
University of Sheffield

CODE CREATORS
SHEFFIELD

2024 PYTHON BEGINNER COURSE

Lesson 7

SHALEV LAU

/in/shalevl

/mushroomhater07

Recap exercise

***Program* (completed in For-loop exercises):**

Ask the user to enter 10 numbers. At the end, tell them the mean average, **maximum and minimum of the values entered.**

Solution:

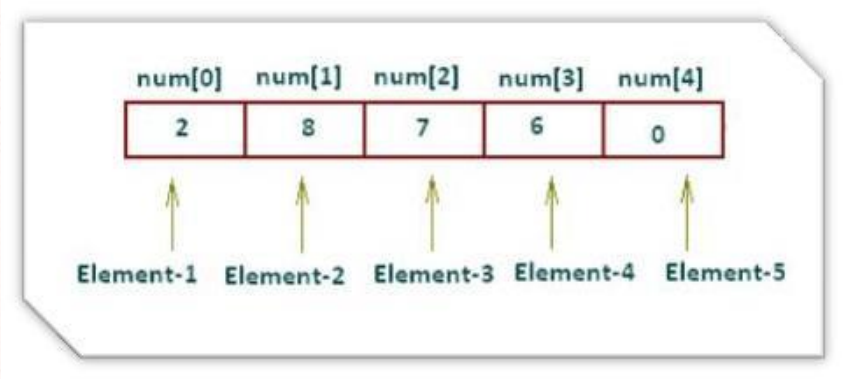
Use 10 different variables to store each number.

"Code Efficiency"

```
answer0 = int(input("enter 1st number"))
answer1 = int(input("enter 2nd number"))
answer2 = int(input("enter 3th number"))
answer3 = int(input("enter 4th number"))
answer4 = int(input("enter 5th number"))
answer5 = int(input("enter 6th number"))
answer6 = int(input("enter 7th number"))
answer7 = int(input("enter 8th number"))
answer8 = int(input("enter 9th number"))
answer9 = int(input("enter 10th number"))
```

List

Each element in the list shares the same name but a different **index**



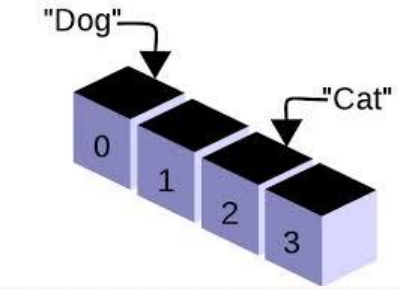
index is a number which indicates the position of the element in the array.

In Python, the first element of a list start from 0

List

We create a list by placing elements inside square brackets [ ], separated by commas.

```
myList = ["car", "bus", "train"]
```



Python list allow you to create a list with mixed DataType

```
mixed_types = [ ["Elton", 90], 95, "John", False ]
```

List

Integer (number)

string

Boolean (True or False)

List operation 1

```
myList = ["car", "taxi", "train"]
```

Access item in list:

```
List[index]
List[start:end]
```

```
myList[1]
```

```
# bus
```

```
myList[1:]
```

```
# ["bus", "train"]
```

```
myList[:2]
```

```
# ["car", "bus"]
```

Change item in list:

```
List[index] = value
```

```
myList[1] = "taxi"
```

Get Length

```
len(List)
```

```
len(myList)
```

```
# 3
```

List operation

```
myList = ["train", "taxi", "car","rocket",1,2]
```

reverse
append
remove
seperator.join

```
myList.reverse()
myList.append("rocket")
myList.remove(2)
":".join(myList)
```

```
# train:taxi:car:rocket
```

