

Welcome to python Course for Artificial Intelligence

This course consists of,

1. Python Basic
2. Python Intermediate
3. Python Advance

**Python Basic:**

**Variables**: Containers for storing data values.

* x = 5
* y = "Hello"

**Data Types:** Categories of data that tell the interpreter how the programmer intends to use the data.

int: Integer numbers

**Num =**

**Pi =**

**Name =**

**Status =**

* num = 10

float: Floating point numbers

* pi = 3.14

str: String (text)

* name = "Alice"

bool: Boolean (True or False)

* is\_student = True

list: Ordered collection

* numbers = [1, 2, 3]

tuple: Ordered, immutable collection

* point = (10, 20)

dict: Unordered, key-value pairs

* person = {"name": "Bob", "age": 25}

**Program 1: Simple Calculator**

This program takes two numbers and performs basic arithmetic operations.

**v# Simple Calculator**

**num1 = 10**

**num2 = 5**

**addition = num1 + num2**

**subtraction = num1 - num2**

**multiplication = num1 \* num2**

**division = num1 / num2**

**print(f"Addition: {addition}")**

**print(f"Subtraction: {subtraction}")**

**print(f"Multiplication: {multiplication}")**

**print(f"Division: {division}")**



**Program 2: Greeting Message**

This program takes a user's name and prints a personalized greeting.

**# Greeting Message**

**name = "Alice"**

**greeting = f"Hello, {name}! Welcome to Python programming."**

**print(greeting)**

**Program 3: List Operations**

This program demonstrates basic list operations like appending, removing, and accessing elements.

**# List Operations**

**fruits = ["apple", "banana", "cherry"]**

**# Adding an element**

**fruits.append("orange")**

**# Removing an element**

**fruits.remove("banana")**

**# Accessing elements**

**first\_fruit = fruits[0]**

**last\_fruit = fruits[-1]**

**print(f"Fruits List: {fruits}")**

**print(f"First Fruit: {first\_fruit}")**

**print(f"Last Fruit: {last\_fruit}")**



**Variables Rules:**

1 Must start with a letter or underscore (\_): Variable names cannot start with a number.

* Valid: variable, \_variable
* Invalid: 1variable

2 Can only contain alphanumeric characters and underscores: No spaces or special characters (except underscore).

* Valid: variable\_1, var\_name
* Invalid: variable-name, var name

3 Case-sensitive: Variable and variable are different variables.

* Example: Variable ≠ variable

4 Cannot be a reserved keyword: You cannot use Python keywords as variable names (e.g., if, else, while, etc.).

* Invalid: if, for, True

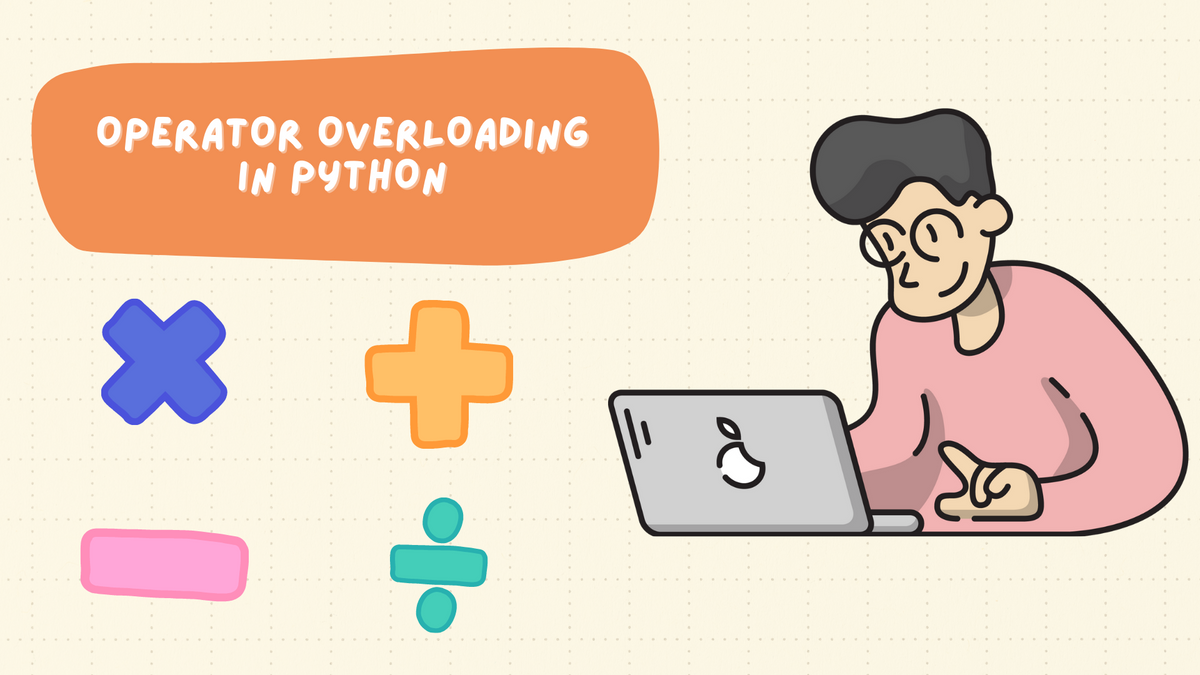
5 Should be descriptive: While not a strict rule, it's good practice to use meaningful names that describe the purpose of the variable.

