Managing Tuples and Sets

Mihaela Danci
Data Analyst

linkedin.com/in/mihaela-danci/

Overview

Tables

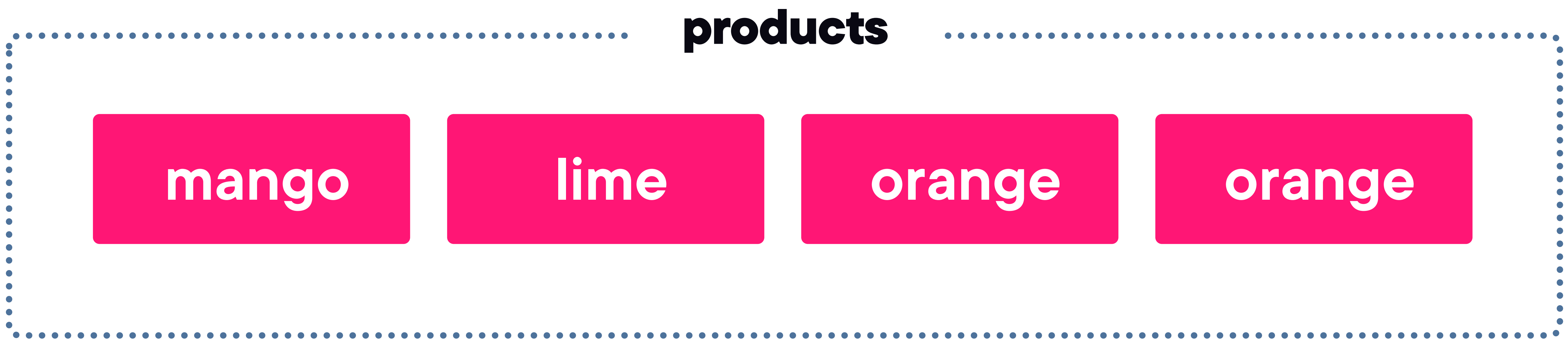
Tuples

- Construct
- Access elements
- Unpack

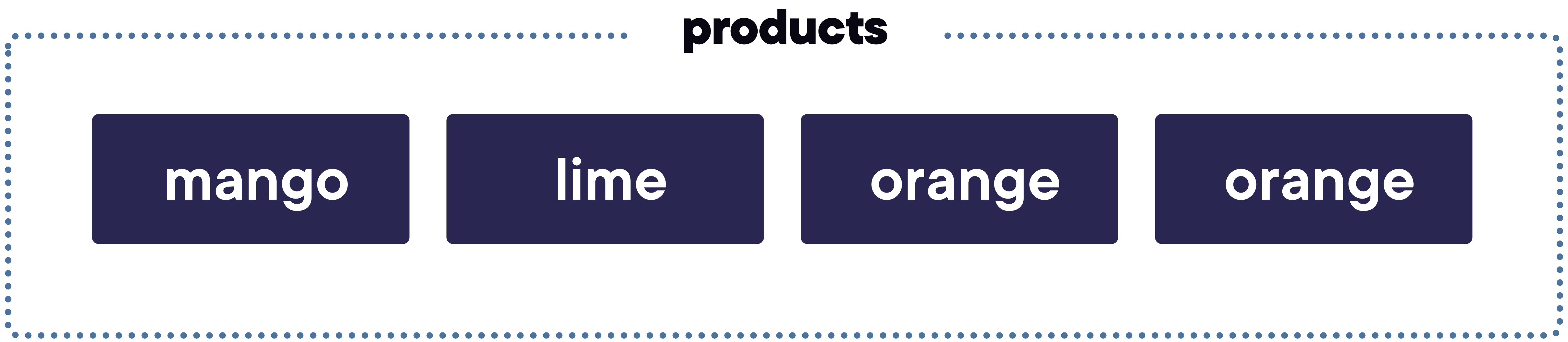
Sets

- Create
- Manipulate
- Logic operations

Tuples



```
products = ('mango', 'lime', 'orange', 'orange')
```



```
products = ('mango', 'lime', 'orange', 'orange')
```

Tuples

```
products =  ('mango', 'lime', 'orange’)
```

```
products = 'mango', 'lime', 'orange'
```

Tuples

```
products =  ('mango', 'lime', 'orange’)
```

```
products = 'mango', 'lime', 'orange'
```

Tuple packing

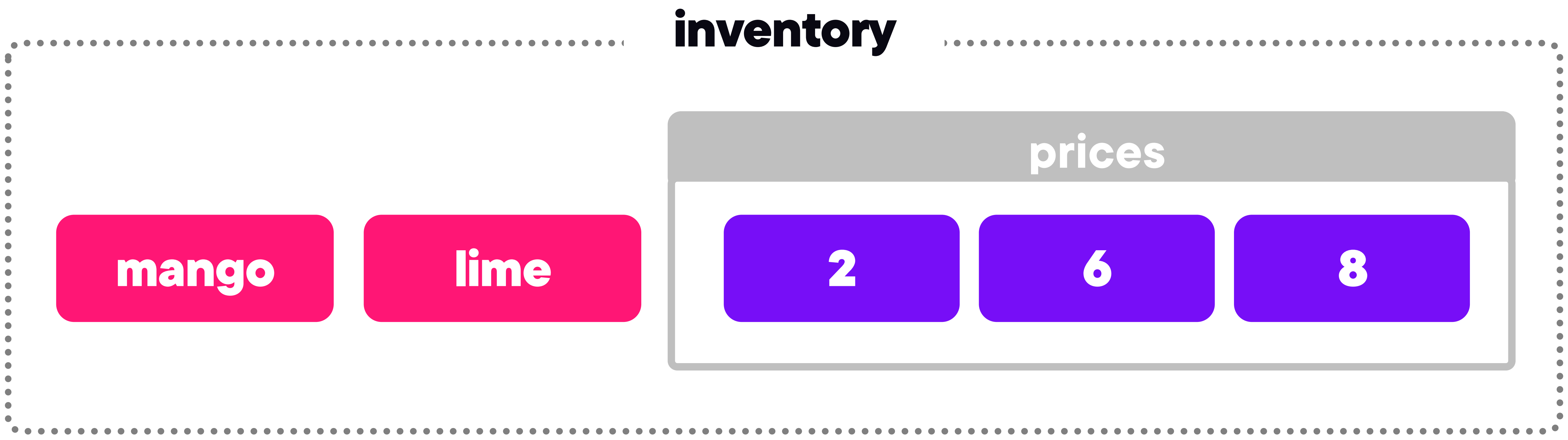
```
products =  ('mango', 'lime', 'orange’)
```

```
products = 'mango', 'lime', 'orange'
```

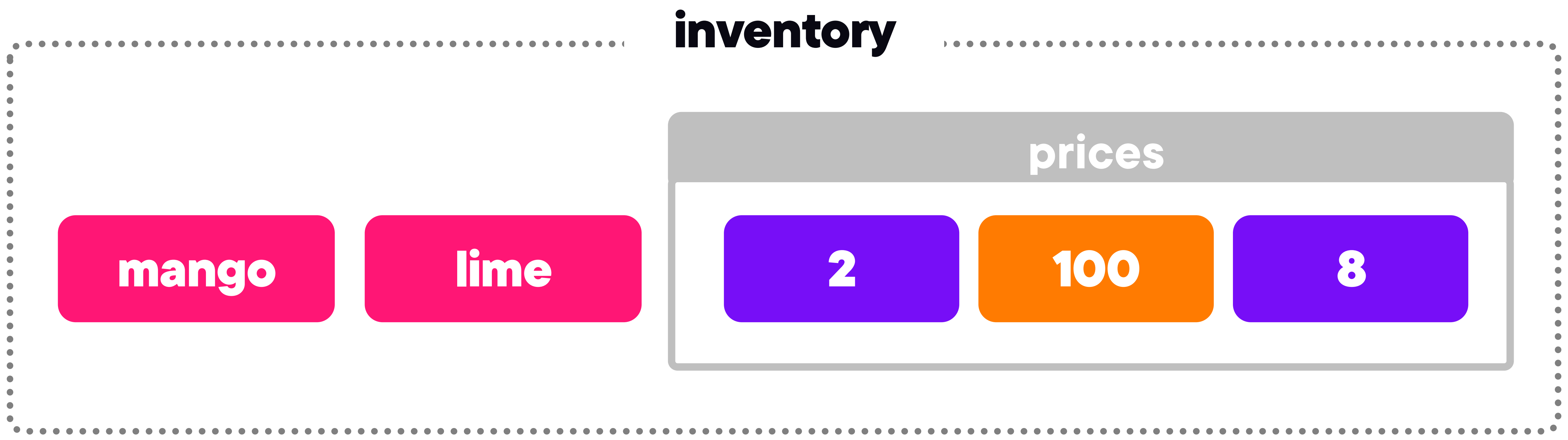
Tuples are a limited version of lists

Tuples are immutable, while lists are mutable.