List1 + List2 =

```
myList = myList + [1, 2]
```

```
if "rocket"  in myList:
    print("impossible")
else:
    print("possible")
```

enactus University of Sheffield

CODE CREATORS SHEFFIELD

For loop in list (detailed

```python
lists = ['apple', 'orange', 'banana’]

# printing the tuples in object directly
for i in enumerate(lists):
    print (i)

# i = counter, item = Actual Value
for i, item in enumerate(lists):
    print (i)
    print (item)

# changing starting index to 100
for i, item in enumerate(lists, 100):
    print (i)
    print (item)
```

enactus University of Sheffield

CODE CREATORS SHEFFIELD

Multi-dimensional list

stores values in rows and columns

Can be store with nested list

```
students_grades = [ ["Elton", 90], ["Bernie", 95], ["John", 40] ]
```

Elton	90
Bernie	95
John	40

Index	0	1	2
0	1	2	3
1	4	5	6

Nested Loop

A nested loop refers to a loop statement inside another loop statement

Useful scenario:

- 2D coordinate (x, y)

	0	1	2	3
0	[0][0]	[0][1]	[0][2]	[0][3]
1	[1][0]	[1][1]	[1][2]	[1][3]
2	[2][0]	[2][1]	[2][2]	[2][3]

```
for i in range(4):
    for j in range(3):
        print(i, j)
```

enactus University of Sheffield

CODE CREATORS SHEFFIELD

Tuple

Tuple - ordered, allowing duplicates, unchangeable.

Same DataType in tuple

```
numbers = (1, 2, -5)
names = ("Elton", "Bernie", "John")
```

Access item: `names[1]`

Get the sum: `sum(numbers)`

Get the length: `len(names)`

Get the maximum: `max(numbers)`

Set

Set - unordered, do not allow duplicates, changeable

Use {} to create a set

```
names = {"Ben", "Alan", "Johnny"}
```

Mixed DataType:

```
detail = {21, True, ‘UK’, 21}
```

Empty Set:

```
empty = {}
```

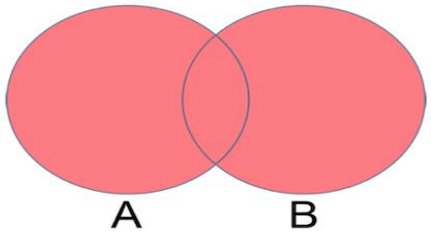
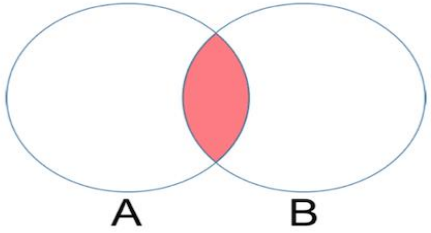
Add item to set:

```
names.add("Jack")
{"Ben", "Alan", "Johnny“, "Jack"}
```