* Example: count, total\_sum

**Python Basic**

**Operators In python**

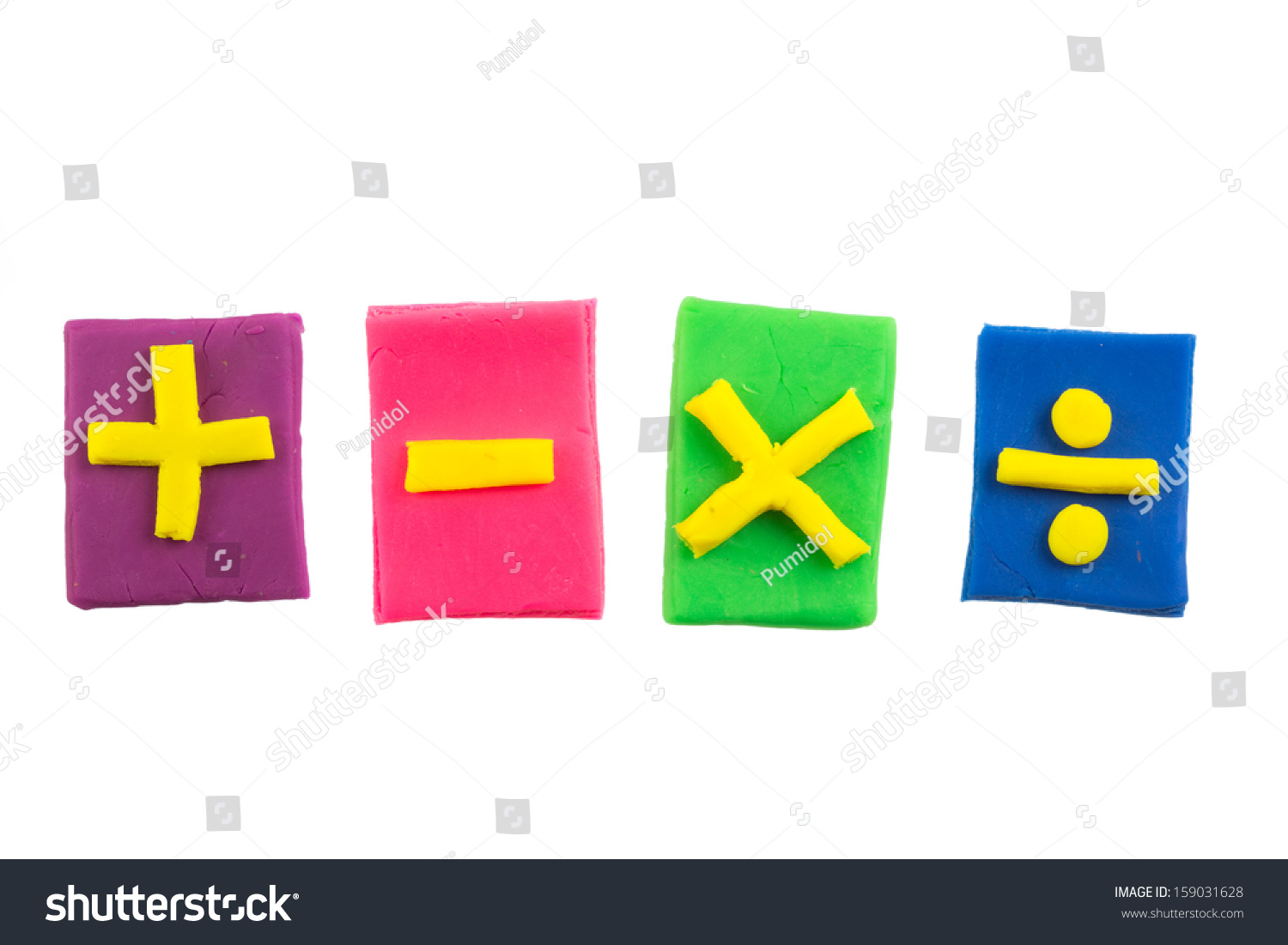
In Python, an operator is a symbol that performs an operation on one or more operands, like + for addition.



Example: 3 + 5 uses the + operator to add the numbers 3 and 5.

Usage: Operators simplify code by performing mathematical, logical, or bitwise operations concisely.

**1 Arithmetic Operators:**





Program:

**# Define variables**

**a = 15**

**b = 4**

**# Perform arithmetic operations**

**sum\_result = a + b**

**diff\_result = a - b**

**product\_result = a \* b**

**quotient\_result = a / b**

**floor\_div\_result = a // b**

**modulus\_result = a % b**

**exponent\_result = a \*\* b**

**# Display results**

**print(f"The sum of {a} and {b} is {sum\_result}")**

**print(f"The difference between {a} and {b} is {diff\_result}")**

**print(f"The product of {a} and {b} is {product\_result}")**

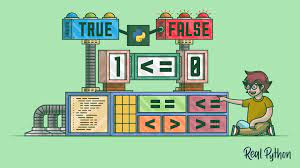
**print(f"The quotient of {a} divided by {b} is {quotient\_result}")**

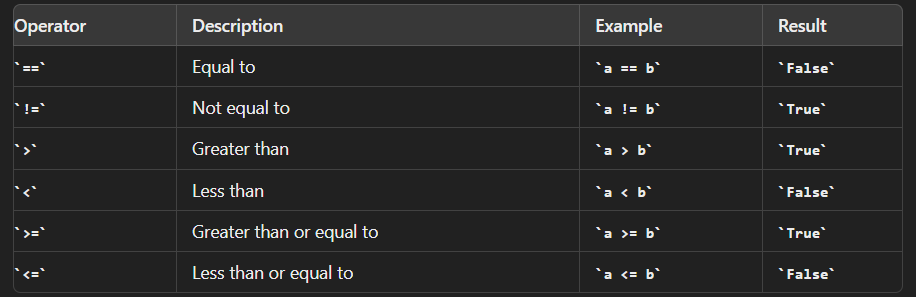
**print(f"The floor division of {a} by {b} is {floor\_div\_result}")**