Nested Tuples



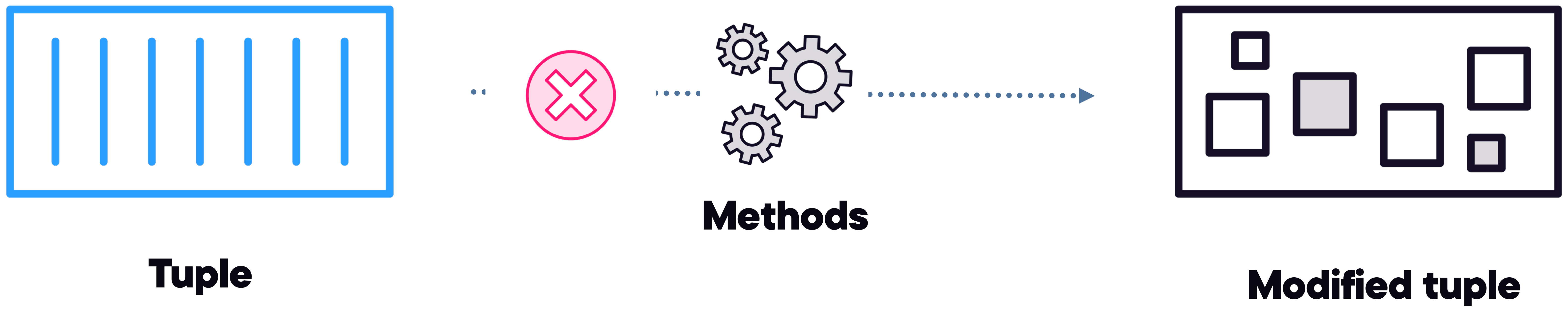
Nested Tuples



Manipulating Tuples

Tuple
Methods
Modified tuple

Manipulating Tuples



Manipulating Tuples

Methods

append

remove

pop

sort

Why use tuples?

Memory management

Memory management

Preserving the data

Temporary variables

Memory management

Preserving the data
Temporary variables

Functions

```
# Function with two arguments
def  math_operations(num1, num2):
     sum = num1 + num2
     diff = num1 - num2
     power = num1 * num2
     div = num1 / num2
     return sum
```

◂ **Create the function**

```
# Function with two arguments

def  math_operations(num1, num2):

     sum = num1 + num2

     diff = num1 - num2

     power = num1 * num2

     div = num1 / num2

     return sum


math_operations(4,2)
```

◀ **Create the function**

◀ **The function returns only one result**

◀ **6**

```python
# Function with two arguments

def  math_operations(num1, num2):

     sum = num1 + num2

     diff = num1 - num2

     power = num1 * num2

     div = num1 / num2

     return (sum, diff, power, div)


math_operations(4,2)
```

◀ **The function returns multiple values in a tuple**

◀ **(6, 2, 8, 2)**

Comparing Tuples

a = (4, 7, 5, 2)

b = (4, 7, 8, 9)

