

## print() 함수의 sep 및 end 인자 활용

print() 함수는 출력되는 값들 사이의 구분자(separator)를 지정하는 `sep` 인자와 마지막에 출력될 문자를 제어할 수 있는 `end` 인자를 제공합니다. 이 장에서는 이 두 인자를 사용하여 출력 형식을 자유롭게 조절하는 방법을 깊이 있게 다룹니다. 특히, 줄바꿈 없이 출력하거나 특정 문자열로 출력을 끝내는 고급 기법을 학습합니다.

## print() 함수의 sep 및 end 인자 활용

### 1. 서론: print() 함수의 출력 제어

Python의

```
print()
```

함수는 콘솔에 정보를 출력하는 가장 기본적인 방법입니다. 하지만 단순히 값을 나열하고 줄바꿈하는 것을 넘어,

```
print()
```

함수는

```
sep
```

(separator)와

```
end
```

두 가지 특별한 인자를 통해 출력 형식을 세밀하게 조정할 수 있습니다. 이 장에서는 이 두 인자의 개념과 활용법을 익혀, 여러분이 원하는 방식으로 파이썬 프로그램의 출력을 디자인할 수 있도록 돕습니다.

```
print()
```

함수는 기본적으로 여러 값을 출력할 때 각 값 사이에 공백을 넣고, 모든 출력이 끝난 후에는 자동으로 줄바꿈을 합니다.

`sep`

와

`end`

인자는 이러한 기본 동작을 변경하여, 공백 대신 다른 구분자를 사용하거나 줄바꿈 없이 출력을 이어갈 수 있게 합니다. 이 장의 목표는 이 인자들을 마스터하여 가독성 높고 효율적인 출력을 구현하는 것입니다.

## 2. `print()` 함수의 기본 동작

`sep`

와

`end`

인자를 이해하기 전에

`print()`

함수의 기본 동작을 살펴보겠습니다.

`print()`

함수에 여러 값을 콤마(

`,`

)로 구분하여 전달하면:

```
python
print("Hello", "World")
print(1, 2, 3)
```

출력 결과는 다음과 같습니다:

Hello World

1 2 3

여기서 확인할 수 있는 두 가지 기본 동작은:

1. **기본 구분자 (Separator)**: 여러 인자 사이에는 \*\*공백 문자(

)\*\*가 자동으로 삽입됩니다.

2. **기본 종료 문자 (End)**: 모든 출력이 끝난 후, 마지막에 \*\*줄바꿈 문자(

\n

)\*\*가 자동으로 추가됩니다.

이제 이 기본 동작을 변경할 수 있는

sep

와

end

인자에 대해 자세히 알아보겠습니다.

### 3.

sep

#### 인자: 출력 값 사이의 구분자 제어

sep

인자는

print()

함수가 여러 인자를 출력할 때 그 인자들 사이에 삽입할 문자열을 지정합니다.

```
sep
```

는 'separator'(구분자)의 약자이며, 기본값은 공백입니다.

### 3.1.

```
sep
```

## 인자의 사용법과 다양한 구분자

```
sep
```

인자는

```
print()
```

함수 호출 시

```
sep="구분자_문자열"
```

형식으로 전달됩니다.

python

## 기본 동작: 공백으로 구분

```
print("사과", "바나나", "오렌지") # 출력: 사과 바나나 오렌지
```

## 쉼표와 공백으로 구분

```
print("사과", "바나나", "체리", sep=", ") # 출력: 사과, 바나나, 체리
```

## 하이픈으로 구분 (날짜 형식)

```
print("2023", "10", "27", sep="-") # 출력: 2023-10-27
```

## 사용자 정의 문자열로 구분

```
print("User", "Login", "Success", sep=" >>> ") # 출력: User >>> Login >>> Success
```

## 빈 문자열로 구분 (붙여쓰기 효과)

```
print("Hello", "World!", sep="") # 출력: HelloWorld!
```

sep

인자는

print()

함수에 2개 이상의 인자가 전달될 때만 의미가 있습니다. 인자가 하나만 있을 때는 구분할 대상이 없으므로

sep

는 영향을 주지 않습니다.

