#lec #3.4

Rising Codeas Goa

12) Frozenset
13) dict
14) None

} datatypes

- creating function
- values()
- items()
- formatted string

12) **Frozenset**

"immutable version of a set."

- similar to set but can't modify.

```
fruits = frozenset(["apple", "banana", "orange"])

for fruit in fruits:
    print(fruit)


# Immutable will raise error
fruits.add("grape")
```

Set operations on Frozenset

- union (|)
- Intersection (&)
- difference (-)
- Symmetric difference (^)

```
fruits1 = frozenset(["apple", "banana", "orange"])
fruits2 = frozenset(["banana", "grape", "kiwi"])

print("Union", fruits1 | fruits2)
print("Intersection", fruits1 & fruits2)
print("difference", fruits1 - fruits2)
print("difference2", fruits2 - fruits1)
print("Symmetric difference", fruits1 ^ fruits2)
```

13) **dict datatype**

"collection of key-value pairs"

- defined using braces '{}'
- key-value pairs separated by colons.

```
# Creating a dictionary of fruits and
# their quantities

fruits_dict = {
    "apple": 5,
    "banana": 10,
    "orange": 8
}
```

```
# Accessing quantity.
print("Quantity of apples:", fruits_dict["apple"])
print("Quantity of banana:", fruits_dict["banana"])
print("Quantity of orange:", fruits_dict["orange"])

# Adding new fruits & its quantity
fruits_dict["grape"] = 15

# Modifying the quantity of bananas
fruits_dict["banana"] = 12

# Removing "orange" entry
del fruits_dict["orange"]

# Iterating through keys
for fruit in fruits_dict:
    print(fruit)

# Iterating through values
for quantity in fruits_dict.values():
    print(quantity)

# Iterating through key-value pairs
for fruit, quantity in fruits_dict.items():
    print(fruit, quantity)
```

```
# checking for key Existence
if "apple" in fruits_dict:
    print("Apple in dictionary")
else:
    print("Apple not in dictionary")
```

14) **None datatype**

"absence of a value or lack of a value. It is often used to indicate that a variable or function does not return any value."

```
def greet(name):
    if name is None:
        print("Hello, anonymous user!")
    else:
        print(f"Hello, {name}!")

# calling function with & without name.
greet("Alice")
greet(None)
```

⇒ created function `greet` that takes name as parameter

⇒ we check if name is None using 'is' operator.

⇒ `print(f"Hello, {name}! ")` ⇒ Formatted string.

→ {name} is a placeholder for value of name variable.

→ name is variable that holds the name passed an arguments to the greet function