a > b

Comparing Tuples

```
a =  ( 4, 7, 5, 2)
```

```
b =  ( 4, 7, 8, 9)
```

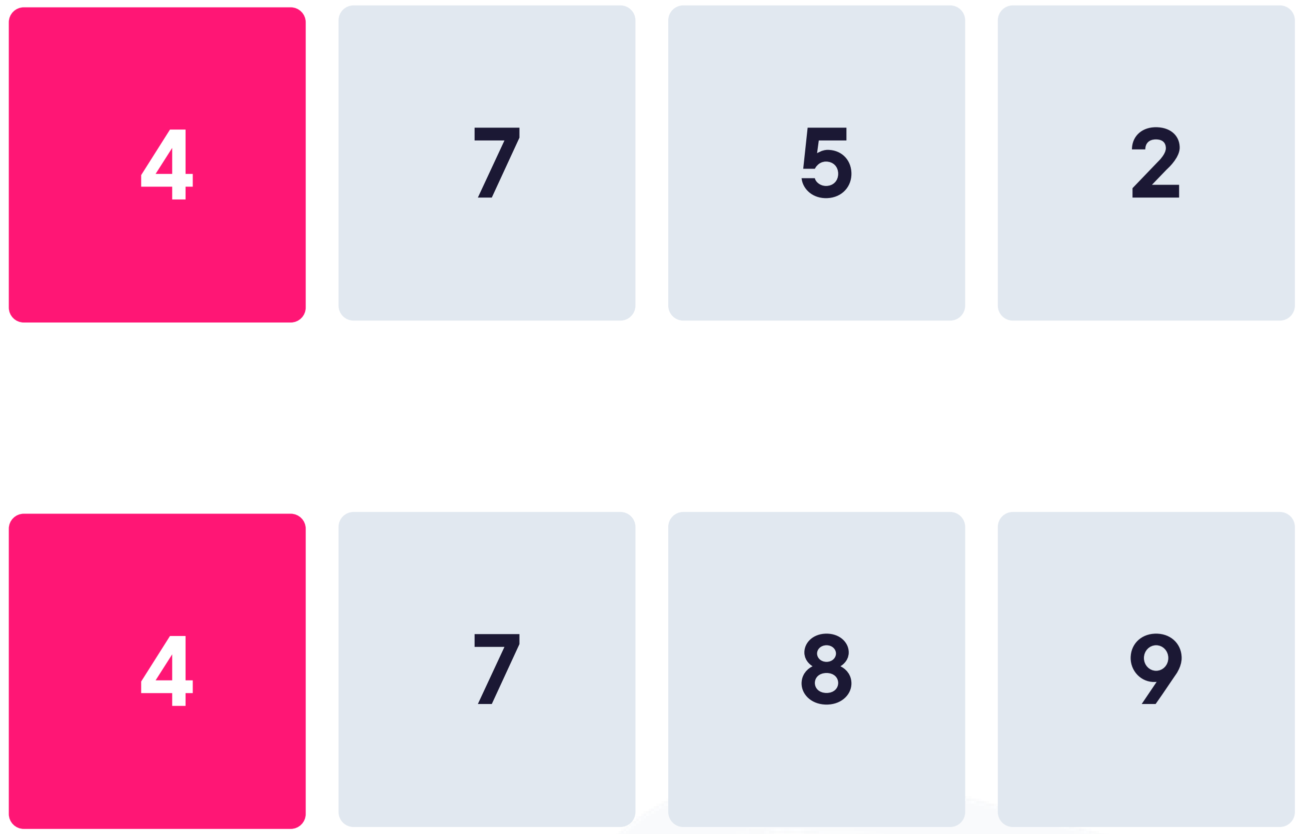
4	7	5	2
4	7	8	9

```
a > b
```

Comparing Tuples

```
a = ( 4, 7, 5, 2)
```

```
b = ( 4, 7, 8, 9)
```

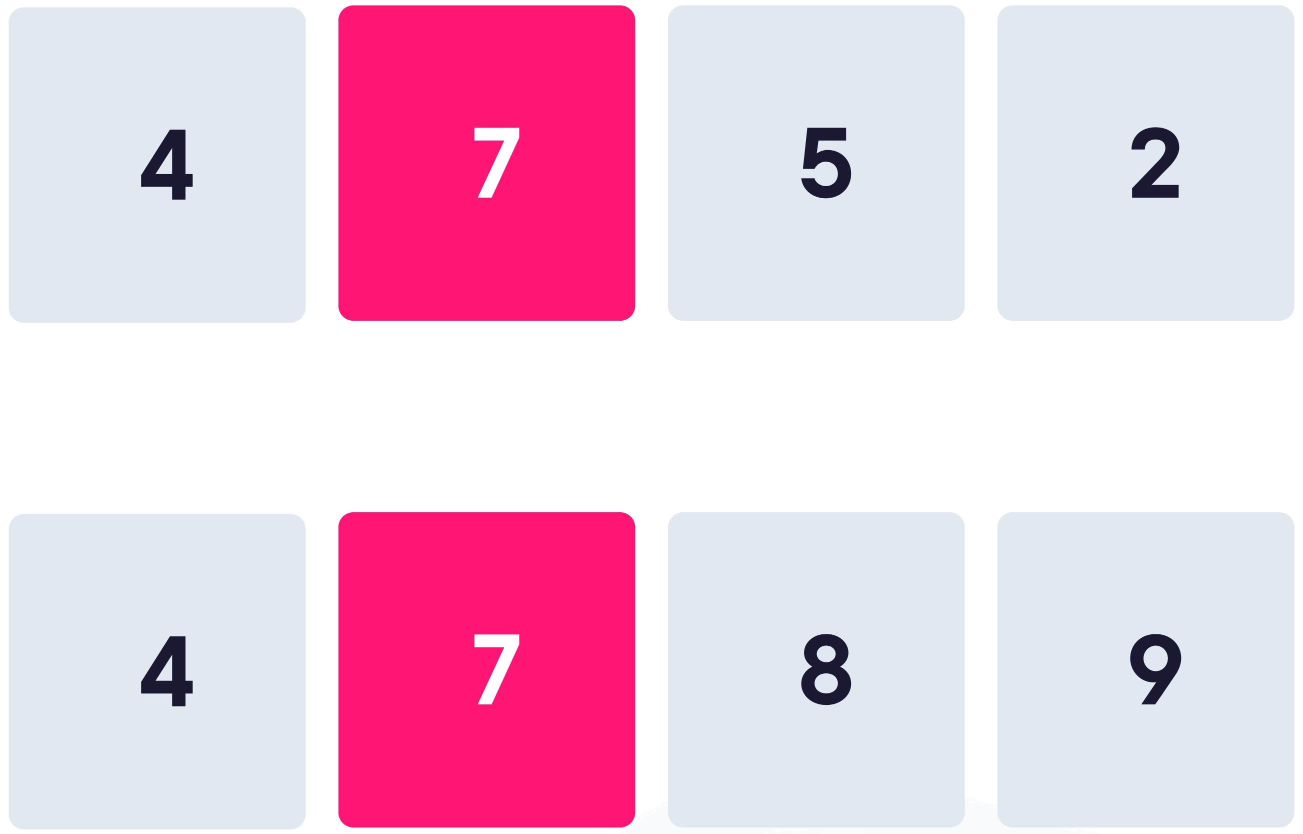


a > b

Comparing Tuples

```
a = ( 4, 7, 5, 2)
```

```
b = ( 4, 7, 8, 9)
```

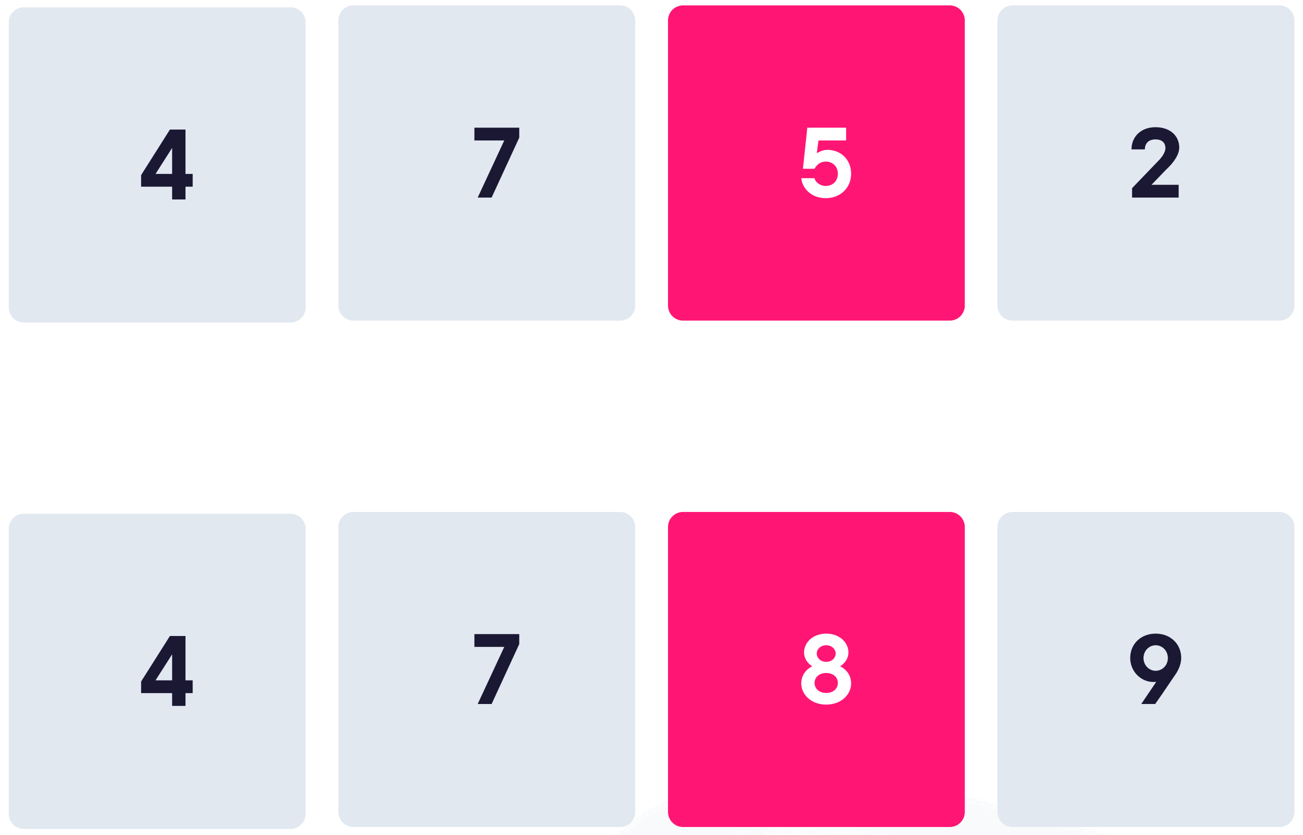


a > b

Comparing Tuples

```
a = ( 4, 7, 5, 2)
```

```
b = ( 4, 7, 8, 9)
```



```
a > b                False
```

Tuples	Lists
Hold items of different data types	Hold items of different data types
Nested tuples	Nested lists
Support indexing, slicing, and membership testing	Support indexing, slicing, and membership testing
Are immutable	Are mutable
Usually store items of different data types	Usually store items of the same data types

VS

Demo

Create tuples

Update an element

- Converting the tuple into a list
- Completing the change
- Converting the list into a tuple

Demo

Manipulating tuples

- Slicing
- Indexing

Unpacking Tuples

Packing Tuples

```
prices =  (0.5, 1.5, 2.5)
```

Packing Tuples

```
prices = (0.5, 1.5, 2.5)
```

Unpacking Tuples

```
( mango,   pear,   banana)  = (0.5,  1.5,  2.5)
```

Unpacking Tuples

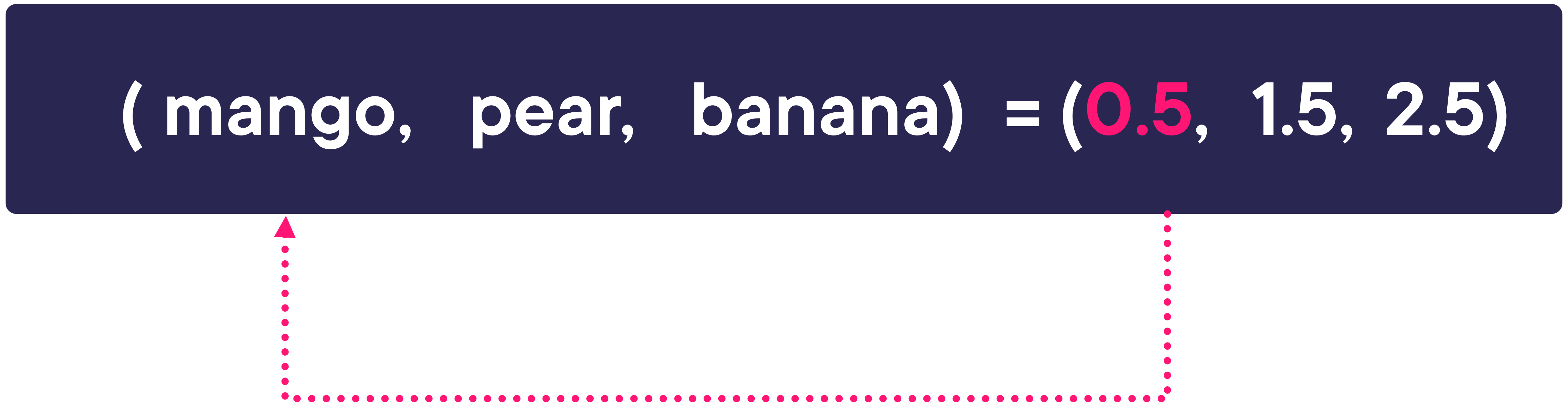
```
( mango,   pear,   banana)  = (0.5,  1.5,  2.5)
```

