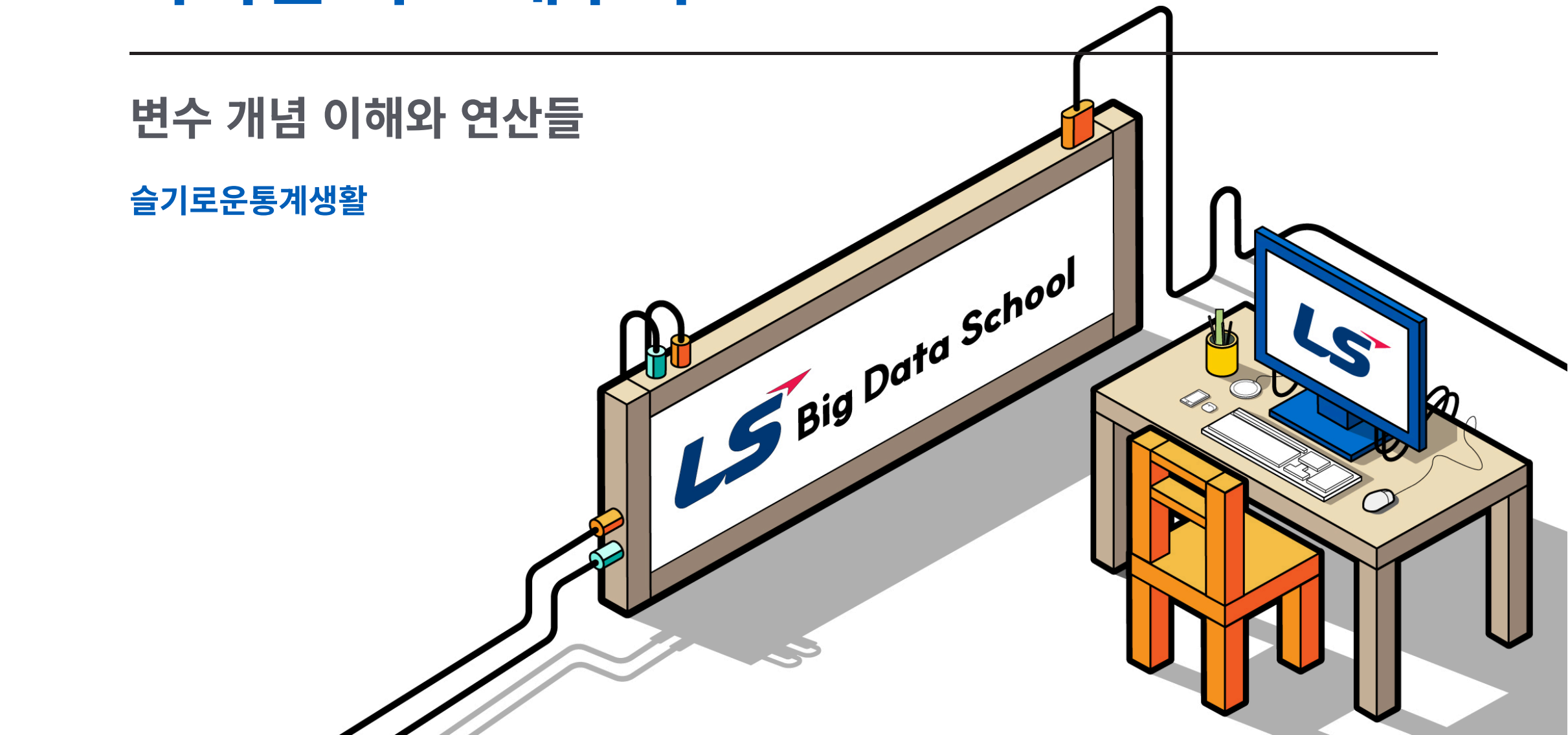
파이썬 기초 배우기

변수 개념 이해와 연산들

슬기로운통계생활



코스 훑어보기.

파이썬에서 변수의 개념에 대해 학습합니다.

변수 개념 이해하기

파이썬에서 변수는 데이터를 저장하는 컨테이너입니다. 변수를 사용하면 코드 내에서 데이터를 더 쉽게 관리할 수 있습니다.

