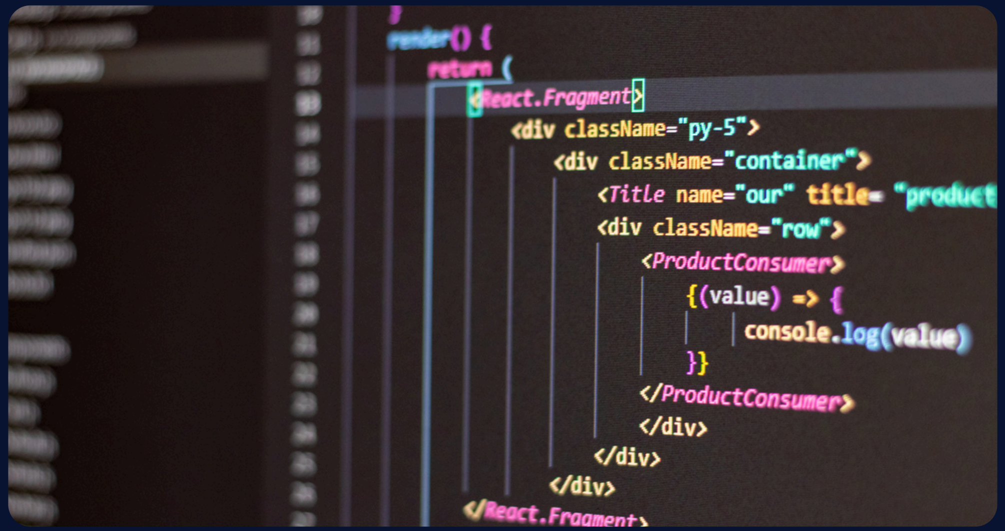
Python



```
render() {
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="our" title= "product
          <div className="row">
            <ProductConsumer>
              {(value) => {
                console.log(value)
              }}
            </ProductConsumer>
          </div>
        </div>
      </div>
    </React.Fragment>
```