### 3.2.

sep

## 인자의 실용적인 활용

- 로그 메시지 형식화: 시각 정보를 포함한 로그 메시지를 일관된 형식으로 출력할 때.

python

```
import datetime
current_time = datetime.datetime.now().strftime("%H:%M:%S")
print(current_time, "[INFO]", "애플리케이션 시작", sep=" | ")
```

**예시 출력: 14:30:05 | [INFO] | 애플리케이션 시작**

- 데이터 테이블 헤더/행 출력: CSV와 같은 형식의 데이터를 출력할 때 열을 구분하는 데 사용됩니다.

python

```
print("이름", "나이", "도시", sep=",")
print("김철수", "30", "서울", sep=",")
```

**출력:**

**이름,나이,도시**

**김철수,30,서울**

### 4.

```
end
```

## 인자: 출력의 끝을 제어

```
end
```

인자는

```
print()
```

함수가 모든 인자를 출력한 후에 마지막에 어떤 문자열을 추가할지 결정합니다.

```
end
```

의 기본값은 줄바꿈 문자(

```
\n
```

)입니다.

### 4.1.

```
end
```

## 인자의 사용법과 줄바꿈 방지 (

```
end=''
```

### )의 중요성

```
end
```

인자는

```
print()
```

함수 호출 시

```
end="종료_문자열"
```

형식으로 전달됩니다.

**가장 중요한 활용법은 줄바꿈을 방지하는 것입니다.**

```
end=''
```

(빈 문자열)을 지정하면

```
print()
```

함수는 출력 후 어떤 문자도 추가하지 않으므로, 다음

```
print()
```

호출의 출력이 현재 줄에 이어서 나타나게 됩니다.

python

```
print("줄바꿈 없이", end="")  
print("연결됩니다.")
```

## 출력: 줄바꿈 없이 연결됩니다.

**참고 자료와의 연계:** 이 기능은 파이썬 프로그래밍에서 매우 중요하게 다루어집니다.

- **참고 자료 1** ([automatetheboringstuffwithpython2ndedition.pdf](#))에서는 "We don't want to automatically print a newline after these spaces, so we also pass

```
end=''
```

to the first

```
print()
```

call." 라고 언급하며, 자동 줄바꿈 방지의 필요성을 강조합니다.

- 참고 자료 2 ([inventyourowncomputergameswithpython4thedition.pdf](#)) 역시

```
end=''
```

를 사용함으로써

```
print()
```

함수가 줄바꿈 대신 빈 문자열을 추가하여 다음 출력이 현재 줄에 이어지게 하는 원리를 설명합니다. 이는 특히 사용자에게 프롬프트를 보여주거나, 진행 상황을 한 줄에 업데이트할 때 필수적인 기법입니다.

## 4.2. 사용자 정의 종료 문자열

```
end
```

인자에는 빈 문자열 외에 특정 문자열을 지정하여 출력을 끝낼 수도 있습니다.

python

```
print("로딩 중", end="...")
```

```
print("완료!")
```

**출력: 로딩 중...완료!**

```
print("이름을 입력하세요", end=": ")
```

```
name = input() # 사용자 입력 대기
```

```
print("안녕하세요,", name, "님!")
```

**출력:**

**이름을 입력하세요: (사용자 입력 후)**

**안녕하세요, [입력된 이름] 님!**

참고 자료 3과의 연계:

```
bigbookofsmallpythonprojects.pdf
```

에서도

```
print(' ', ' ', end='')
```

와 같이

```
end
```

인자를 활용하여 출력의 마지막에 특정 문자열을 추가하는 예시를 보여줍니다. 이는 여러

```
print()
```

호출의 결과를 시각적으로 연결하거나 복잡한 출력 형식을 구축할 때 유용합니다.

### 4.3.

```
end
```

## 인자의 실용적인 활용

- **진행 상황 표시:** 긴 작업의 진행률을 한 줄에 업데이트하여 보여줄 때

```
\r
```

(캐리지 리턴)과 함께 사용됩니다.

```
python
import time
for i in range(5):
    print(f"처리 중 {i+1}/5...", end="\r", flush=True)
    time.sleep(0.5)
print("\n처리 완료!")
```

**출력: (0.5초마다 갱신되는) 처리 중 1/5... -> ... -> 처리 중 5/5...**

**처리 완료!**

```
(
```

```
flush=True
```

는 버퍼링으로 인한 지연 없이 즉시 출력을 갱신합니다.)

