Double-click (or enter) to edit

🖨️ Topic: `print()` Function in Python

The `print()` function is used to display output to the screen/console.

```
print("Welcome to PyShaala!")
```

```
Welcome to PyShaala!
```

```
print('Hello Wolrd!')
```

```
Hello Wolrd!
```

🔷 Syntax:

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

🔷 Default Arguments:

- `sep=' '` : Separator between objects (default is space)
- `end='\n'` : What to print at end (default is newline)
- `file=sys.stdout` : Where to print (default is screen)
- `flush=False` : Buffer flush

🔷 Examples:

1. *objects – One or more items to print

- This can be any number of values separated by commas.

🔶 Example:

```
print("Hello", "World", 123, 10.5)
```

```
Hello World 123 10.5
```

2. sep=' ' – Separator between the objects

- By default, a space (' ') separates multiple objects.

🔶 Example:

```
print("Python", "is", "fun",sep='*')
```

```
Python*is*fun
```

```
print("2025", "07", "29", sep='-')
```

```
2025-07-29
```

3. end='\n' – String appended after the last object

- By default, a newline ('\n') is added at the end of each print().

🔶 Example:

```
print("Hello", end=' ')
print("World")
```

```
Hello World
```

```
print("Hello", end=' ')
print("World")
```

```
Hello World
```

```
print("Line1", end='***')
print("Line2")
```

```
Line1***Line2
```

```
print("please do like", "share" ,"subscribe", end='!\n')
print("please do like", "share" ,"subscribe", sep=', ', end='!\n')
```

```
please do like share subscribe!
please do like, share, subscribe!
```

4. file=sys.stdout – Where to print (default is screen) You can redirect the output to a file or another writable stream.

🔶 Example:

```
with open("output.txt", "w") as f:
    print("Hello , please do subscribe our channel", file=f)
```

5. flush=False – Force flushing the output buffer

- This is useful for real-time logging. By default, Python buffers output, and flush=False. If flush=True, it immediately writes the output.

🔶 Example:

```
import time

# Without flush=True (might not show output until the loop ends)
for i in range(5):
    print(f"Loading {i+1}/5", end='\r')  # overwrite on same line
    time.sleep(1)

```

```
import time

for i in range(5):
    print(f"Loading {i+1}/5", end='\r', flush=True)
    time.sleep(1)

```

✅ Here's how you can force the difference to show up:

- Try writing to a file instead of the console.

🔁 Without flush=True:

```
import time

with open("log1.txt", "w") as f:
    for i in range(25):
        f.write(f"Step {i+1}/25\n")
        time.sleep(1)
```

Now open log1.txt while the script is running — it will stay empty or incomplete until the buffer is flushed (when file closes or buffer is full).

✅ With flush=True:

```
import time

with open("log2.txt", "w") as f:
    for i in range(25):
        f.write(f"Step {i+1}/25\n")
        f.flush()  # force flush to file immediately
        time.sleep(1)
```

Now log2.txt will update line by line every second.

✅ When to use flush=True? ✅

- Logging real-time status updates
- Writing output to a file or terminal where buffering causes delay
- Debugging scripts that appear frozen (due to buffered output)
- Showing a spinner/progress bar live

```
import time
import sys

def progress_bar(total=20, sleep_time=0.2):
    for i in range(total + 1):
        percent = int(100 * i / total)
        bar = '=' * i + '-' * (total - i)
        sys.stdout.write(f'\r[{bar}] {percent}%')
        sys.stdout.flush()
        time.sleep(sleep_time)
    print("\n✅ Done!")

progress_bar()

```

```
[====================] 100%
✅ Done!
```

```python
import time
import sys

def spinner(seconds=5):
    spin_chars = ['|', '/', '-', '\\']
    end_time = time.time() + seconds
    idx = 0

    print("Loading ", end='', flush=True)
    while time.time() < end_time:
        sys.stdout.write(spin_chars[idx % len(spin_chars)])
        sys.stdout.flush()
        time.sleep(0.1)
        sys.stdout.write('\b')  # backspace to overwrite the spinner
        idx += 1
    print("✅ Done!")

spinner()

```

```
Loading ✅ Done!
```

🧠 Practice Exercise: Try it Yourself!

1. Print your name using the `print()` function.
2. Print multiple values separated by a comma.
3. Use the `sep` and `end` parameters in a single print statement.
4. Print the following sentence in a single line without using multiple print statements:
   "Learning Python is fun!"