enactus University of Sheffield

CODE CREATORS SHEFFIELD

Set

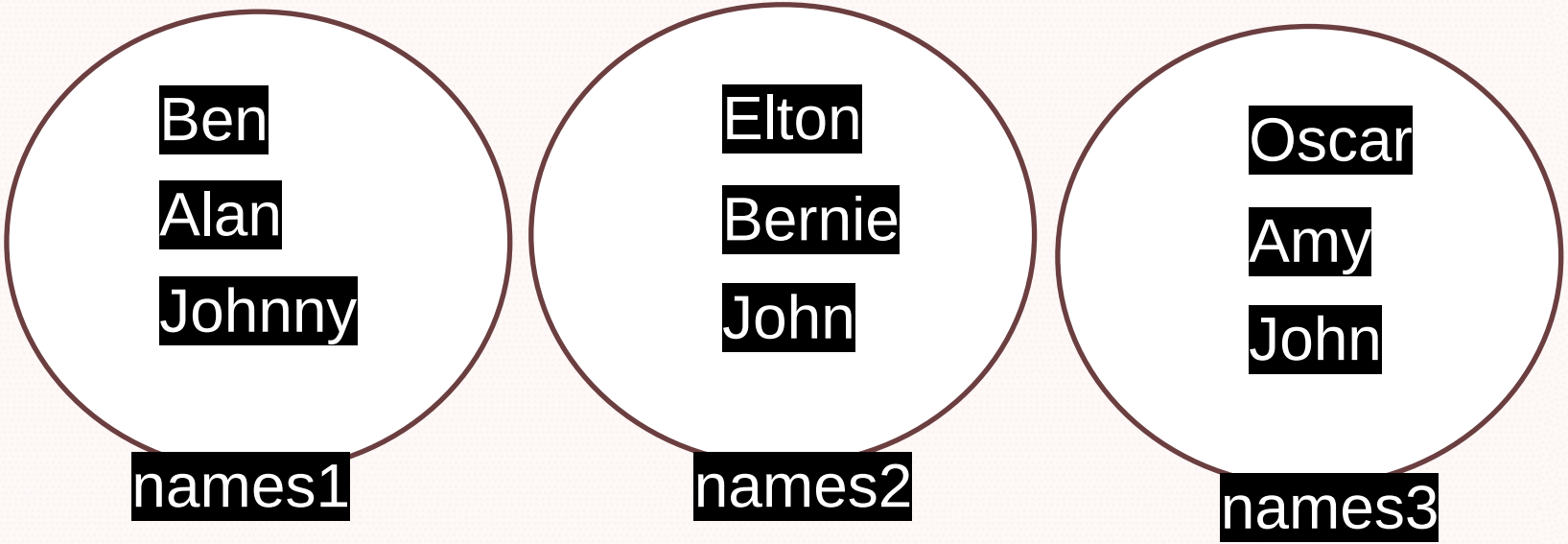
Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection		$A \cap B$, is the set of all values that are members of both A and B.

enactus University of Sheffield

CODE CREATORS SHEFFIELD

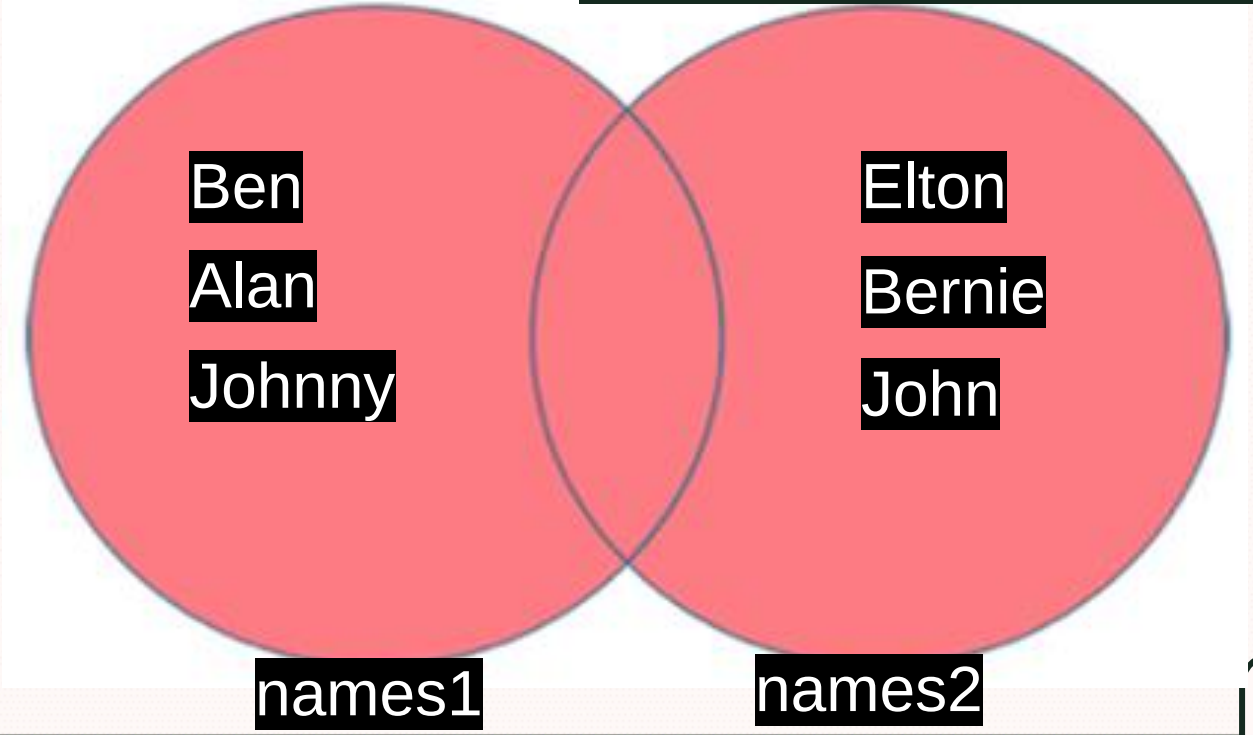
Sets

```
names1 = {"Ben", "Alan", "Johnny"}
names2 = {"Elton", "Bernie", "John"}
names3 = {"Oscar", "Amy", "John"}
```