**print(f"The modulus of {a} and {b} is {modulus\_result}")**

**print(f"The result of {a} raised to the power of {b} is {exponent\_result}")**

**2 Comparison Operators:**





Program

**# Define variables**

**a = 15**

**b = 4**

**# Perform comparison operations**

**equal\_result = (a == b)**

**not\_equal\_result = (a != b)**

**greater\_than\_result = (a > b)**

**less\_than\_result = (a < b)**

**greater\_equal\_result = (a >= b)**

**less\_equal\_result = (a <= b)**

**# Display results**

**print(f"Is {a} equal to {b}? {equal\_result}")**

**print(f"Is {a} not equal to {b}? {not\_equal\_result}")**

**print(f"Is {a} greater than {b}? {greater\_than\_result}")**

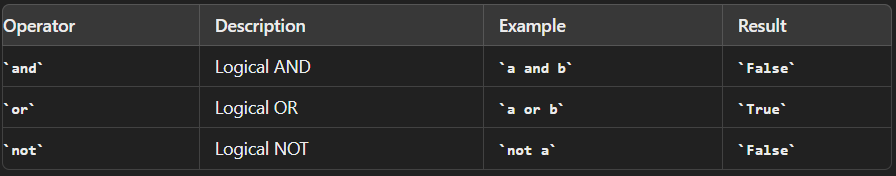
**print(f"Is {a} less than {b}? {less\_than\_result}")**

**print(f"Is {a} greater than or equal to {b}? {greater\_equal\_result}")**

**print(f"Is {a} less than or equal to {b}? {less\_equal\_result}")**

**3 Logical Operators:**





Program

**# Define variables**

**a = True**

**b = False**

**# Perform logical operations**

**and\_result = a and b**

**or\_result = a or b**

**not\_a\_result = not a**

**not\_b\_result = not b**

**# Display results**

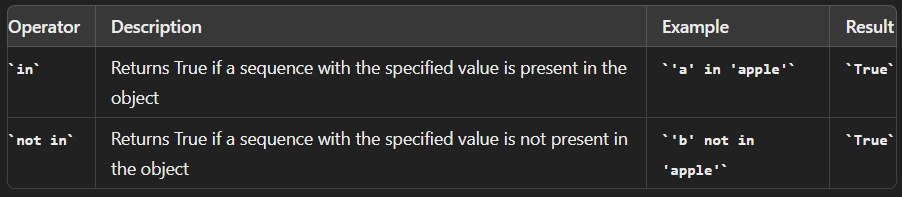
**print(f"The result of {a} and {b} is {and\_result}")**

**print(f"The result of {a} or {b} is {or\_result}")**

**print(f"The result of not {a} is {not\_a\_result}")**

**print(f"The result of not {b} is {not\_b\_result}")**

**4 Membership Operators**



**# Define variables**

**sequence = [1, 2, 3, 4, 5]**

**element1 = 3**

**element2 = 6**

**# Perform membership operations**

**in\_result1 = element1 in sequence**

**in\_result2 = element2 in sequence**

**not\_in\_result1 = element1 not in sequence**

**not\_in\_result2 = element2 not in sequence**

**# Display results**

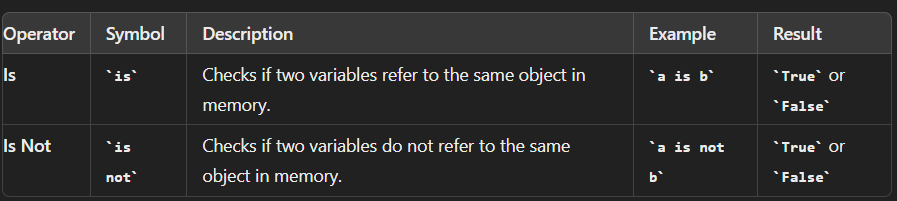
**print(f"Is {element1} in {sequence}? {in\_result1}")**

**print(f"Is {element2} in {sequence}? {in\_result2}")**

**print(f"Is {element1} not in {sequence}? {not\_in\_result1}")**

**print(f"Is {element2} not in {sequence}? {not\_in\_result2}")**

**5 Identity Operators**



**# Example variables**

**a = [1, 2, 3]**

**b = [1, 2, 3]**

**c = a**

**# Using `is`**

**print(a is b) # False, as a and b are different objects**

**print(a is c) # True, as a and c refer to the same object**

**# Using `is not`**

**print(a is not b) # True, as a and b are different objects**

**print(a is not c) # False, as a and c refer to the same object**

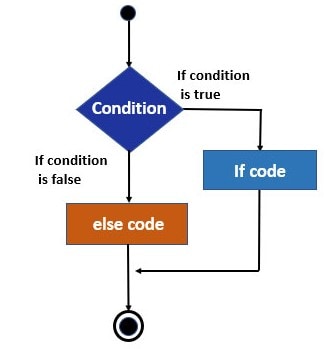
Python Basic:

**3 Control Flow Statement in python**

Control flow statements are used to determine the flow of execution of a program based on certain conditions.

