Python print() Function – Concepts and Examples

Study notes based on the concepts discussed

1. What is the print() Function?

The print() function is one of the most commonly used functions in Python. It is generally used to display information on the screen.

```
print("Hello, World!")
Output:
Hello, World!
```

2. Printing Strings

Text values can be displayed by placing them inside quotation marks. Python supports both single quotes and double quotes.

```
print("Beautiful, World!")
print("Cruel, World!")
Output:
Beautiful, World!
Cruel, World!
```

```
print('Hello')
print("Hello")
```

Both statements display the same text.

3. Standard Syntax of print()

The standard syntax of the Python print() function is:

```
print(*objects, sep=' ', end='\n', file=None, flush=False)
print(*objects, sep=' ', end='\n', file=None)
print(*objects, sep=' ', end='\n')
print(*objects, sep=' ')
print(*objects)
```

Note: The * means that print() can accept **zero or more arguments**.

The main parts of the syntax are:

Argument	Meaning
*objects	One or more values to be displayed.
sep	Separator placed between multiple objects. Default: a space (' ').
end	Text placed after the output. Default: a newline ('\n').
file	Specifies the output stream. Default: None.
flush	Controls whether the output is forcibly flushed.

	Default: False.

4. Printing Numbers

```
print(0, 100, 200, 300, 400)
Output:
0 100 200 300 400
```

When multiple objects are passed to print(), Python separates them with a space by default.

5. Printing Decimal Numbers

```
print(0, 100, 200.5, 300, 400.9)
Output:
0 100 200.5 300 400.9
```

The print() function can display integers and floating-point numbers together.

6. Printing Multiple Values

```
print(10, 20, 30, 40)
Output:
10 20 30 40
```

The values are passed as separate arguments to print(). The default separator is a space.

7. The sep Argument

The sep argument specifies the separator used between multiple objects. Its default value is a single space.

```
print(0, 100, 200, 300, 400, sep='-')
Output:
0-100-200-300-400

print(0, 100, 200, 300, 400, sep='$')
Output:
0$100$200$300$400

print(0, 100, 200, 300, 400, sep='^')
Output:
0^100^200^300^400
```

8. sep Can Be Used with Strings

```
print("Welcome", "to", "EngineersTutor.com")
Output:
Welcome to EngineersTutor.com

print("Welcome", "to", "EngineersTutor.com", sep="-")
```

```
Output:
Welcome-to-EngineersTutor.com
```

9. Printing a Variable

```
a = 5
print(a)
Output:
5
```

Here, a is associated with the integer value 5, and print(a) displays that value.

10. Printing Text and a Variable Together

```
a = 5
print("Value of a is:", a)
Output:
Value of a is: 5
```

Two objects are passed to print(): the string 'Value of a is:' and the variable a. The default separator is a space.

11. The end Argument

The end argument specifies what should be printed after the complete output. Its default value is a newline character, '\n'.

```
print("Hello")
print("World")
Output:
Hello
World
```

```
print("Hello", end=" ")
print("World")
Output:
Hello World
```

12. sep and end Together

```
print(10, 20, 30, sep="-", end="!")
Output:
10-20-30!
```

Here, sep='-' controls what appears between the objects, while end='!' controls what appears after the complete output.

13. Difference Between sep and end

sep controls the text placed between multiple objects:

```
print(10, 20, 30, sep="-")
Output:
10-20-30
```

end controls the text placed after the complete output:

```
print(10, 20, 30, end="!")
Output:
10 20 30!
```

Remember: sep → between objects; end → after the output.

14. The file Argument – Important Concept

The file argument specifies where the output should be written. Its default value in the print() function syntax is None.

```
print(*objects, sep=' ', end='\n', file=None, flush=False)
```

When file is left at its default value of None, print() sends the output to the standard output stream (sys.stdout) in normal Python execution. For beginner-level use, it is sufficient to remember that print() normally displays output on the screen.

15. Why is print() Important?

The print() function is useful for displaying information, variable values, calculation results, and for debugging programs.

```
print("Welcome to Python")
age = 20
print(age)
print(10 + 20)
Output:
30

x = 10
y = 20
print("x =", x)
print("y =", y)
Output:
x = 10
y = 20
```

16. Complete Example

```
print("Hello, World!")
print("Beautiful, World!")
print("Cruel, World!")

print(0, 100, 200, 300, 400)
print(0, 100, 200.5, 300, 400.9)

print(0, 100, 200, 300, 400, sep='-')
print(0, 100, 200, 300, 400, sep='$')
print(0, 100, 200, 300, 400, sep='^')

print("Welcome to EngineersTutor.com")
print("Welcome to EngineersTutor.com.")
print("Welcome to EngineersTutor.com...")

a = 5
print("Value of a is:", a)
```

17. Quick Reference

Purpose	Example
Print text	print("Hello, World!")
Print numbers	print(10, 20, 30)
Use a separator	print(10, 20, 30, sep="-")
Print a variable	a = 5\nprint(a)
Print text and a variable	print("Value of a is:", a)
Change the ending	print("Hello", end=" ")\nprint("World")

18. Summary

The print() function is used to display output in Python. The standard syntax is:

```
print(*objects, sep=' ', end='\n', file=None, flush=False)
```

The most important concepts for beginners are:

```
*objects → values to print
sep → separator between multiple objects
end → what is printed after the output
file → output stream; default is None
flush → controls flushing of the output stream
```

In short, print() is a fundamental Python function. Understanding objects, sep, end, and the basic meaning of file and flush provides a strong foundation for learning Python input/output.