Unpacking

Unpacking Tuples

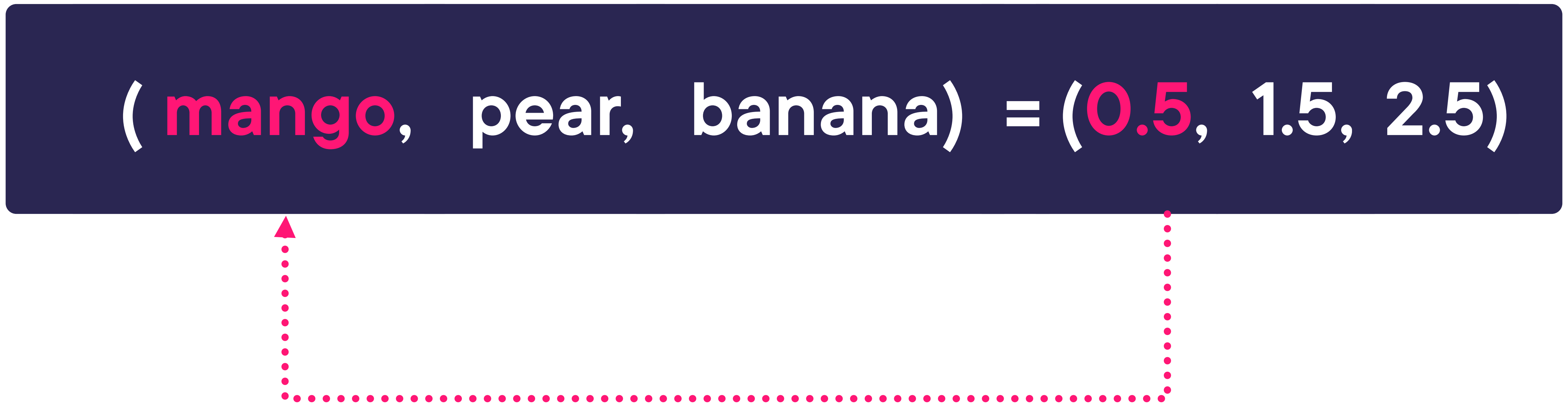
```
( mango,   pear,   banana)  = (0.5,  1.5,  2.5)
```

Unpacking Tuples



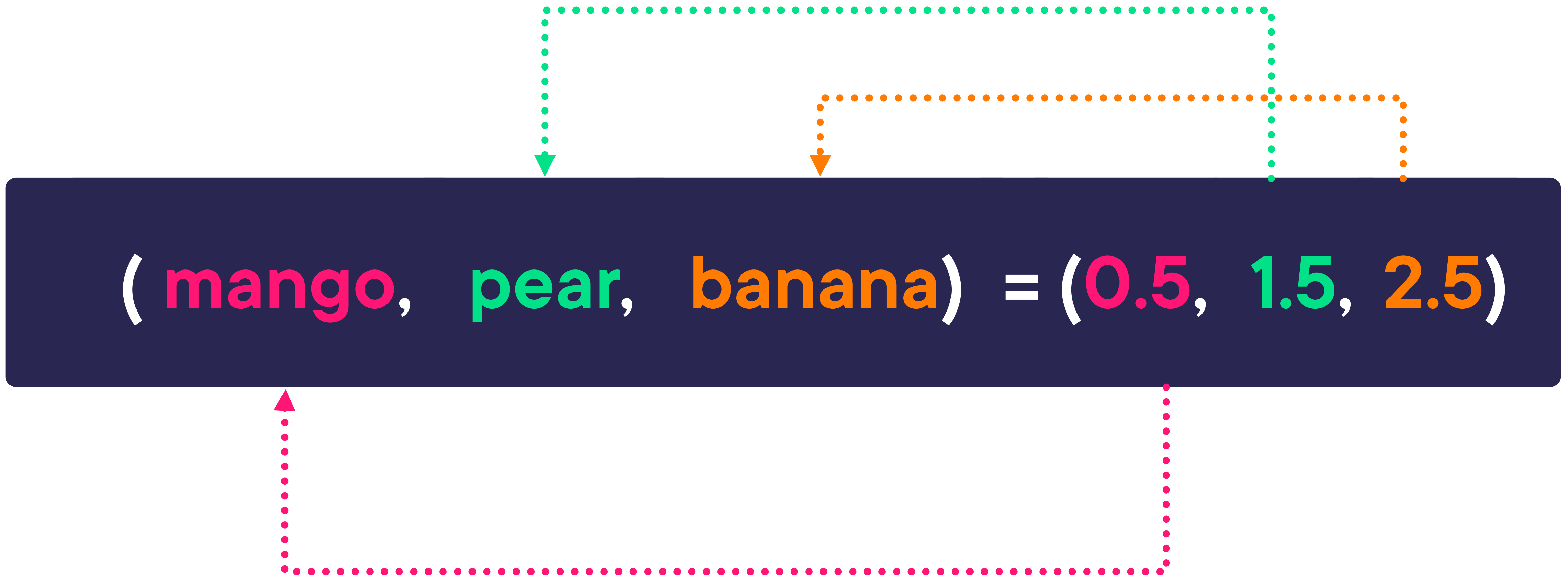
```
( mango,   pear,   banana)  = (0.5,  1.5,  2.5)
```

Unpacking Tuples



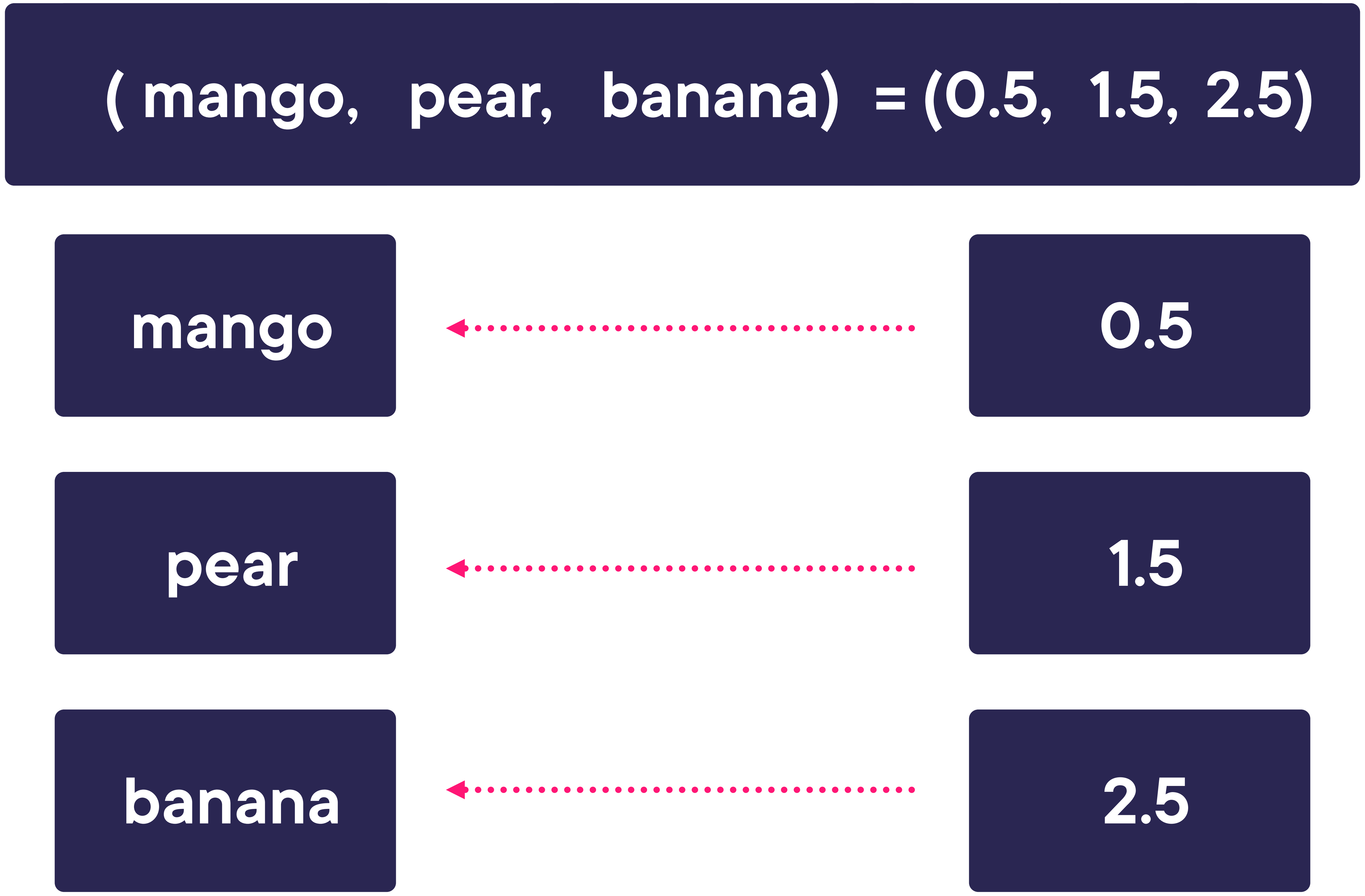
```
( mango,   pear,   banana)  = (0.5,  1.5,  2.5)
```

Unpacking Tuples



```
( mango,   pear,   banana)  = (0.5,  1.5,  2.5)
```

Unpacking Tuples



Accesing Elements

Indexing

Unpacking

We must have the same number of values on both sides of the assignment operator.