**If-Else Statement**

**Used to test two conditions**



**# is it apple or something else.**

**item = “Apple”**

**if item == “Apple”:**

**print(f“items is {item} “)**

**else:**

**print(f”item is something else”)**

**Nested if-else**

If within another if body for testing conditions

**x = 20**

**if x > 10:**

**print("x is greater than 10")**

**if x > 15:**

**print("x is also greater than 15")**

**else:**

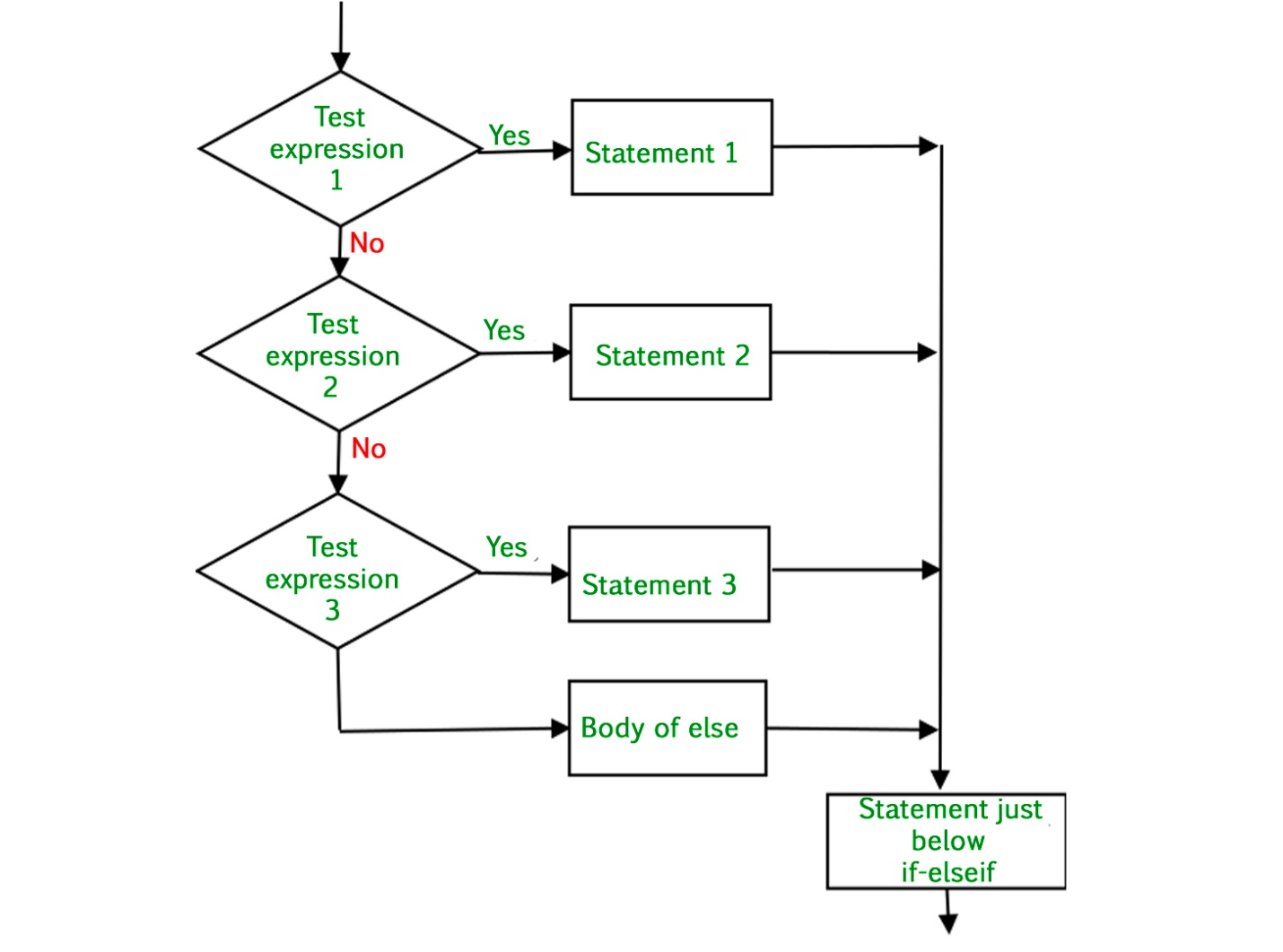
**print("x is not greater than 15")**

**else:**

**print("x is not greater than 10")**

**if-elif-else**

Statement: Executes different blocks of code for multiple conditions.