변수 선언과 할당

파이썬에서 변수를 선언하고 값을 할당하는 것은 매우 간단합니다. 변수 이름을 정하고 값을 할당하려면 `=` 연산자를 사용합니다.

```
# 변수에 값을 할당
number = 10
greeting = "안녕하세요!"
```

변수 사용하기

변수에 값을 할당한 후에는, 변수 이름을 통해 해당 값을 언제든지 참조할 수 있습니다.

```
# 변수 값을 출력
print(number)
```

```
## 10
```

```
print(greeting)
```

```
## 안녕하세요!
```

이 예제에서 number 변수는 정수 10을, `greeting` 변수는 문자열 "안녕하세요!"를 각각 저장하고 있습니다. `print()` 함수를 사용하여 이들 변수의 내용을 콘솔에 출력합니다.

변수의 동적 타이핑

파이썬은 동적 타입 언어입니다. 이는 변수의 데이터 타입을 미리 선언할 필요가 없다는 의미입니다. 변수의 타입은 할당된 값에 따라 자동으로 결정됩니다.

```
# 변수 타입 변경
number = "이제 문자열입니다"
print(number)
```

```
## 이제 문자열입니다
```

Python에서 변수명 짓기

파이썬에서 변수를 선언할 때는 몇 가지 중요한 규칙과 관례를 따라야 합니다. 이러한 규칙들은 코드의 가독성과 유지 보수성을 높이는 데 기여합니다.

변수명 규칙

파이썬에서 변수명을 지을 때 따라야 하는 기본적인 규칙들입니다.

1. 변수명은 알파벳(대소문자 구분), 숫자, 밑줄(_)로 구성할 수 있으나, 숫자로 시작할 수는 없습니다.
2. 파이썬은 대소문자를 구분합니다. 예를 들어, `data`와 `Data`는 서로 다른 변수입니다.
3. 파이썬의 예약어는 변수명으로 사용할 수 없습니다. 예: `if`, `for`, `class`, 등.
4. 변수명은 그 용도나 저장된 데이터를 명확히 설명할 수 있도록 의미 있는 이름을 사용하는 것이 좋습니다.

좋은 변수명 예제

적절한 변수명은 코드의 이해를 돕고, 나중에 코드를 유지보수할 때 큰 도움이 됩니다.

- `user_age` : 사용자의 나이를 나타냅니다.
- `total_price` : 총 가격을 나타냅니다.
- `num_items` : 항목 수를 나타냅니다.
- `is_active` : 활성 상태 여부를 나타냅니다.

변수명 스타일

파이썬 커뮤니티에서 권장하는 몇 가지 변수명 스타일입니다.

- **스네이크 케이스(snake_case)** : 모든 글자를 소문자로 하고 단어 사이에 밑줄을 사용합니다. 함수와 변수명에 주로 사용됩니다.
- **카멜 케이스(CamelCase)** : 각 단어의 첫 글자를 대문자로 시작합니다. 클래스 이름에 주로 사용됩니다.

이러한 규칙을 따르는 것은 코드를 보다 효율적으로 작성하고, 다른 개발자들과의 협업 시에도 이해하기 쉬운 코드를 유지하는 데 도움이 됩니다.

기본 계산관련 연산자 학습하기

파이썬에서는 다양한 종류의 연산자를 제공하여 데이터 처리와 계산을 용이하게 합니다. 기본적인 산술 연산자부터 시작해 보겠습니다.

변수 할당 및 기본 산술 연산자

먼저, 두 변수 `a`와 `b`에 각각 숫자를 할당하고, 이를 사용하여 기본적인 산술 연산을 수행해 보겠습니다.

```
# 변수 할당
a = 10
b = 3.3

# 기본 산술 연산자
print("a + b =", a + b)  # 덧셈
```

```
## a + b = 13.3
```

```
print("a - b =", a - b)  # 뺄셈
```

```
## a - b = 6.7
```

변수 할당 및 기본 산술 연산자

```
print("a * b =", a * b)  # 곱셈
```

```
## a * b = 33.0
```

```
print("a / b =", a / b)  # 나눗셈
```

```
## a / b = 3.0303030303030303
```

```
print("a % b =", a % b)  # 나머지
```

```
## a % b = 0.10000000000000053
```

변수 할당 및 기본 산술 연산자

```
print("a // b =", a // b)  # 몫
```

```
## a // b = 3.0
```

```
print("a ** b =", a ** b)  # 거듭제곱
```

```
## a ** b = 1995.2623149688789
```

비교관련 연산자 학습하기

비교 연산자는 두 값의 관계를 평가하는 데 사용됩니다. 이 결과는 조건문과 반복문에서 중요한 역할을 합니다. 파이썬에서 사용되는 비교 연산자들과 그 의미를 아래 표에서 확인할 수 있습니다.