Union

```
names1 = {"Ben", "Alan", "Johnny"}
names2 = {"Elton", "Bernie", "John"}
names1.union(names2)
```

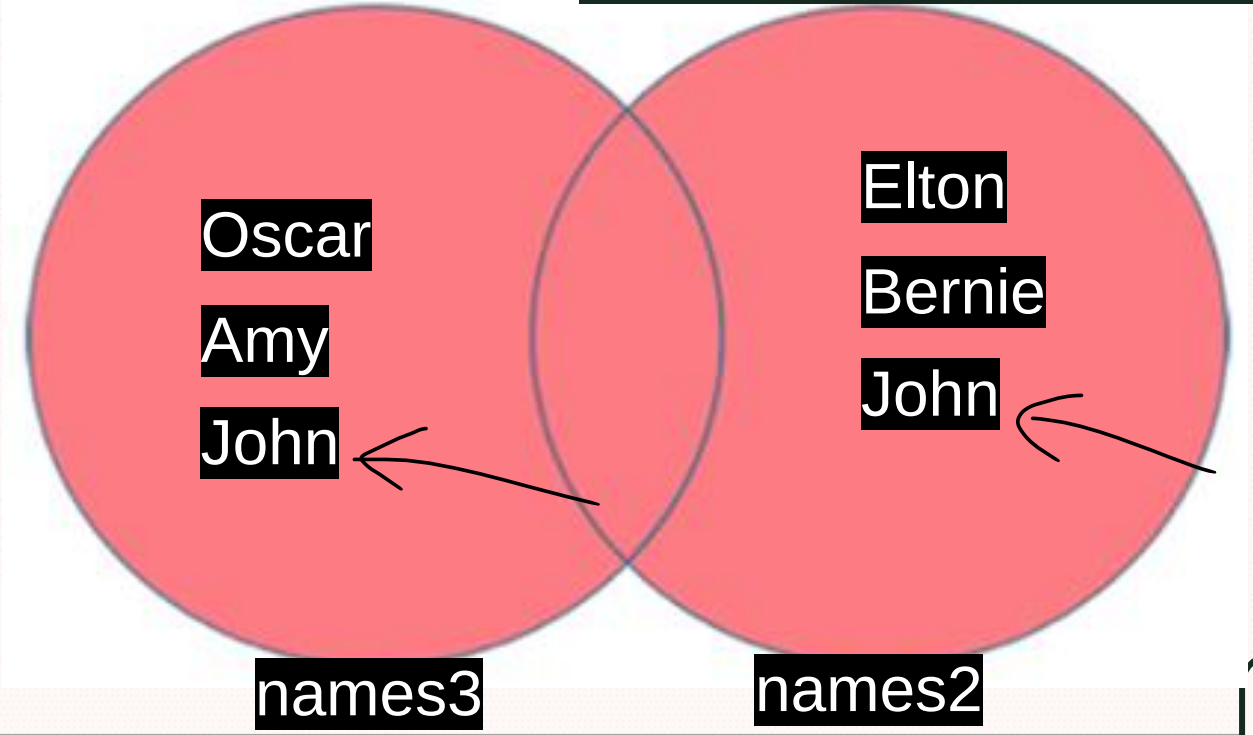


enactus University of Sheffield