**# Simple Calculator using if-elif-else**

**print("Select operation:")**

**print("1. Addition")**

**print("2. Subtraction")**

**print("3. Multiplication")**

**print("4. Division")**

**# Take input from the user**

**choice = input("Enter choice (1/2/3/4): ")**

**# Check if choice is one of the four options**

**if choice ==’1’**

**num1 = float(input("Enter first number: "))**

**num2 = float(input("Enter second number: "))**

**print(f”Addition of {num1} and {num2} = {num1 + num2} “)**

**elif choice ==’2’:**

**num1 = float(input("Enter first number: "))**

**num2 = float(input("Enter second number: "))**

**print(f”Substraction of {num1} and {num2} = {num1 - num2} “)**

**elif choice ==’3’:**

**num1 = float(input("Enter first number: "))**

**num2 = float(input("Enter second number: "))**

**print(f”Mult of {num1} and {num2} = {num1 \* num2} “)**

**elif choice ==’4’**

**num1 = float(input("Enter first number: "))**

**num2 = float(input("Enter second number: "))**

**print(f”Addition of {num1} and {num2} = {num1 + num2} “)**

**else:**

**print("Invalid input")**

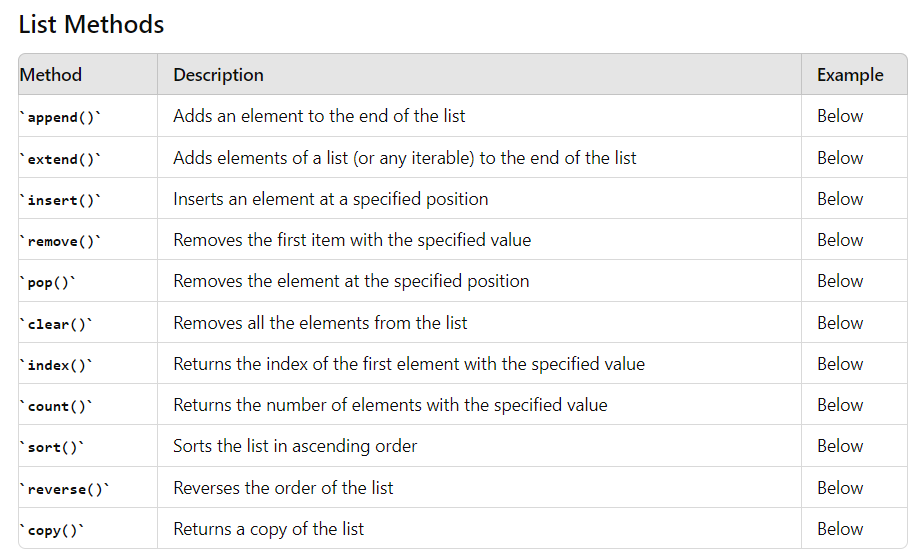
**Data Structure in Python:**

1 list and its methods

2 tuples its methods

3 dictionaries its methods

List Methods:



**Example Programs for List Methods**

**1 append()**

fruits = ["apple", "banana", "cherry"]

fruits.append("orange")

print(fruits) # Output: ['apple', 'banana', 'cherry', 'orange']

**2 extend()**

fruits = ["apple", "banana", "cherry"]

more\_fruits = ["orange", "mango"]

fruits.extend(more\_fruits)

print(fruits) # Output: ['apple', 'banana', 'cherry', 'orange', 'mango']

**3 insert()**

fruits = ["apple", "banana", "cherry"]

fruits.insert(1, "orange")

print(fruits) # Output: ['apple', 'orange', 'banana', 'cherry']

**4 remove()**

fruits = ["apple", "banana", "cherry"]

fruits.remove("banana")

print(fruits) # Output: ['apple', 'cherry']

**5 pop()**

fruits = ["apple", "banana", "cherry"]

fruits.pop(1)

print(fruits) # Output: ['apple', 'cherry']

**6 clear()**

fruits = ["apple", "banana", "cherry"]

fruits.clear()

print(fruits) # Output: []

**7 index()**

fruits = ["apple", "banana", "cherry"]

index = fruits.index("banana")

print(index) # Output: 1

**8 count()**

fruits = ["apple", "banana", "cherry", "banana"]

count = fruits.count("banana")

print(count) # Output: 2

**9 sort()**

fruits = ["cherry", "banana", "apple"]

fruits. Sort()

print(fruits) # Output: ['apple', 'banana', 'cherry']

**10 reverse()**

fruits = ["apple", "banana", "cherry"]

fruits.reverse()

print(fruits) # Output: ['cherry', 'banana', 'apple']

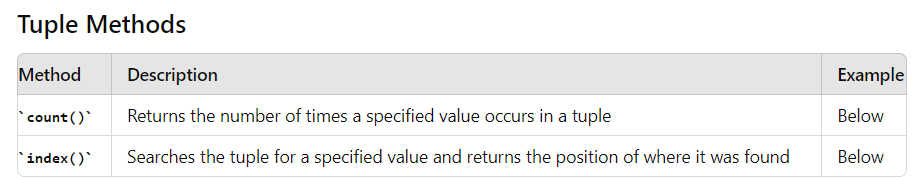
**11 copy()**

fruits = ["apple", "banana", "cherry"]

new\_fruits = fruits.copy()

print(new\_fruits) # Output: ['apple', 'banana', 'cherry']

Tuple In python



**Example Programs for Tuple Methods**

**count()**

numbers = (1, 2, 3, 2, 2, 4)

count = numbers.count(2)