연산자	설명
`==`	동등. 두 값이 같음
`!=`	부등. 두 값이 다름
`<`	미만. 왼쪽 값이 오른쪽 값보다 작음
`>`	초과. 왼쪽 값이 오른쪽 값보다 큼
`<=`	이하. 왼쪽 값이 오른쪽 값보다 작거나 같음
`>=`	이상. 왼쪽 값이 오른쪽 값보다 크거나 같음

이 연산자들은 수학적 비교를 기반으로 하며, 논리적 조건 평가에 사용됩니다. 이어지는 섹션에서는 이 연산자들을 실제 코드 예제와 함께 더 자세히 살펴보겠습니다.

비교 연산자의 종류

파이썬에서 사용되는 주요 비교 연산자는 다음과 같습니다:

```
# 동등 비교
a = 10
b = 20
print("a == b:", a == b)  # a와 b가 같은지 비교
```

```
## a == b: False
```

```
# 부등 비교
print("a != b:", a != b)  # a와 b가 다른지 비교
```

```
## a != b: True
```

비교 연산자의 종류

파이썬에서 사용되는 주요 비교 연산자는 다음과 같습니다:

```
# 크기 비교
print("a < b:", a < b)   # a가 b보다 작은지 비교
```

```
## a < b: True
```

```
print("a > b:", a > b)   # a가 b보다 큰지 비교
```

```
## a > b: False
```

비교 연산자의 종류

파이썬에서 사용되는 주요 비교 연산자는 다음과 같습니다:

```
# 크거나 같음, 작거나 같음
print("a <= b:", a <= b)  # a가 b보다 작거나 같은지 비교
```

```
## a <= b: True
```

```
print("a >= b:", a >= b)  # a가 b보다 크거나 같은지 비교
```

```
## a >= b: False
```

비교 연산자의 사용 예

비교 연산자는 변수 또는 표현식의 값을 평가하여 조건문에서 매우 유용하게 사용됩니다. 예를 들어, 사용자 입력값의 검증, 데이터의 필터링, 특정 조건에 따른 처리 등에 사용됩니다.

```
# 사용자 나이 검증 예제
user_age = 25
is_adult = user_age >= 18
print("성인입니까?", is_adult)
```

```
## 성인입니까? True
```

논리 관련 연산자 학습하기

논리 연산자는 주로 불리언(참 또는 거짓) 값을 조작하는 데 사용됩니다. 이 연산자들은 복잡한 조건문을 구성할 때 매우 유용하게 사용됩니다.

- '불리언(Boolean)'은 수학자 조지 불(George Boole)의 이름에서 유래되었으며, 컴퓨터 과학에서는 참(True) 또는 거짓(False)의 두 가지 값만을 가질 수 있는 데이터 타입을 지칭합니다.

논리 연산자의 종류

아래 표는 파이썬에서 사용되는 주요 논리 연산자들과 그 의미를 설명합니다.

연산자	설명
`and`	두 조건이 모두 참일 때 참
`or`	두 조건 중 하나라도 참일 때 참
`not`	조건의 불리언 값을 반전 (참이면 거짓, 거짓이면 참)

논리 연산자 사용 예제

이제 이 연산자들을 어떻게 실제 코드에서 사용하는지 살펴보겠습니다.

```
# 논리 연산자 예제
a = True
b = False

# and 연산자
print("a and b:", a and b)  # False, a와 b 둘 다 참이어야 참 반환
```

```
## a and b: False
```

논리 연산자 사용 예제

```
# or 연산자
print("a or b:", a or b)     # True, a와 b 중 하나라도 참이면 참 반환
```

```
## a or b: True
```

```
# not 연산자
print("not a:", not a)       # False, a의 반대 불리언 값 반환
```

```
## not a: False
```

복합 대입 연산자 학습하기

복합 대입 연산자는 값을 연산하고 그 결과를 같은 변수에 할당하는 축약된 방법을 제공합니다. 이 연산자들은 코드를 더 간결하게 만들고, 자주 사용되는 연산을 더 효율적으로 처리할 수 있게 돕습니다.

복합 대입 연산자의 종류

아래 표는 파이썬에서 사용되는 주요 복합 대입 연산자들과 그 의미, 사용 예를 설명합니다.

- `연사자=` 형식임을 기억하면 좋습니다.