Demo

Unpacking tuples

Sets

Properties

Are unordered

Don’t accept duplicates

Are mutable

Can be transformed into immutable sets with frozenset() command

Their elements are immutable

Creating Sets

{ }

set ()

Demo

Construct sets

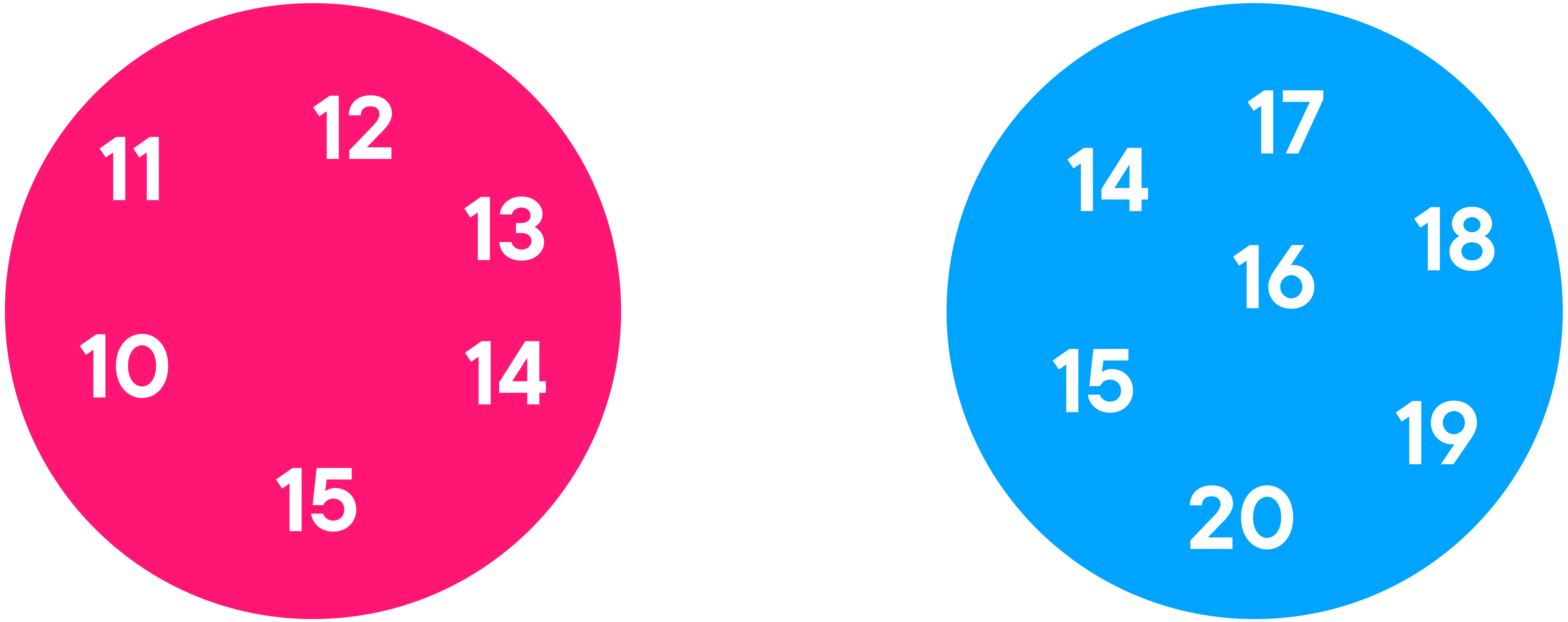
Demo

Add elements

Remove elements

Logic Operations

Creating Sets

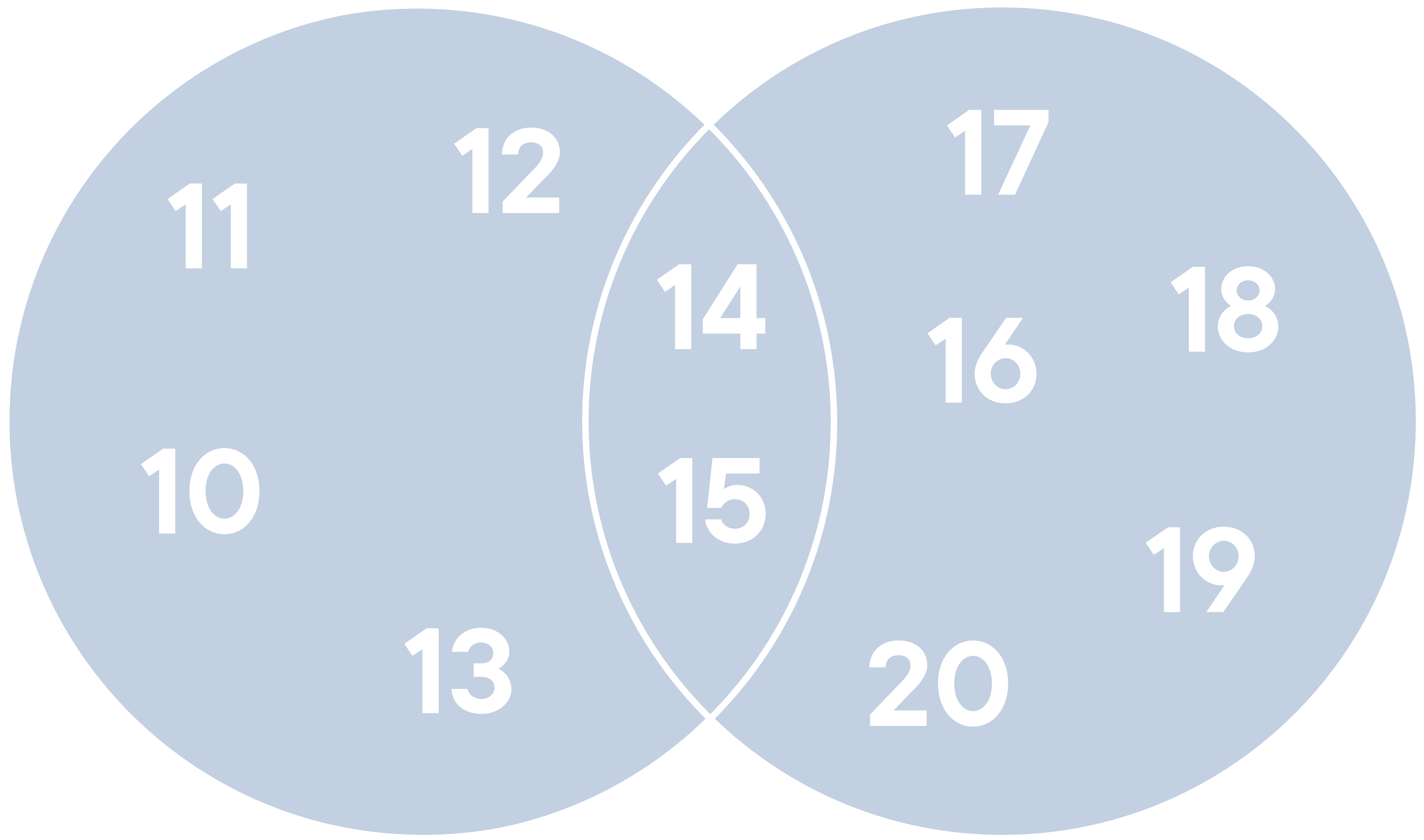


A = {10, 11, 12, 13, 14, 15}

B = {14, 15, 16, 17, 18, 19, 20}

A.intersection(B)