- **한 줄에 여러 정보 나열:** 반복문 내에서 데이터를 한 줄에 나열하고 싶을 때.

```
python
for i in range(1, 6):
```

```
print(i, end=" ")
print("숫자 출력 완료.")
```

## 출력:

1 2 3 4 5

숫자 출력 완료.

## 5.

```
sep
```

와

```
end
```

## 인자 함께 활용하기

```
sep
```

와

```
end
```

인자는 서로 독립적이지만, 함께 사용하면 매우 유연하고 강력한 출력 형식을 만들 수 있습니다.

### 예시 1: 쉼표로 구분하고 특정 기호로 끝맺음

여러 값을 쉼표로 구분한 후, 줄바꿈 대신 특정 기호를 추가하여 다음 출력을 같은 줄에 이어서 보여줍니다.

python

```
print("사과", "바나나", "오렌지", sep=", ", end=" => 다음 과일은? ")
print("포도", "수박", sep=", ", end=".\\n")
```

**출력: 사과, 바나나, 오렌지 => 다음 과일은? 포도, 수박.**

### 예시 2: 테이블 형식의 데이터 출력

```
sep
```

를 탭(

```
\t
```

)으로 설정하여 열을 정렬하고,

```
end
```

를 줄바꿈(

```
\n
```

)으로 설정하여 각 행을 구분합니다. 리스트의 요소를

```
*
```

연산자로

```
print()
```

함수에 전달하는 기법을 함께 활용합니다.

python

```
headers = ["이름", "나이", "직업"]  
data1 = ["김철수", "30", "개발자"]
```

```
print(*headers, sep="\t")  
print(*data1, sep="\t")
```

**출력:**

**이름 나이 직업**

**김철수 30 개발자**

## 6. 결론 및 실습 과제

```
print()
```

함수의

`sep`

와

`end`

인자는 파이썬 프로그램의 출력을 정교하게 제어할 수 있는 핵심적인 도구입니다.

- `sep`

**인자:** 여러 값을 출력할 때 값들 사이의 구분자를 정의하며, 기본값은 공백입니다.

- `end`

**인자:**

`print()`

함수 호출의 마지막에 추가될 문자열을 정의하며, 기본값은 줄바꿈 문자(

`\n`

)입니다. 특히

`end=''`

는 줄바꿈 없이 출력을 이어갈 때 매우 유용합니다.

이 두 인자를 자유자재로 활용함으로써 여러분의 파이썬 프로그램은 훨씬 더 유용하고 사용자 친화적인 출력을 생성할 수 있을 것입니다. 다음 실습 과제를 통해 직접 코드를 작성하며 학습한 내용을 다져보세요.

## 6.1. 실습 과제

1. **개인 정보 출력:** 자신의 이름, 나이, 좋아하는 취미를

`print()`

함수로 한 줄에 출력하되, 각 정보 사이에

|

(공백-수직선-공백) 문자를 구분자로 사용하고, 출력의 마지막에는

-- END --

라는 문자로 끝나도록 해보세요.

- 예시:

홍길동 | 25 | 독서 -- END --

## 2. 연속 숫자 출력:

for

루프를 사용하여 1부터 7까지의 숫자를 한 줄에

\*

로 구분하여 출력하고, 마지막 숫자 뒤에는

!

를 붙여 줄바꿈하도록 해보세요.

- 예시:

1\*2\*3\*4\*5\*6\*7!

## 3. 별표 삼각형 만들기:

print()

함수와

`end`

인자를 여러 번 사용하여 다음과 같은 별표 삼각형 패턴을 만들어 보세요.

•

\*\*

(힌트: 각

`print()`

호출이 줄바꿈을 하지 않도록

`end=''`

를 사용하고, 마지막에만 줄바꿈을 추가하는 방식을 고려해 보세요.)

이 과제들을 통해

`sep`

와

`end`

인자에 대한 이해를 심화하고, 실제 프로그래밍 문제 해결에 적용하는 능력을 키울 수 있을 것입니다. 성공적인 학습을 기원합니다!

이전 챕터

다음 챕터