print(count) # Output: 3

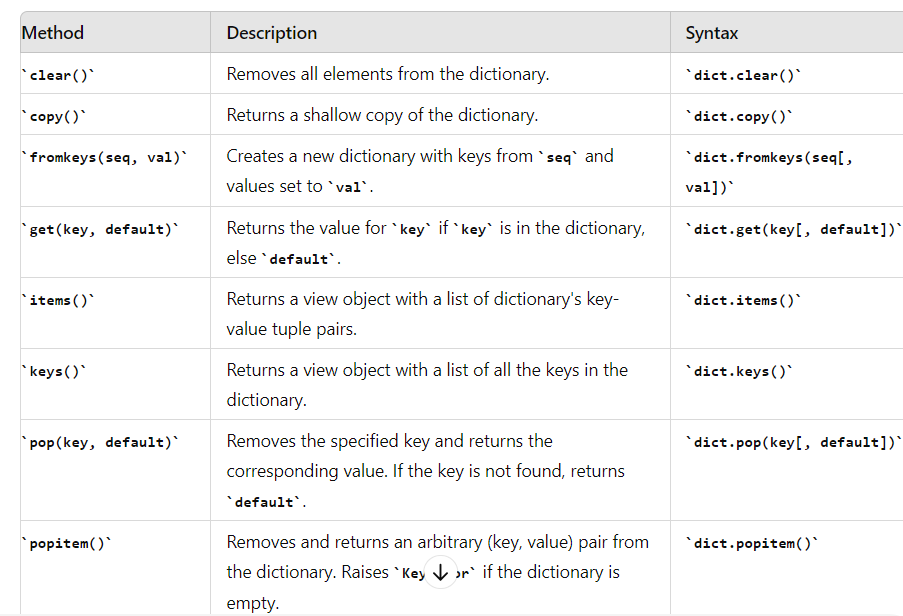
**index()**

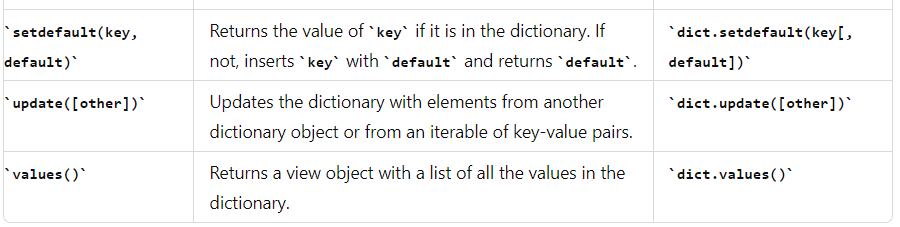
numbers = (1, 2, 3, 4, 2)

index = numbers.index(2)

print(index) # Output: 1

**Dictionary in python**





**Example Programs for Dictionary Methods**

**clear()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

capitals.clear()

print(capitals) # Output: {}

**copy()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

new\_capitals = capitals.copy()

print(new\_capitals) # Output: {'France': 'Paris', 'Germany': 'Berlin', 'Italy': 'Rome'}

**fromkeys()**

keys = ('a', 'b', 'c')

value = 0

new\_dict = dict.fromkeys(keys, value)

print(new\_dict) # Output: {'a': 0, 'b': 0, 'c': 0}

**get()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

capital = capitals.get("Germany")

print(capital) # Output: Berlin

**items()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

items = capitals.items()

print(items) # Output: dict\_items([('France', 'Paris'), ('Germany', 'Berlin'), ('Italy', 'Rome')])

**keys()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

keys = capitals.keys()

print(keys) # Output: dict\_keys(['France', 'Germany', 'Italy'])

**pop()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

removed\_value = capitals.pop("Germany")

print(removed\_value) # Output: Berlin

print(capitals) # Output: {'France': 'Paris', 'Italy': 'Rome'}

**popitem()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

last\_item = capitals.popitem()

print(last\_item) # Output: ('Italy', 'Rome')

print(capitals) # Output: {'France': 'Paris', 'Germany': 'Berlin'}

**setdefault()**

capitals = {"France": "Paris", "Germany": "Berlin"}

value = capitals.setdefault("Italy", "Rome")

print(value) # Output: Rome

print(capitals) # Output: {'France': 'Paris', 'Germany': 'Berlin', 'Italy': 'Rome'}

**update()**

capitals = {"France": "Paris", "Germany": "Berlin"}

new\_entries = {"Italy": "Rome", "Spain": "Madrid"}

capitals.update(new\_entries)

print(capitals) # Output: {'France': 'Paris', 'Germany': 'Berlin', 'Italy': 'Rome', 'Spain': 'Madrid'}

**values()**

capitals = {"France": "Paris", "Germany": "Berlin", "Italy": "Rome"}

values = capitals.values()

print(values) # Output: dict\_values(['Paris', 'Berlin', 'Rome'])

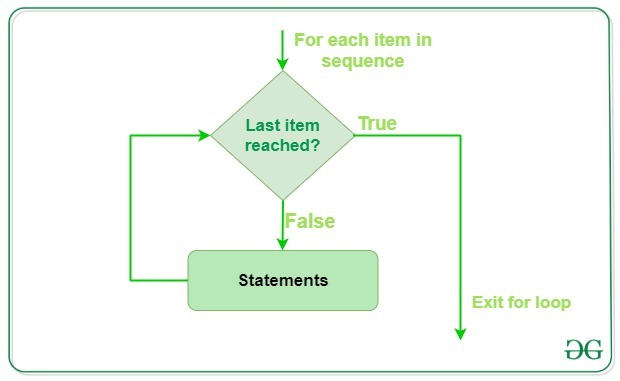
**Loops and Functions**

**Loops:**

The loop is a set of statements that are used to execute a set of statements more than one time.

**For Loop**

 a control flow statement that is used to repeatedly execute a group of statements as long as the condition is satisfied



**Program 1: For Loop with a List**

**# List of numbers**

**numbers = [1, 2, 3, 4, 5]**

**# Using a for loop to iterate over the list**

**print("List:")**

**for num in numbers:**

**print(num)**

**Program 2: For Loop with a Tuple**

# Tuple of fruits

fruits = ('apple', 'banana', 'cherry')

# Using a for loop to iterate over the tuple

print("\nTuple:")

for fruit in fruits:

print(fruit)

### Program 3: For Loop with a Dictionary

# Dictionary of fruits with their prices

fruit\_prices = {'apple': 0.50, 'banana': 0.30, 'cherry': 0.75}

# Using a for loop to iterate over the dictionary

print("\nDictionary:")

for fruit, price in fruit\_prices.items():

print(f"The price of {fruit} is ${price}")

**Program 3: For loop with range function**

# Using range to generate a sequence of numbers

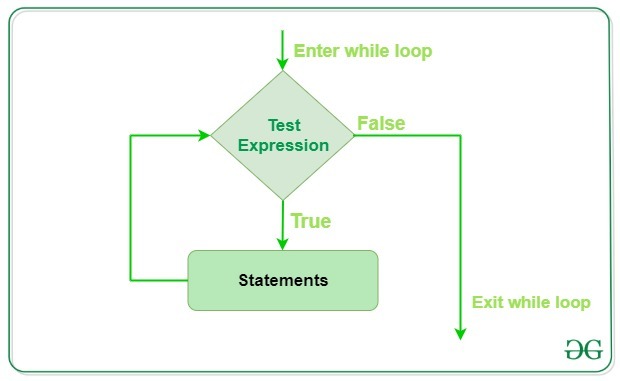
print("\nRange:")

for i in range(1, 6):

print(i)

**While Loop**

a control flow statement that allows a block of code to be executed an indeterminate number of times, so long as the associated condition holds true.