Intersection

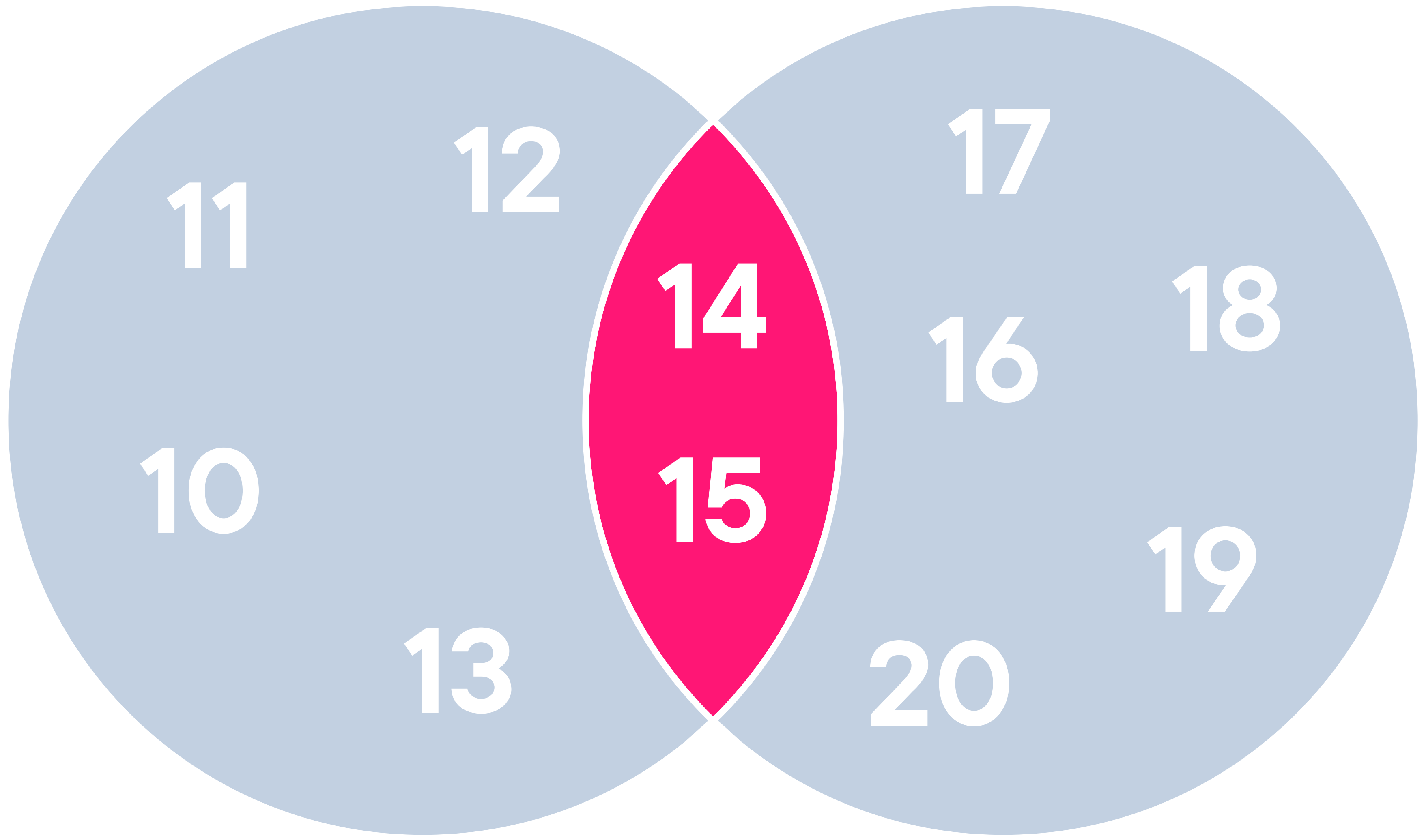


A = {10, 11, 12, 13, 14, 15}

B = {14, 15, 16, 17, 18, 19, 20}

A.intersection(B)

Intersection



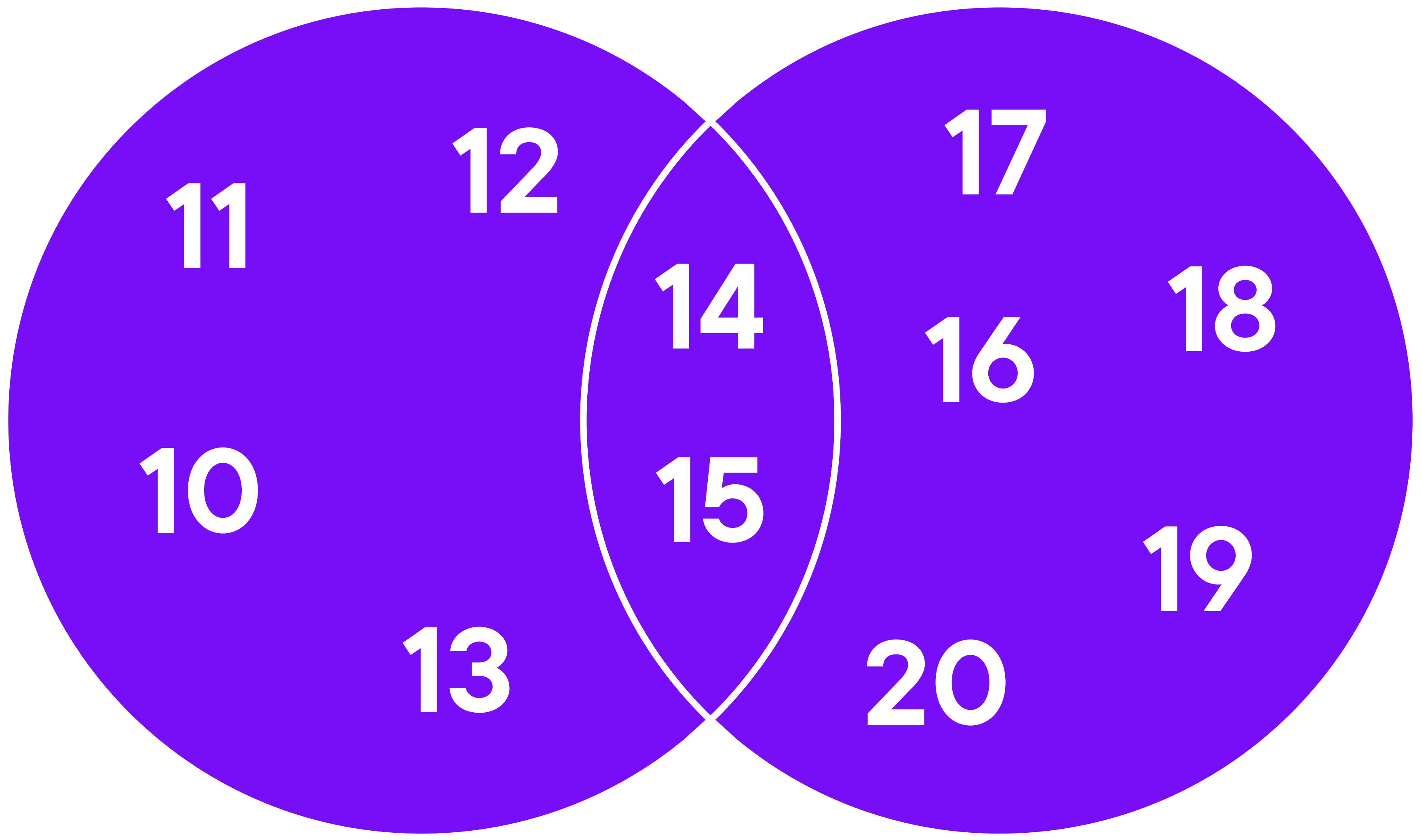
A = {10, 11, 12, 13, 14, 15}

B = {14, 15, 16, 17, 18, 19, 20}

A.intersection(B)

