7. Functions and Methods

❖ Defining and calling functions in Python.

In Python, functions are blocks of reusable code designed to perform specific tasks. They help in organizing code, reducing redundancy, and enhancing clarity. Here's a comprehensive guide on defining and calling functions in Python:

Defining a Function

To define a function, use the def keyword followed by the function name and parentheses. The code block within the function is indented.

Syntax:

```
def function_name(parameters):
    # code block
```

Calling a Function

To execute a function, simply call it by its name followed by parentheses.

Syntax:

```
function_name()
```

Functions with Parameters

Functions can accept parameters (also known as arguments) to perform operations using the provided values.

Syntax:

```
def function_name(param1, param2):
    # code block
```

❖Function arguments (positional, keyword, default).

1. Positional Arguments

- **Definition:** These are the most common type of arguments. The values are assigned to parameters based on their position in the function call.
- **Usage:** When calling a function, the arguments must be provided in the exact order as the parameters are defined.

2. Keyword Arguments

- **Definition:** Arguments are passed by explicitly specifying the parameter names, allowing them to be provided in any order.
- **Usage:** Enhances code readability and is especially useful when a function has multiple parameters.

3. Default Arguments

- **Definition:** Parameters can have default values. If a value is provided during the function call, it overrides the default; otherwise, the default value is used.

- **Usage:** Useful for parameters that often have the same value, reducing the need to specify them every time.

❖ Scope of variables in Python.

In Python, the scope of a variable determines where it can be accessed or modified. Python follows the LEGB rule to resolve variable names, which stands for Local, Enclosing, Global, and Built-in scopes..

Types of Variable Scopes

1. Local Scope

Variables declared inside a function are local to that function and cannot be accessed outside it.

2. Global Scope

Variables declared at the top level of a script or module are global and can be accessed from anywhere within the module.

3. Nonlocal Scope

The nonlocal keyword allows a nested function to modify a variable in its enclosing (but non-global) scope.

4. Built-in Scope

This scope contains Python's built-in functions and exceptions. If a variable name conflicts with a built-in name, the local or global variable takes precedence.

❖ Built-in methods for strings, lists, etc.

In Python, strings and lists are two of the most commonly used data types, and they come with a rich set of built-in methods that make data manipulation straightforward and efficient. Here's an overview of the most important built-in methods for strings and lists, along with their descriptions and examples.

String Methods

Python strings are immutable sequences of characters, and they offer a wide array of methods for text processing.

Common String Methods :

Method	Description
.capitalize()	Converts the first character to uppercase.
.casefold()	Converts the string to lowercase for caseless comparisons.
.center(width)	Centers the string within a specified width, padding with spaces.
.count(sub)	Returns the number of occurrences of substring sub.
.encode(encoding)	Encodes the string using the specified encoding.
.endswith(suffix)	Returns True if the string ends with the specified suffix.
.expandtabs(tabsize)	Replaces tab characters with spaces, using tabsize spaces per tab.
.find(sub)	Returns the lowest index where substring sub is found, or -1 if not found.
.index(sub)	Like .find(), but raises a ValueError if the substring is not found.
.isalnum()	Returns True if all characters are alphanumeric.
.isalpha()	Returns True if all characters are alphabetic.

Method	Description
.isdecimal()	Returns True if all characters are decimal characters.
.isdigit()	Returns True if all characters are digits.
.isidentifier()	Returns True if the string is a valid identifier.
.islower()	Returns True if all characters are lowercase.
.isnumeric()	Returns True if all characters are numeric characters.
.isprintable()	Returns True if all characters are printable or the string is empty.
.isspace()	Returns True if all characters are whitespace characters.
.istitle()	Returns True if the string is title cased.
.isupper()	Returns True if all characters are uppercase.
.join(iterable)	Concatenates elements of iterable into a single string, separated by the string.
.ljust(width, fillchar)	Left-justifies the string within a specified width, padding with fillchar.
.lower()	Converts all characters to lowercase.
.lstrip(chars)	Removes leading characters in chars from the string.

Method	Description
.replace(old, new)	Returns a copy of the string with all occurrences of old replaced by new.
.rfind(sub)	Returns the highest index where substring sub is found, or -1 if not found.
.rindex(sub)	Like .rfind(), but raises a ValueError if the substring is not found.
.rjust(width, fillchar)	Right-justifies the string within a specified width, padding with fillchar.
.rstrip(chars)	Removes trailing characters in chars from the string.
.split(sep)	Splits the string into a list of substrings based on the delimiter sep.
.splitlines()	Splits the string at line boundaries.
.startswith(prefix)	Returns True if the string starts with the specified prefix.
.strip(chars)	Removes leading and trailing characters in chars from the string.
.swapcase()	Swaps case of all characters.
.title()	Converts the first character of each word to uppercase.

Method	Description
.translate(table)	Returns a copy of the string with characters mapped through table.
.upper()	Converts all characters to uppercase.
.zfill(width)	Pads the string on the left with zeros to make it width characters long.

List Methods

Python lists are mutable sequences, commonly used to store collections of homogeneous items.

Common List Methods :

Method	Description
.append(x)	Adds an item x to the end of the list.
.clear()	Removes all items from the list.
.copy()	Returns a shallow copy of the list.
.count(x)	Returns the number of occurrences of item x in the list.

Method	Description
.extend(iterable)	Extends the list by appending all the items from the iterable.
.index(x)	Returns the index of the first occurrence of item x.
.insert(i, x)	Inserts item x at position i.
.pop([i])	Removes and returns the item at position i. If no index is specified, removes and returns the last item.
.remove(x)	Removes the first occurrence of item x.
.reverse()	Reverses the elements of the list in place.
.sort()	Sorts the items of the list in place.