CODE CREATORS SHEFFIELD

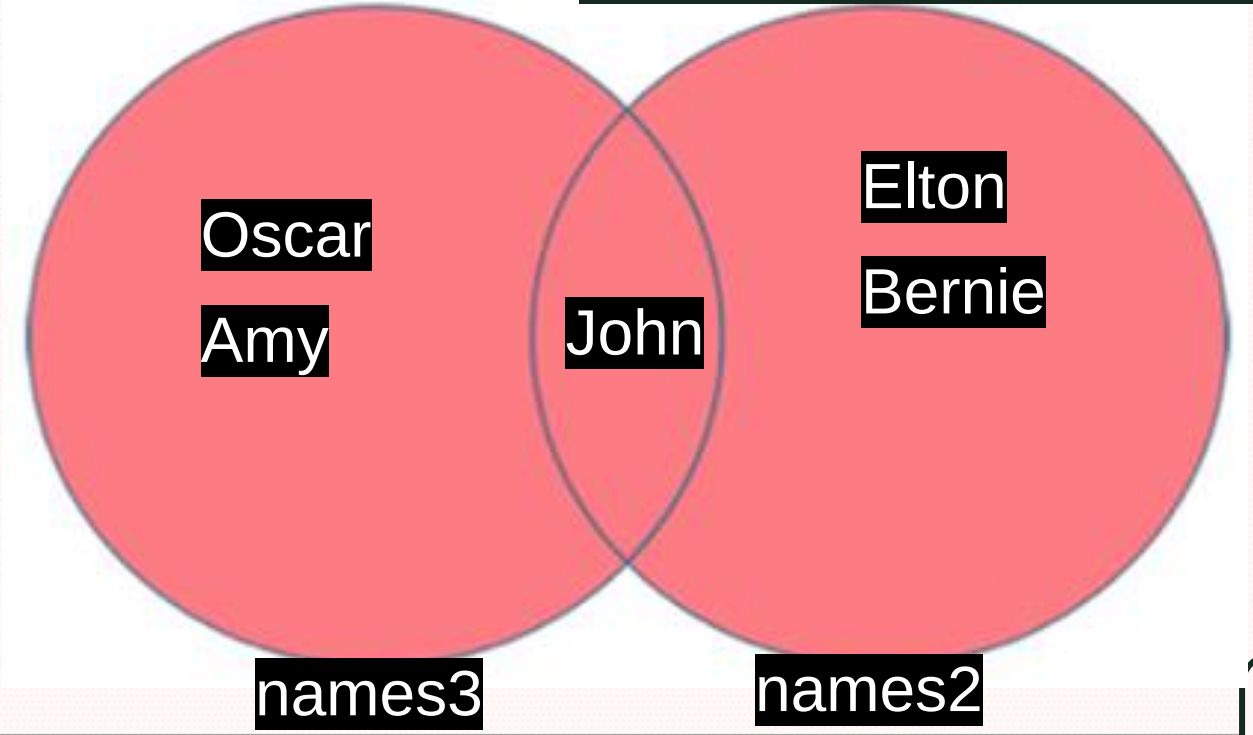
Union

```
names2 = {"Elton", "Bernie", "John"}
names3 = {"Oscar", "Amy", "John"}
names3.union(names2)
```