### Program 1: While Loop with a List

# List of numbers

numbers = [1, 2, 3, 4, 5]

# Using a while loop to iterate over the list

print("List:")

index = 0

while index < len(numbers):

print(numbers[index])

index += 1

### Program 2: While Loop with a Tuple

# Tuple of fruits

fruits = ('apple', 'banana', 'cherry')

# Using a while loop to iterate over the tuple

print("\nTuple:")

index = 0

while index < len(fruits):

print(fruits[index])

index += 1

### Program 3: While Loop with a Dictionary

# Dictionary of fruits with their prices

fruit\_prices = {'apple': 0.50, 'banana': 0.30, 'cherry': 0.75}

# Converting dictionary items to a list for iteration

items = list(fruit\_prices.items())

# Using a while loop to iterate over the dictionary

print("\nDictionary:")

index = 0

while index < len(items):

fruit, price = items[index]

print(f"The price of {fruit} is ${price}")

index += 1

### Program 4: While Loop with Range

# Using a while loop to generate a sequence of numbers

print("\nRange:")

i = 1

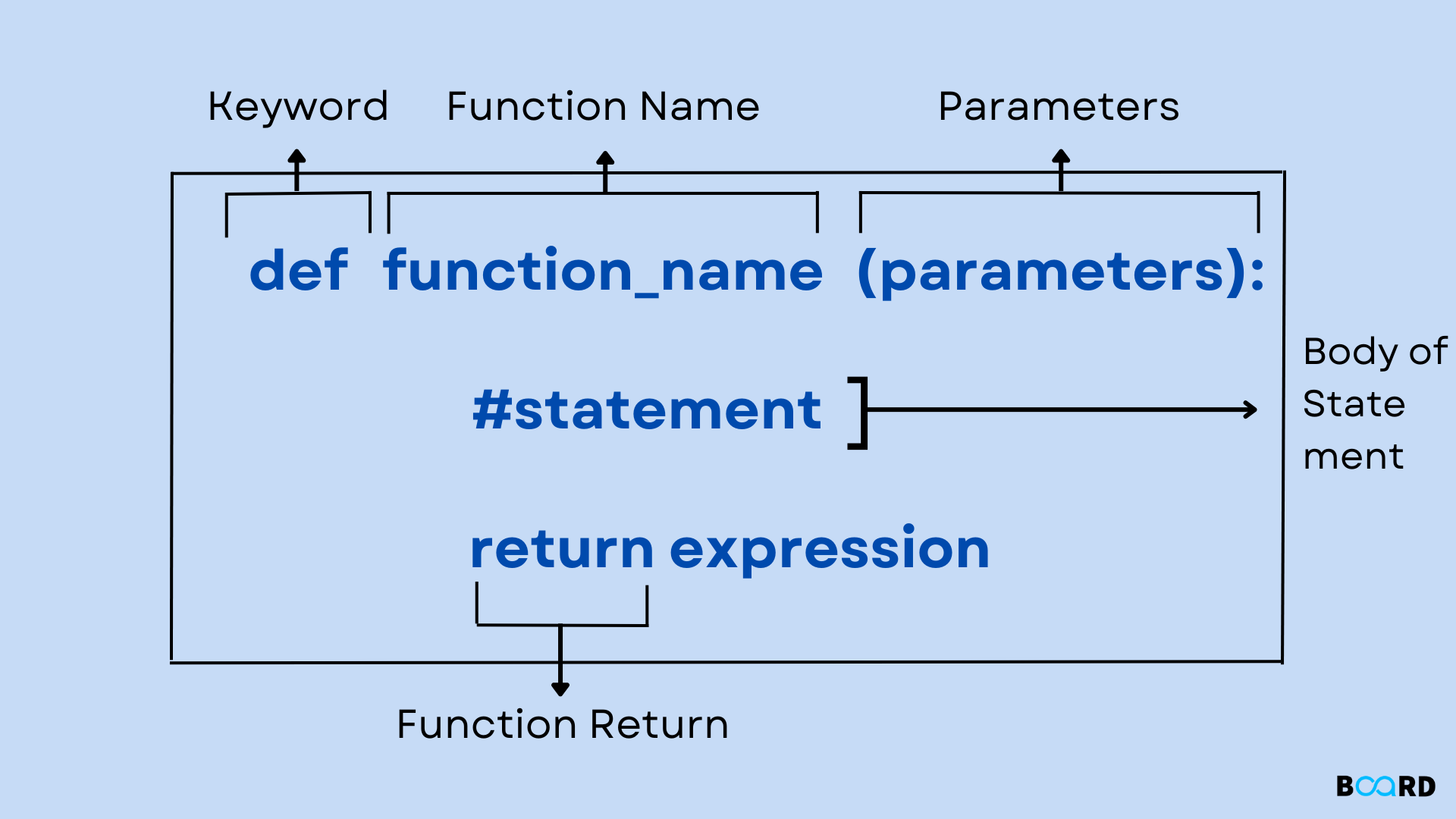
while i < 6:

print(i)

i += 1

**Functions in python**

a block of program statements which can be used repetitively in a program



### Program 1: Function with Parameters and Arguments

def greet(name, age):

print(f"Hello, {name}! You are {age} years old.")