연산자	설명	사용 예	풀어서 쓰기
`+=`	왼쪽 변수에 오른쪽 값을 더하고 결과를 할당	`a += 10`	`a = a + 10`
`-=`	왼쪽 변수에서 오른쪽 값을 빼고 결과를 할당	`a -= 10`	`a = a - 10`
`*=`	왼쪽 변수에 오른쪽 값을 곱하고 결과를 할당	`a *= 10`	`a = a * 10`
`/=`	왼쪽 변수를 오른쪽 값으로 나누고 결과를 할당	`a /= 10`	`a = a / 10`
`%=`	왼쪽 변수를 오른쪽 값으로 나눈 나머지를 할당	`a %= 10`	`a = a % 10`
`**=`	왼쪽 변수를 오른쪽 값의 거듭제곱 후 결과를 할당	`a **= 10`	`a = a ** 10`
`//=`	왼쪽 변수를 오른쪽 값으로 나눈 몫을 할당	`a //= 10`	`a = a // 10`

복합 대입 연산자 사용 예제

이제 이 연산자들을 어떻게 실제 코드에서 사용하는지 살펴보겠습니다.

```
# 복합 대입 연산자 예제
a = 100

a += 10
print("a += 10:", a)  # a = a + 10
```

```
## a += 10: 110
```

```
a -= 20
print("a -= 20:", a)  # a = a - 20
```

```
## a -= 20: 90
```

복합 대입 연산자 사용 예제

```
a *= 2
print("a *= 2:", a)  # a = a * 2
```

```
## a *= 2: 180
```

```
a /= 2
print("a /= 2:", a)  # a = a / 2
```

```
## a /= 2: 90.0
```

```
a %= 14
print("a %= 14:", a)  # a = a % 3
```

```
## a %= 14: 6.0
```

복합 대입 연산자 사용 예제

```
a **= 2
print("a **= 2:", a)  # a = a ** 2
```

```
## a **= 2: 36.0
```

```
a //= 2
print("a //= 2:", a)  # a = a // 2
```

```
## a //= 2: 18.0
```

연산자 적용 우선순위

파이썬에서 연산자 우선순위는 표현식 내에서 연산자가 평가되는 순서를 결정합니다. 이 우선순위를 이해하는 것은 복잡한 표현식을 올바르게 계산하고 프로그래밍 오류를 방지하는 데 중요합니다.

연산자 우선순위 표

아래 표는 파이썬에서 사용되는 주요 연산자들의 우선순위를 높은 것부터 낮은 것까지 나열하며, 각 연산자의 의미를 추가적인 칼럼으로 설명합니다.

우선순위	연산자	설명
1	`()`, `[]`, `{}`	괄호: 그룹화 및 우선순위 지정
2	`**`	지수 연산: 거듭제곱 계산
3	`+x`, `-x`, `~x`	단항 연산자: 각각 양수, 음수, 비트 NOT 연산
4	`*`, `/`, `//`, `%`	곱셈, 나눗셈, 정수 나눗셈, 나머지 연산
5	`+`, `-`	이항 연산자: 덧셈과 뺄셈
6	`<<`, `>>`	비트 시프트: 비트를 왼쪽 혹은 오른쪽으로 이동

연산자 우선순위 표

Table:

우선순위	연산자	설명
7	`&`	비트 AND: 비트 단위 AND 연산
8	`^`	비트 XOR: 비트 단위 XOR 연산
9	`|`	비트 OR: 비트 단위 OR 연산
10	`==, !=, <, <=, >, >=`	비교 연산자: 등식 및 부등식 평가
11	`not`	논리 NOT: 불리언 반전
12	`and`	논리 AND: 불리언 AND 연산
13	`or`	논리 OR: 불리언 OR 연산

연산자 우선순위 예제

이제 연산자 우선순위가 어떻게 적용되는지 실제 예제를 통해 살펴보겠습니다.

```
# 연산자 우선순위 예제
result = 10 + 2 * 3 ** 2 / 6 - (4 + 2) ** 2
print("Result of the expression:", result)
```

```
## Result of the expression: -23.0
```

문자열에 대한 `+` 연산자 적용

파이썬에서 문자열은 매우 유연하게 다룰 수 있는 데이터 타입 중 하나입니다. 여기서는 문자열 변수를 만들고, 이를 사용하는 `+` 연산자에 대해 알아보겠습니다.