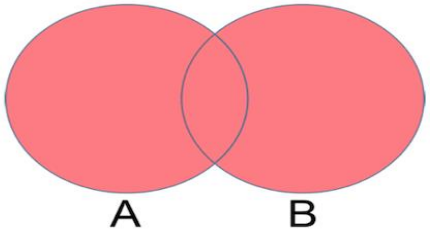
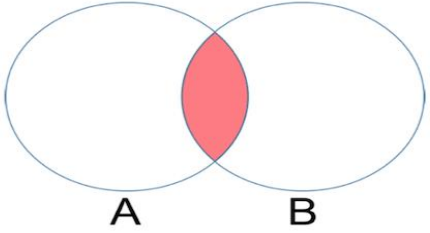
Union

```
names2 = {"Elton", "Bernie", "John"}
names3 = {"Oscar", "Amy", "John"}
names3.union(names2)
```



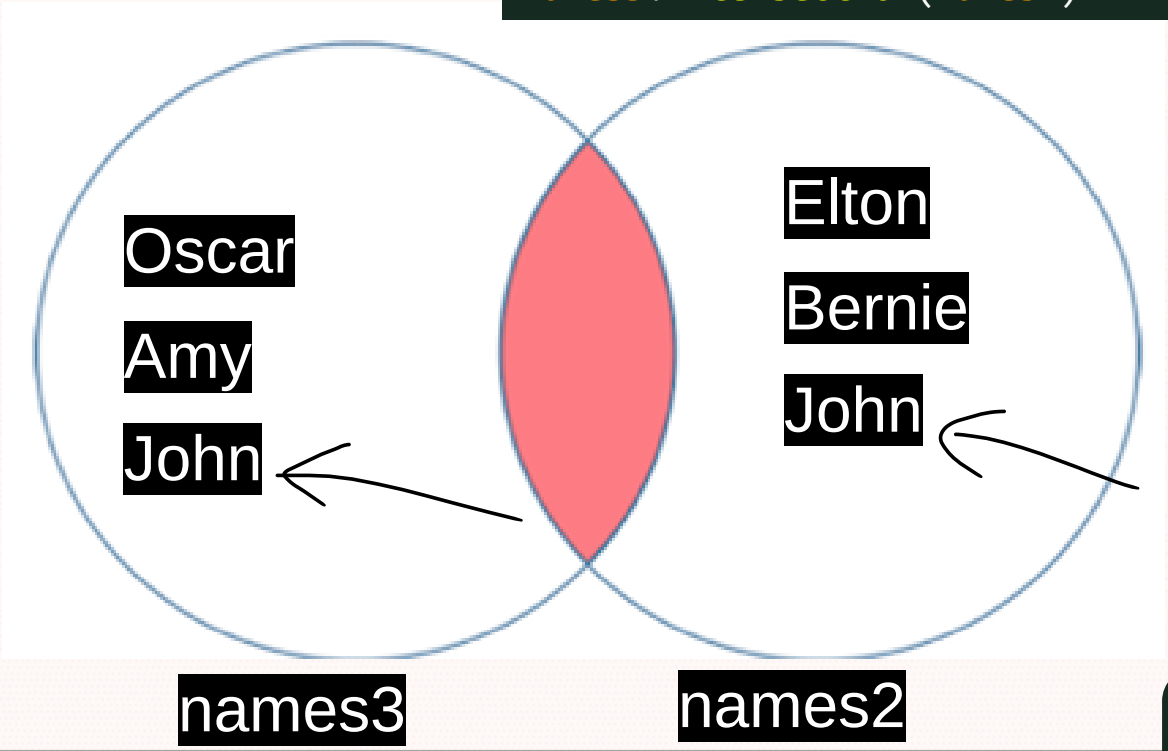
enactus
University of Sheffield
CODE CREATORS
SHEFFIELD

Set

Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection		$A \cap B$, is the set of all values that are members of both A and B.