Python Sets

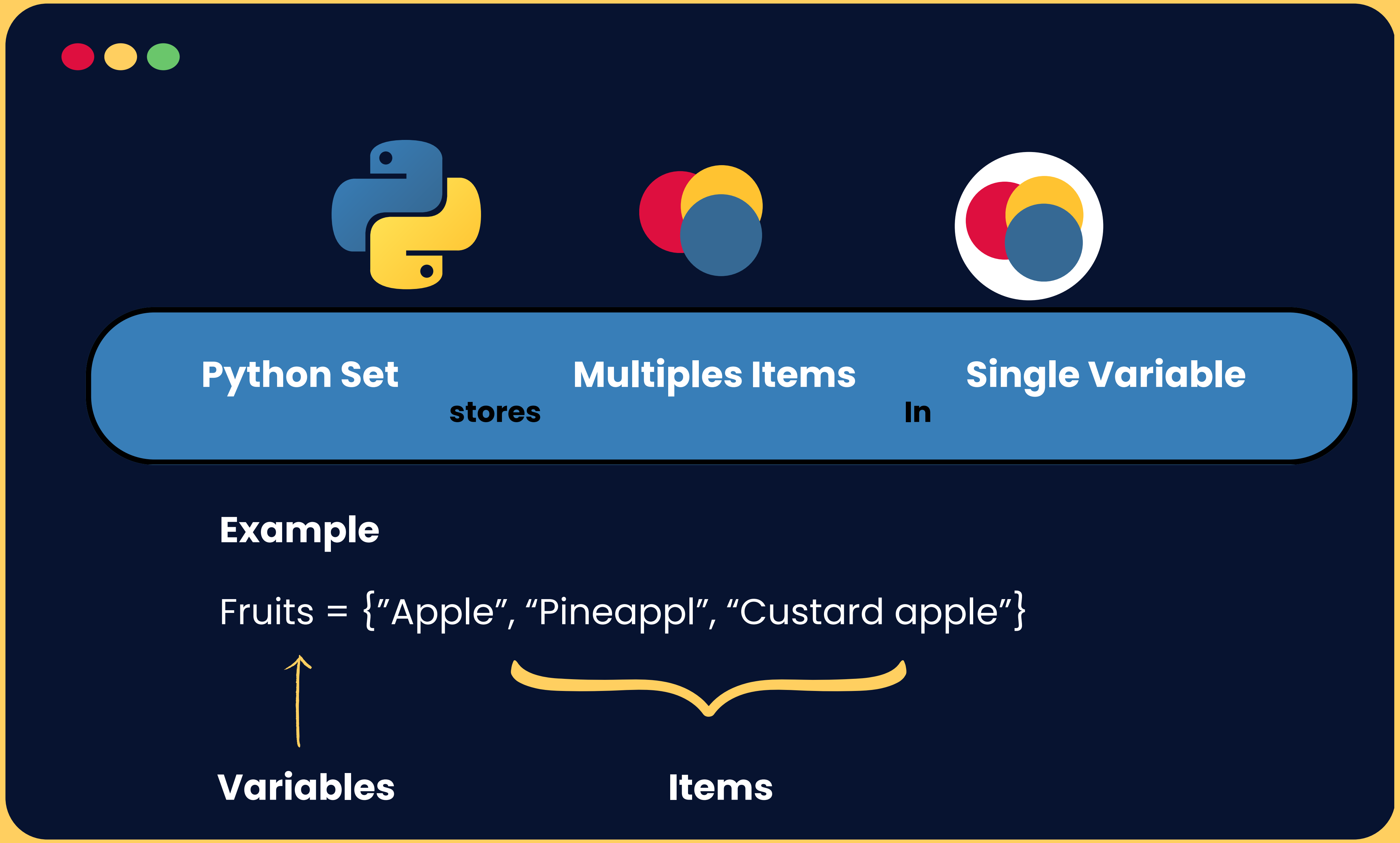
Presenting By

Prachiti Salvi

A Quick Overview

Introduction to Sets

- A set is a collection of unique, unordered elements in Python.
- It does not allow duplicate values and is defined using {} or the set() function.



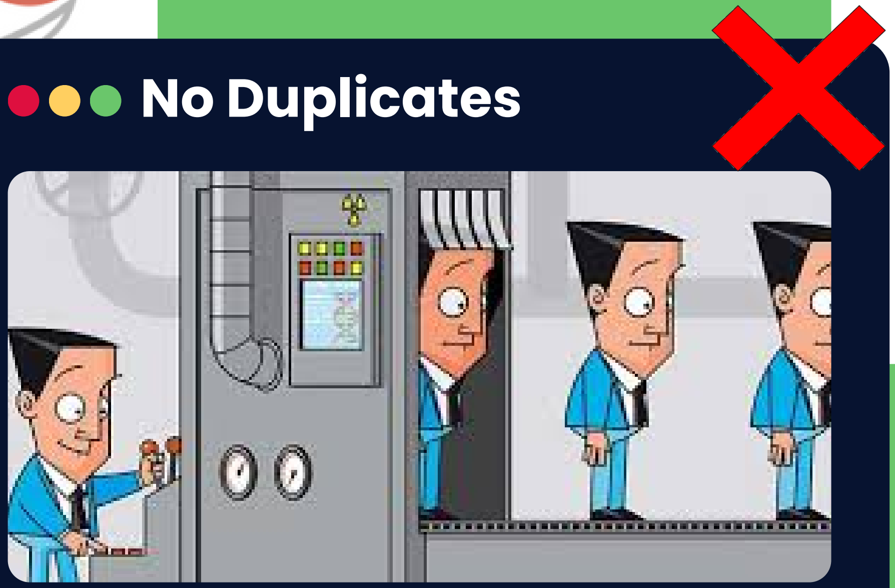
```
Fruits = {"Apple", "Pineappl", "Custard apple"}
```

Characteristics of Sets

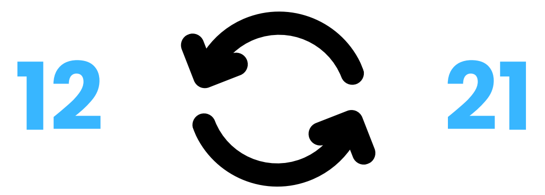
- Unordered → Elements do not maintain a specific order
- No Duplicates → Each element appears only once
- Mutable → We can add or remove elements
- Supports Set Operations → Union, intersection, difference, etc.

Unordered

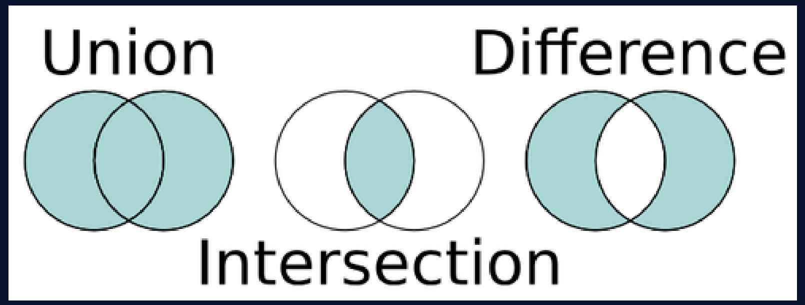
No Duplicates



Mutable



Supports Set Operations



How to create Sets?