- 문자열을 다룰 때 `+` 연산자는 두 문자열을 연결하는 데 사용됩니다. 이를 문자열 연결(concatenation)이라고 합니다.

```
# 문자열 변수 할당
str1 = "Hello, "
str2 = "world!"

# 문자열 연결
result = str1 + str2
print("Concatenated string:", result)
```

```
## Concatenated string: Hello, world!
```

문자열과 숫자의 덧셈

파이썬에서는 문자열과 숫자를 직접 더하려고 하면 에러가 발생합니다. 이는 데이터 타입이 서로 다르기 때문에 발생하는 현상입니다.

```
# 문자열과 숫자의 덧셈 시도
number = 123
new_result = str1 + number
```

```
## TypeError: can only concatenate str (not "int") to str
```

위의 예제에서 `str1`은 문자열이고, `number`는 정수입니다. 이 두 타입을 `+` 연산자로 더하려고 할 때 파이썬은 `TypeError`를 발생시키고, 연산을 수행할 수 없음을 알려줍니다.

문자열에 대한 `*` 연산자 적용

파이썬에서 `*` 연산자는 문자열을 반복하는 데 사용됩니다. 이 연산자를 사용하면 지정된 횟수만큼 문자열을 반복하여 새로운 문자열을 생성할 수 있습니다.

```
# 문자열 변수 할당
str1 = "Hello! "

# 문자열 반복
repeated_str = str1 * 3
print("Repeated string:", repeated_str)
```

```
## Repeated string: Hello! Hello! Hello!
```

문자열과 숫자의 곱셈

문자열과 숫자의 곱셈은 오직 **정수와의 곱셈**만 가능합니다. 실수나 다른 타입의 데이터와의 곱셈은 타입 에러를 발생시키게 됩니다.

```
float_str = str1 * 2.5
```

```
## TypeError: can't multiply sequence by non-int of type 'float'
```

```
print("Error:", e)
```

```
## NameError: name 'e' is not defined
```

문제 1: 사용자 정보 출력

1. 변수 name에 사용자 이름 "홍길동"을 저장합니다.
2. 변수 age에 사용자 나이 23을 저장합니다.
3. name과 age를 사용하여 다음 문장을 출력하세요:

안녕하세요! 제 이름은 홍길동이고, 나이는 23살입니다.

문제 1: 사용자 정보 출력 해답

```
# 변수 선언
name = "홍길동"
age = 23

# 출력
print("안녕하세요! 제 이름은", name, "이고, 나이는", age, "살입니다.")
```

```
## 안녕하세요! 제 이름은 홍길동 이고, 나이는 23 살입니다.
```

문제 2: 문자열 반복하기

1. 변수 pattern에 문자열 "*-"를 저장합니다.
2. pattern 문자열을 15번 반복하여 다음과 같은 문자열을 출력하세요.

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
```

문제 2: 문자열 반복하기 해답

```
# 변수 선언
pattern = "*-"

# 문자열 반복
result = pattern * 15
print(result)
```

```
## *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
```

문제 3: 복합 대입 연산자 활용

1. 변수 result에 100을 저장합니다.
2. 아래 연산을 순서대로 적용하세요.
    - result에 25를 더합니다.
    - result를 7로 나눈 나머지를 저장합니다.
    - result의 값을 2로 곱합니다.
3. 최종 result 값을 출력하세요.

문제 3: 복합 대입 연산자 활용 해답

```
# 변수 선언
result = 100

# 복합 대입 연산 적용
result += 25        # 100 + 25
result %= 7         # 125 % 7
result *= 2         # 6 * 2

# 결과 출력
print("최종 result 값:", result)
```

```
## 최종 result 값: 12
```

문제 4: 숫자의 크기 비교

1. 변수 x에 25, 변수 y에 30을 저장합니다.
2. 두 변수의 관계를 비교하는 연산을 사용하여 다음 문장을 각각 출력하세요.
    - x가 y보다 작다.
    - x가 y보다 크다.
    - x와 y가 같다.

출력 결과 예시:

x가 y보다 작다: True x가 y보다 크다: False x와 y가 같다: False

문제 4: 숫자의 크기 비교 해답

```
# 변수 선언
x = 25
y = 30

# 비교 결과 출력
print("x가 y보다 작다:", x < y)
```

```
## x가 y보다 작다: True
```

```
print("x가 y보다 크다:", x > y)
```

```
## x가 y보다 크다: False
```

```
print("x와 y가 같다:", x == y)
```

문제 5: 연산자 우선순위 이해

아래 표현식을 계산하고 결과를 출력하세요.

expression = 8 *2 + 5* *2 // 4 - 7

문제 5: 연산자 우선순위 이해 해답

```
# 표현식 계산
expression = 8 * 2 + 5 ** 2 // 4 - 7

# 결과 출력
print("Expression 결과:", expression)
```

```
## Expression 결과: 15
```