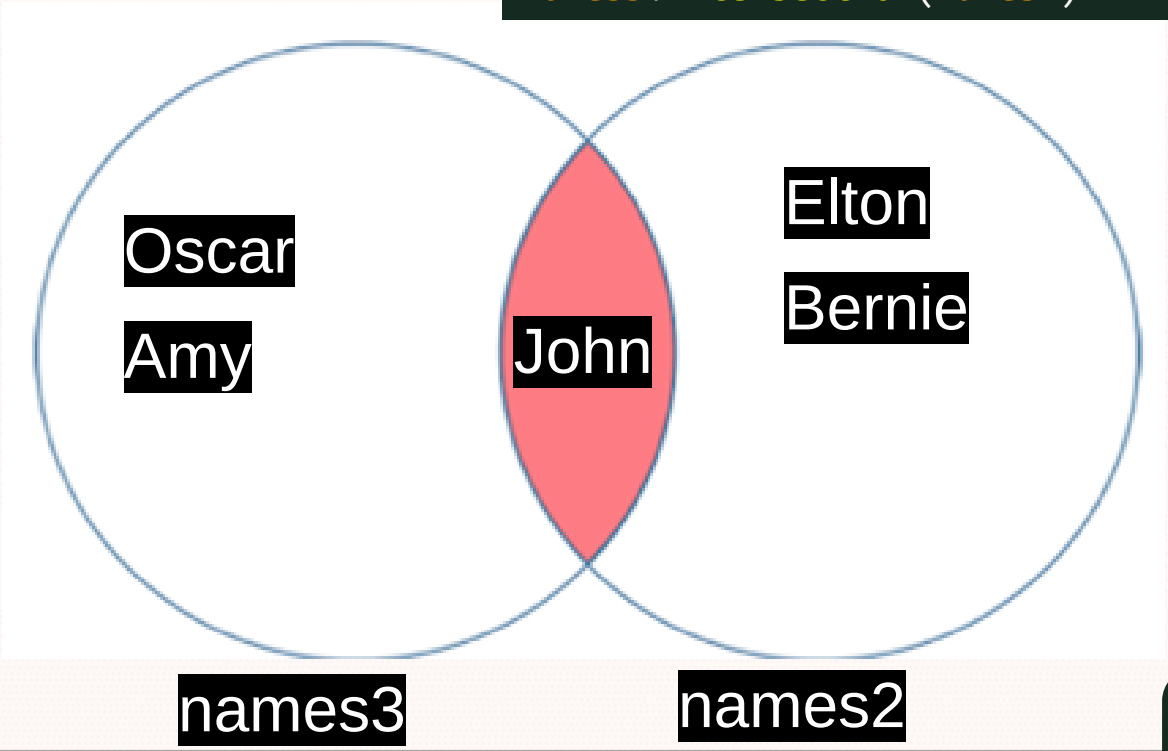
Intersect

```
names2 = {"Elton", "Bernie", "John"}
names3 = {"Oscar", "Amy", "John"}
names3.intersection(names2)
```



Intersect

```
names2 = {"Elton", "Bernie", "John"}
names3 = {"Oscar", "Amy", "John"}
names3.intersection(names2)
```



Comparison

	Mutable	Ordered	Indexing / Slicing	Duplicate Elements
List	✓	✓	✓	✓
Tuple	✗	✓	✓	✓
Set	✓	✗	✗	✗

Class Exercise

1. Initialise a list of 10 names. Output the array to the screen. **Now alter your code to output the list to the screen in reverse order.**

2. Write a program to choose 100 random numbers between 1 and 100. Count and display how many numbers are in the range 1-10, 11-20, 21-30 etc.

Challenge Exercise

Declare a list of 7 integers. Initialise each element to 0.

C: Use a loop and random numbers to simulate the rolling of a single die.

B: If a 1 is rolled, increase the value of element 1 in the array by 1, if a 2 is rolled, increase the value of element 2 by 1.

A: When all 200 rolls have been simulated, output the tally to the console.