Syntax:

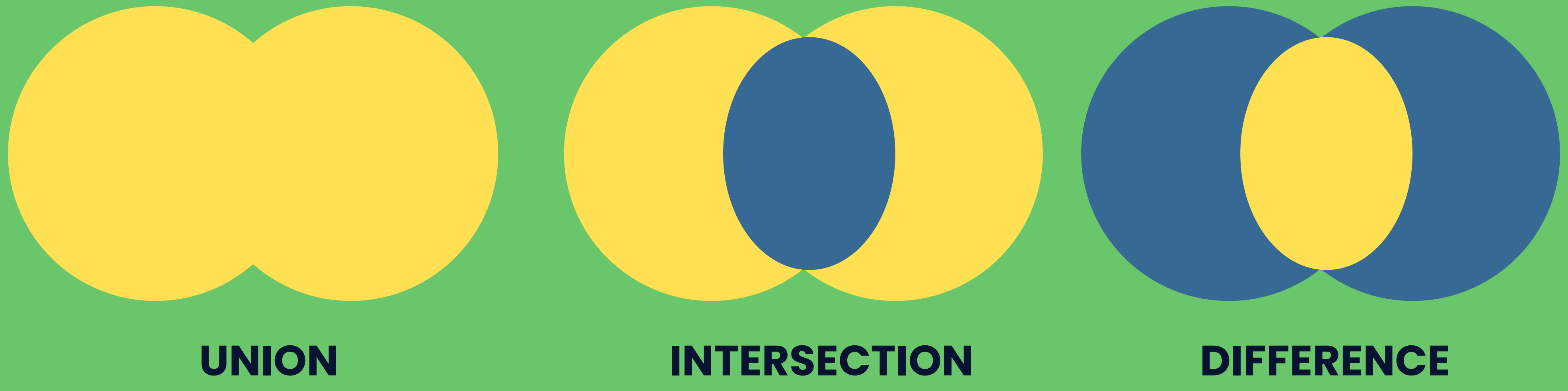
```
my_set = {1, 2, 3, 3}
print(my_set)
```

Creating an Empty Set:

```
my_set = set()  # Not {}
```

- Define a variable to store the set.
- Define a set using curly braces {} or the set() function.
- my_set = {1, 2, 3, 3} # Duplicates are automatically removed
- print(my_set) # {1, 2, 3}
- Print the set to verify its contents.

Set Operations



Union (|) → Combines sets

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
print(set1 | set2)  # {1, 2, 3, 4, 5}
```

Intersection (&) → Finds common elements

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
print(set1 & set2)  # {3}
```

Difference (-) → Finds elements in one set but not another

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
print(set1 - set2)  # {1, 2}
```

Adding & Removing Elements

.add()

```
my_set = {"apple", "banana"}

# Add a single element
my_set.add("orange")
print(my_set)  # {'apple', 'banana', 'orange'}

# Add multiple elements
my_set.update(["grape", "mango"])
print(my_set)  # {'apple', 'banana', 'orange', 'grape',
'mango'}
```

.remove()

```
my_set = {'apple', 'banana', 'orange', 'grape', 'mango'}

# remove a single element
my_set.remove("banana")
print(my_set)  # {'apple', 'orange', 'grape', 'mango'}

# Remove all elements
my_set.clear()
print(my_set)  # set()
```

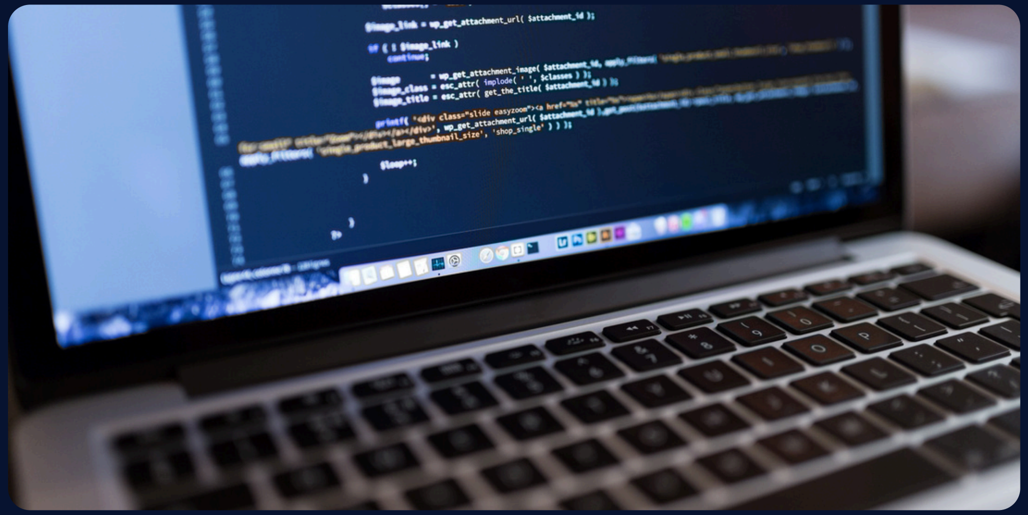
Set Methods

- .copy() → Returns a copy of the set
- .pop() → Removes and returns an arbitrary element
- .clear() → Removes all elements
- .issubset(set2) → Checks if set1 is a subset of set2
- .issuperset(set2) → Checks if set1 is a superset of set2

Use Cases of Sets

Use Cases of Sets

- Removing Duplicates → Extract unique elements from a list
- Mathematical Operations → Union, intersection, and difference
- Efficient Lookups → Checking membership in constant time (O(1))
- Data Filtering → Eliminating duplicate values from large datasets



Thank You

📌 Created by: Prachiti Salvi

🔗 GitHub: @theprachitisalvi

🔗 LinkedIn: Prachiti Salvi