14, 15

Union

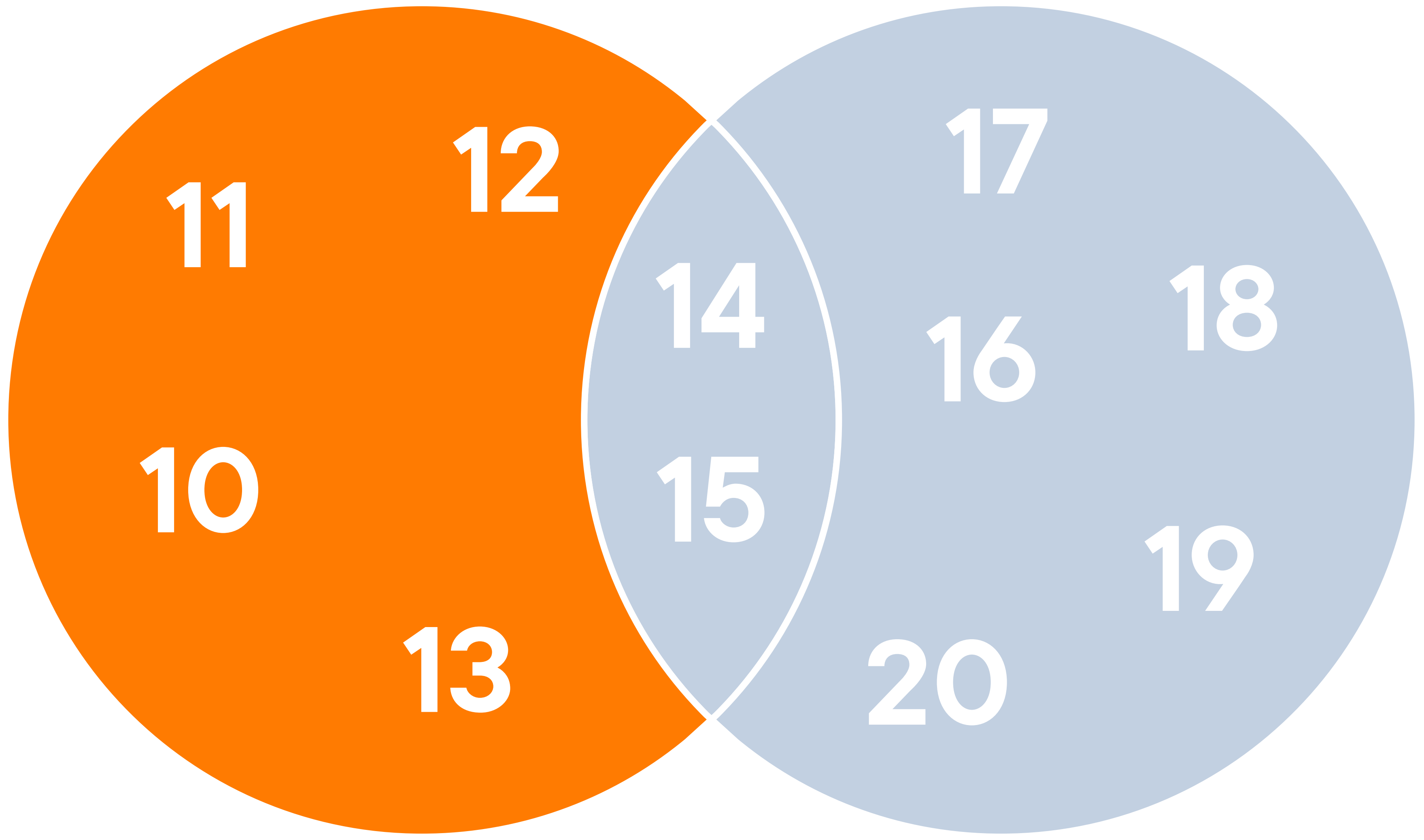


A = {10, 11, 12, 13, 14, 15}

B = {14, 15, 16, 17, 18, 19, 20}

A.union(B)

Difference



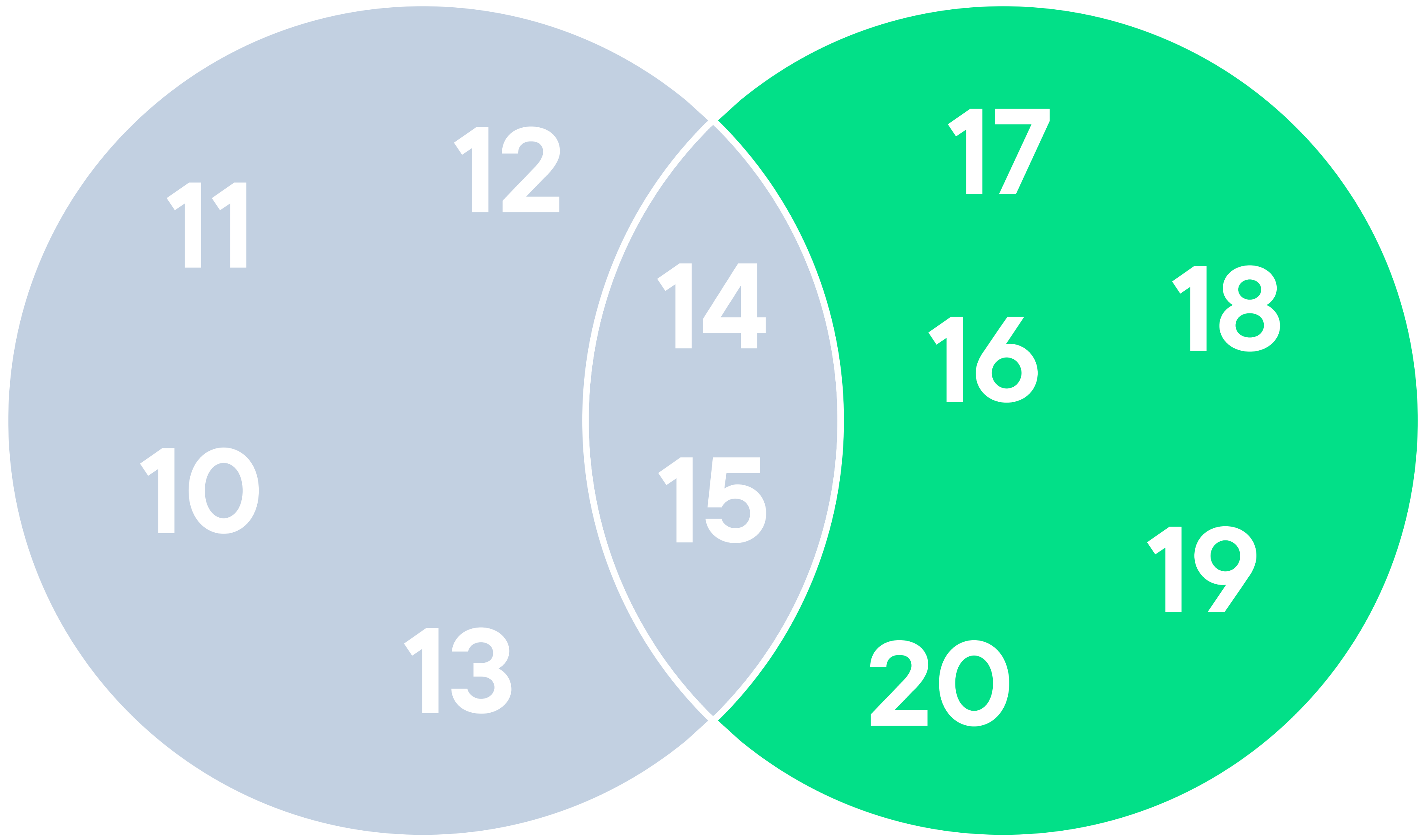
A = {10, 11, 12, 13, 14, 15}

B = {14, 15, 16, 17, 18, 19, 20}

A.difference(B)

10, 11, 12, 13

Difference



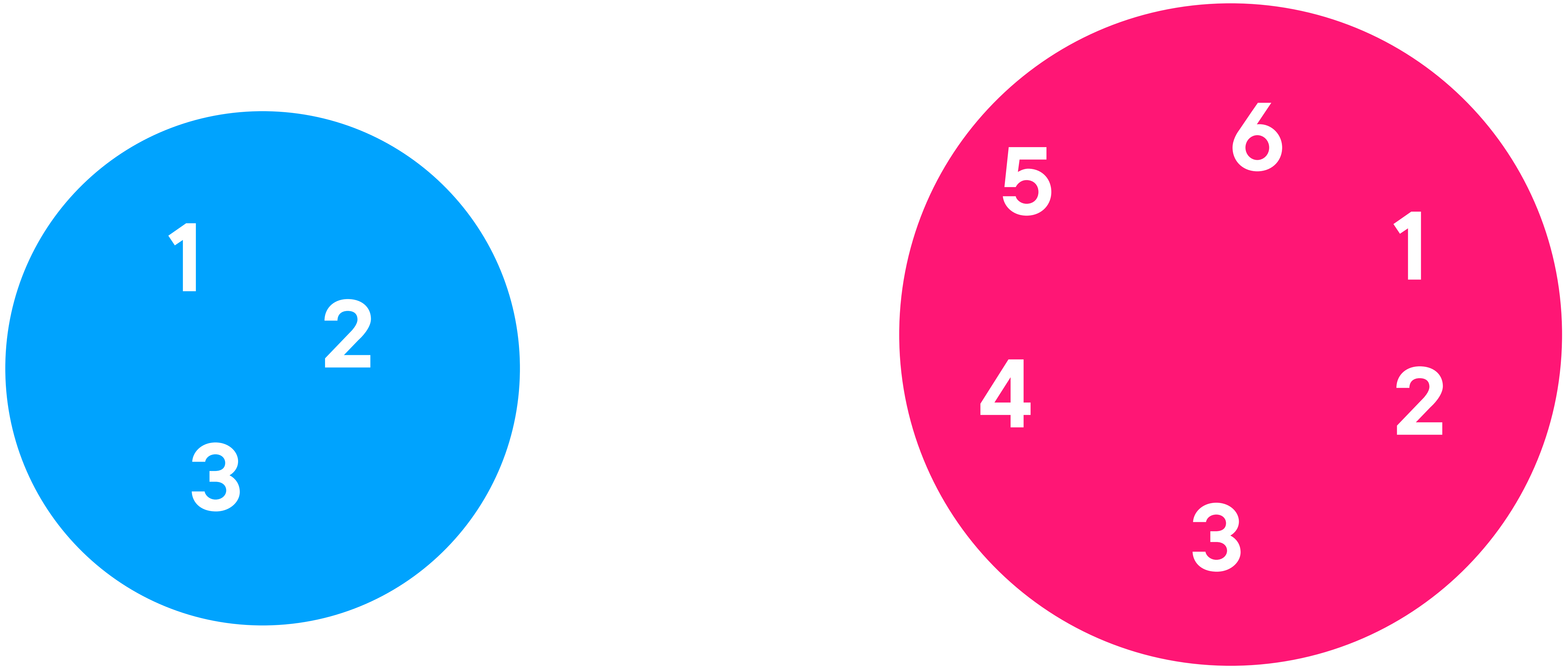
A = {10, 11, 12, 13, 14, 15}

B = {14, 15, 16, 17, 18, 19, 20}

B.difference(A)

