1. Build-in Function in Python

Built-in functions in Python are pre-defined functions that come with the language and can be used without the need to declare or define them. They serve various purposes, from mathematical operations to data manipulation and input/output. Examples of built-in functions include print() for displaying output, len() to find the length of objects like lists, and sorted() for sorting iterables.

In [1]:
```
print("\nBUILT-IN FUNCTIONS\n" + "-"*40)
```

```
BUILT-IN FUNCTIONS
----------------------------------------
```

Using print() to display a message

In [2]:
```
print("Hello, World!")
```

```
Hello, World!
```

Calculating the length of a list using len()

In [3]:
```
sample_list = [1, 2, 3, 4, 5]
length_of_list = len(sample_list)
print(f"Length of the sample list: {length_of_list}")
```

```
Length of the sample list: 5
```

Using sorted() to sort a list

In [4]:
```
sorted_list = sorted(sample_list, reverse=True)
print(f"Sorted list in descending order: {sorted_list}")
```

```
Sorted list in descending order: [5, 4, 3, 2, 1]
```

2. Library Function in Python

Library functions in Python are organized sets of code provided by external modules or packages to perform specific tasks. Unlike built-in functions, they need to be imported before use. These functions expand Python's capabilities, allowing developers to achieve complex operations without writing code from scratch. Popular libraries like `math` for mathematical operations and `os` for operating system interactions offer numerous such functions.

In [5]:
```
print("\nLIBRARY FUNCTIONS\n" + "-"*40)
```

```
LIBRARY FUNCTIONS
----------------------------------------
```

In [6]:
```
# Loading the math library to implement the library function in pythor
import math
```

Using sqrt() from math library to calculate square root

In [8]:
```
num = 25
sqrt_num = math.sqrt(num)
print(f"Square root of {num} is: {sqrt_num}")
```

```
Square root of 25 is: 5.0
```

Using factorial() from math library

In [7]:
```
fact = math.factorial(5)
print(f"Factorial of 5 is: {fact}")
```

```
Factorial of 5 is: 120
```

3.User Defined Function

User-defined functions in Python are custom functions created by programmers to perform specific tasks. They allow for code reusability and modular programming. Defined using the `def` keyword, these functions can accept parameters and return values. They enhance code clarity by breaking down complex problems into smaller, manageable tasks, enabling more organized and efficient coding.

In [10]:
```
print("\nUSER-DEFINED FUNCTIONS\n" + "-"*40)
```

```
USER-DEFINED FUNCTIONS
----------------------------------------
```

Defining a simple function to add two numbers

In [8]:
```
def add(a, b):
    return a + b
```

Using the user-defined function

In [12]:
```
result = add(10, 20)
print(f"Result of addition: {result}")
```

```
Result of addition: 30
```

Defining a function to check if a number is even

In [9]:
```
def is_even(num):
    return num % 2 == 0
```

Using the user-defined function

In [10]:
```
number = 7
if is_even(number):
    print(f"{number} is even.")
else:
    print(f"{number} is odd.")
```

```
7 is odd.
```

In [ ]:
```

```