# Calling the function with arguments

greet("Alice", 30)

greet("Bob", 25)

### Program 2: Function with Default Parameter Values

def greet(name, age=18):

print(f"Hello, {name}! You are {age} years old.")

# Calling the function without the age argument (using the default value)

greet("Alice")

# Calling the function with both arguments

greet("Bob", 25)

### Program 3: Function with Keyword Arguments

def describe\_pet(animal\_type, pet\_name):

print(f"I have a {animal\_type} named {pet\_name}.")

# Calling the function with keyword arguments

describe\_pet(animal\_type="dog", pet\_name="Buddy")

describe\_pet(pet\_name="Whiskers", animal\_type="cat")

These examples demonstrate:

1. **Program 1**: Using functions with parameters and arguments.
2. **Program 2**: Using functions with default parameter values.
3. **Program 3**: Using functions with keyword arguments to explicitly specify parameter values.

**String manipulation in python**

**Basic String Operations**

1. **Concatenation**: Combining strings using +

python

Copy code

str1 = "Hello"

str2 = "World"

result = str1 + " " + str2

print(result) # Output: Hello World

1. **Repetition**: Repeating strings using \*

python

Copy code

str1 = "Hello"

result = str1 \* 3

print(result) # Output: HelloHelloHello

1. **String Length**: Using len()

python

Copy code

str1 = "Hello"

print(len(str1)) # Output: 5

### String Methods

1. **upper()**: Converts all characters to uppercase

python

Copy code

str1 = "Hello"

print(str1.upper()) # Output: HELLO

1. **lower()**: Converts all characters to lowercase

python

Copy code

str1 = "Hello"

print(str1.lower()) # Output: hello

1. **capitalize()**: Capitalizes the first character of the string

python

Copy code

str1 = "hello world"

print(str1.capitalize()) # Output: Hello world

1. **title()**: Capitalizes the first character of each word

python

Copy code

str1 = "hello world"

print(str1.title()) # Output: Hello World

1. **strip()**: Removes leading and trailing whitespace

python

Copy code

str1 = " Hello World "

print(str1.strip()) # Output: Hello World

1. **replace()**: Replaces a substring with another substring

python

Copy code

str1 = "Hello World"

print(str1.replace("World", "Python")) # Output: Hello Python

1. **split()**: Splits the string into a list based on a delimiter

python

Copy code

str1 = "Hello World"

print(str1.split()) # Output: ['Hello', 'World']

1. **join()**: Joins elements of a list into a single string with a specified delimiter

python

Copy code

list1 = ["Hello", "World"]

print(" ".join(list1)) # Output: Hello World

1. **find()**: Returns the index of the first occurrence of a substring

python

Copy code

str1 = "Hello World"

print(str1.find("World")) # Output: 6

1. **count()**: Counts the occurrences of a substring

python

Copy code

str1 = "Hello World"

print(str1.count("o")) # Output: 2

1. **startswith() and endswith()**: Checks if the string starts or ends with a specific substring

python

Copy code

str1 = "Hello World"

print(str1.startswith("Hello")) # Output: True

print(str1.endswith("World")) # Output: True

### Advanced String Methods

1. **format()**: Formats strings using placeholders

python

Copy code

name = "Noor"

age = 25

str1 = "My name is {} and I am {} years old.".format(name, age)

print(str1) # Output: My name is Noor and I am 25 years old.

1. **f-strings**: Formatted string literals (Python 3.6+)

python

Copy code

name = "Noor"

age = 25

str1 = f"My name is {name} and I am {age} years old."

print(str1) # Output: My name is Noor and I am 25 years old.

1. **zfill()**: Pads the string on the left with zeros

python

Copy code

str1 = "42"

print(str1.zfill(5)) # Output: 00042

1. **ljust() and rjust()**: Left-justifies or right-justifies the string in a field of a given width

python

Copy code

str1 = "Hello"

print(str1.ljust(10, '-')) # Output: Hello-----

print(str1.rjust(10, '-')) # Output: -----Hello

1. **partition()**: Splits the string into a tuple of three elements

python

Copy code

str1 = "Hello World"

print(str1.partition(" ")) # Output: ('Hello', ' ', 'World')

1. **maketrans() and translate()**: Mapping characters for translation

python

Copy code

intab = "aeiou"

outtab = "12345"

trantab = str.maketrans(intab, outtab)

str1 = "This is a test"

print(str1.translate(trantab)) # Output: Th3s 3s 1 t2st

### Example Programs

1. **Reverse a String**

python

Copy code

def reverse\_string(s):

return s[::-1]

print(reverse\_string("Hello World")) # Output: dlroW olleH

1. **Check if a String is a Palindrome**

python

Copy code

def is\_palindrome(s):

s = s.lower()

return s == s[::-1]

print(is\_palindrome("Madam")) # Output: True

print(is\_palindrome("Hello")) # Output: False

1. **Count Vowels in a String**

python

Copy code

def count\_vowels(s):

vowels = "aeiouAEIOU"

count = 0

for char in s:

if char in vowels:

count += 1

return count

print(count\_vowels("Hello World")) # Output: 3

1. **Remove Duplicates from a String**

python

Copy code

def remove\_duplicates(s):

return "".join(sorted(set(s), key=s.index))

print(remove\_duplicates("Hello World")) # Output: Helo Wrd