16, 17, 18, 19, 20

Subset

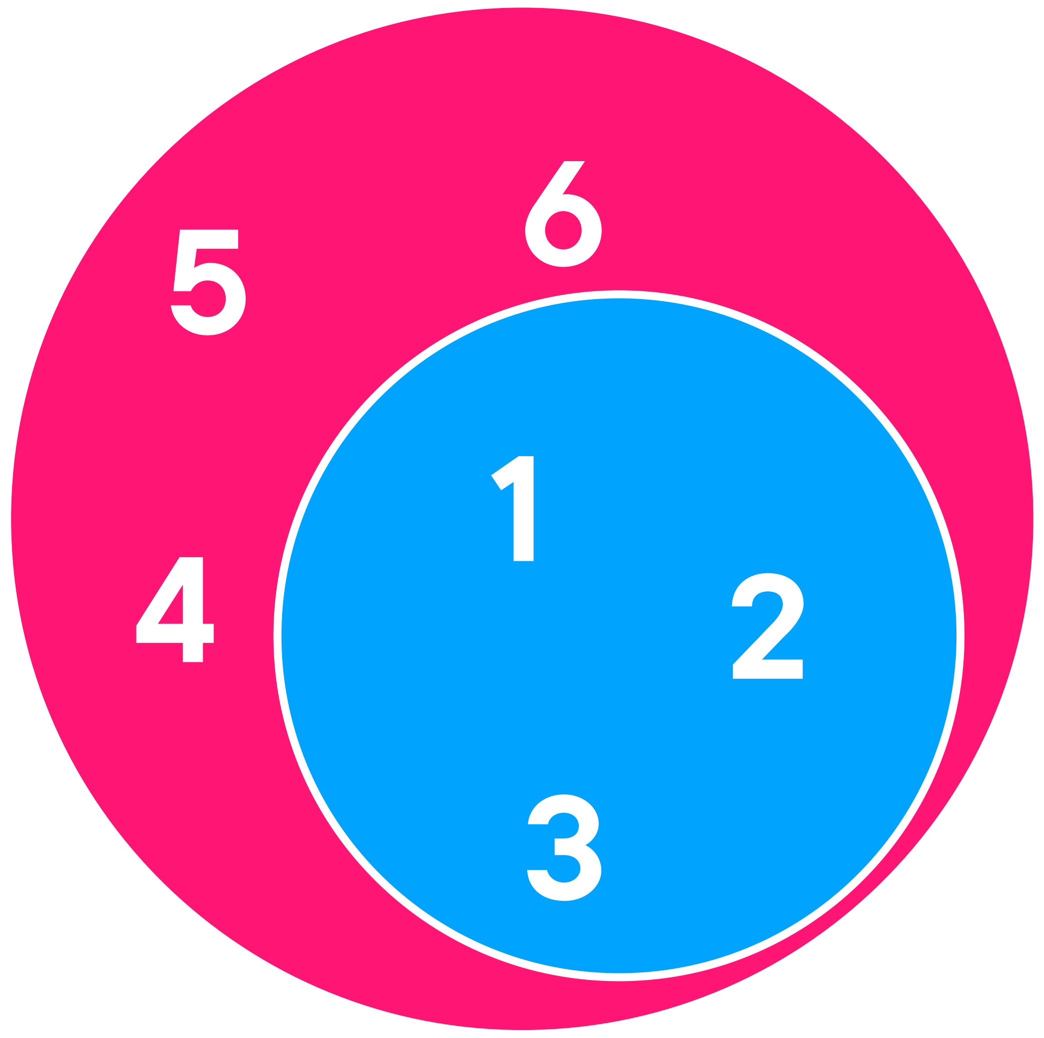


X = {1, 2, 3}

Y = {1, 2, 3, 4, 5, 6}

X.issubset(Y)

Subset

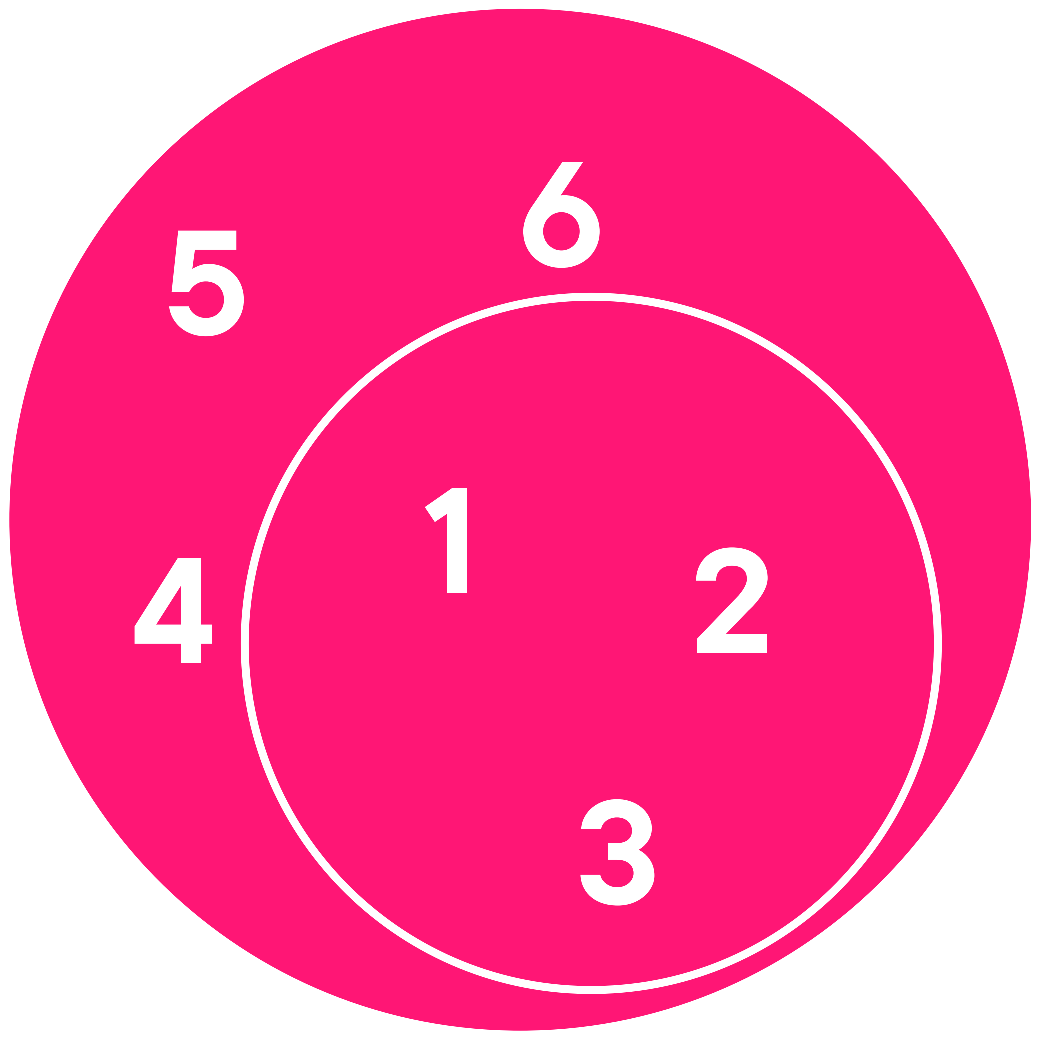


X = {1, 2, 3}

Y = {1, 2, 3, 4, 5, 6}

```
X.issubset(Y)
True
```

Superset



X = {1, 2, 3}

Y = {1, 2, 3, 4, 5, 6}

Y.issuperset(X)

True

Demo

Logical operations

Summary

Tuples

- Limited version of lists
- Immutable
- Memory management
- Return multiple values in a function

Sets

- Hold unique and unordered elements
- Mutable
- Methods to add and remove elements
- Logical operations

Up Next:

Using Dictionaries