***Introduction to the print() function in Python***

The print() function in Python is a built-in function used to display output to the console or other standard output devices. It is one of the most fundamental functions for interacting with users and debugging code.

***Formatting outputs using f-strings and format()***

Python provides several ways to format output, with f-strings (formatted string literals) and the format() method being two of the most popular and versatile.

F-strings (Formatted String Literals)

F-strings were introduced in Python 3.6 and offer a concise and readable way to embed expressions inside string literals.

Syntax: Prefix the string literal with f or F, and place expressions within curly braces {} inside the string.

Embedding Variables and Expressions:

Python

name = "Alice"

age = 30

print(f"My name is {name} and I am {age} years old.")

# Output: My name is Alice and I am 30 years old.

result = 10 \* 5

print(f"The result of 10 \* 5 is {result}.")

# Output: The result of 10 \* 5 is 50.

***Converting user input into different data types (e.g., int, float, etc.).***

Converting user input into different data types, such as integers (int), floating-point numbers (float), or other types, is a common requirement in programming. This process is often called type conversion or typecasting.

The input() function in many languages, like Python, typically reads user input as a string. Therefore, if numerical operations are required, the string input must be explicitly converted to a numerical data type.

Here are examples of how to convert user input to int and float in Python:

1. Converting to an Integer (int):

Python

# Prompt the user for input

user\_input\_str = input("Enter an integer: ")

# Convert the string input to an integer

try:

user\_input\_int = int(user\_input\_str)

print(f"You entered the integer: {user\_input\_int}")

print(f"Data type: {type(user\_input\_int)}")

except ValueError:

print("Invalid input: Please enter a whole number.")

Explanation:

The input() function gets the user's input as a string.

The int() function attempts to convert this string into an integer.

A try-except block is used for error handling, as int() will raise a ValueError if the string cannot be converted (e.g., if the user enters text or a decimal number).

2. Converting to a Floating-Point Number (float):

Python

# Prompt the user for input

user\_input\_str = input("Enter a decimal number: ")

# Convert the string input to a float

try:

user\_input\_float = float(user\_input\_str)

print(f"You entered the float: {user\_input\_float}")

print(f"Data type: {type(user\_input\_float)}")

except ValueError:

print("Invalid input: Please enter a decimal number.")

Explanation:

Similar to int(), the float() function converts the string input to a floating-point number.

The try-except block handles potential ValueError exceptions if the input is not a valid float.

General Principle:

The general principle for converting user input to other data types involves using the appropriate built-in type conversion functions (e.g., str(), bool(), list(), tuple(), set() for other types) and implementing robust error handling to manage cases where the input does not conform to the expected data type.

***Opening files in different modes ('r', 'w', 'a', 'r+', 'w+')***

Yes, you can open a file for both reading and writing by using one of the following modes:

'r+' : Opens the file for both reading and writing. The file pointer is at the beginning of the file. ...

'w+' : Opens the file for both writing and reading. ...

'a+' : Opens the file for both appending and reading.

***Using the open() function to create and access files***

By using the open() function, we can open a file in the current directory as well as a file located in a specified location with the help of its path. In this example, we are opening a file "gfg. txt" located in the current directory and "gfg1. txt" located in a specified location.

***Using the open() function to create and access files***

By using the open() function, we can open a file in the current directory as well as a file located in a specified location with the help of its path. In this example, we are opening a file "gfg. txt" located in the current directory and "gfg1. txt" located in a specified location.

***Reading from a file using read(), readline(), readlines()***

In Python, the read(), readline(), and readlines() methods are used to retrieve content from a file object opened in read mode. Each method offers a distinct approach to reading data:

read(size=-1):

This method reads the entire content of the file and returns it as a single string.

If an optional size argument is provided, it reads up to size characters (or bytes in binary mode) from the file.

Subsequent calls to read() will continue from where the previous read left off.