20250410 파이썬 자료형과 연산자4

5절. 연산자

5.1 산술연산자

연산자	설명	예시
`+`	덧셈	`3 + 2 = 5`
`-`	뺄셈	`5 - 2 = 3`
`*`	곱셈	`3 * 2 = 6`
`/`	나눗셈	`5 / 2 = 2.5`
`//`	몫 연산자	`5 // 2 = 2`
`%`	나머지 연산자	`5 % 2 = 1`
`**`	제곱	`3 ** 2 = 9`

5.2 할당연산자

연산자	설명	동일 표현
`=`	기본 할당	`x = 5`
`+=`	더하기 후 할당	`x += 3` ≡ `x = x + 3`
`-=`	빼기 후 할당	`x -= 3` ≡ `x = x - 3`
`*=`	곱하기 후 할당	`x *= 3` ≡ `x = x * 3`
`/=`	나누기 후 할당	`x /= 3` ≡ `x = x / 3`
`//=`	몫 계산 후 할당	`x //= 3` ≡ `x = x // 3`
`%=`	나머지 계산 후 할당	`x %= 3` ≡ `x = x % 3`

5.3 논리연산자

기본 논리연산자

연산자	설명
`and`	논리연산만 가능. False로 판별되는 첫번째 항의 결과 반환. 모든 항이 참이면 마지막 항을 반환
`or`	논리연산만 가능. 참으로 판별되는 첫번째 항의 결과 반환. 모든 항이 거짓이면 마지막 항을 반환
`not`	조건의 결과를 반대로 변환

비트 논리연산자

연산자	설명
&	논리연산자와 비트연산자로 모두 사용 가능
|	논리연산자와 비트연산자로 모두 사용 가능
~	비트 NOT 연산

예시 코드

python

```python
# and 논리연산자
print((10 > 3) and (10 > 5))  # True
print(0 and 5)  # 0 (첫 번째 항이 False로 평가되어 0 반환)
print(1 and 5)  # 5 (모든 항이 True로 평가되어 마지막 항 반환)

# or 논리연산자
print(1 or 0)  # 1 (첫 번째 항이 True로 평가되어 1 반환)
print(0 or '')  # '' (모든 항이 False로 평가되어 마지막 항 반환)

# & 논리연산자로도 사용 가능
print((10 > 3) & (10 > 5))    # True

# & 비트연산자로 사용
print(12 & 1)  # 0
# 12: 0000 1100
#  1: 0000 0001
# 결과: 0000 0000

# | 비트연산자 예시
print(12 | 1)  # 13
# 12: 0000 1100
#  1: 0000 0001
# 결과: 0000 1101
```

6절. 문자열 다루기

문자열 조작 메서드

메서드/기능	설명	예시
슬라이싱	`변수[from:stop:step]` 형식으로 문자열 일부 추출	`"Python"[1:4]` → `"yth"`
`len(문자열)`	문자 개수 반환	`len("Python")` → `6`
`upper()`	대문자로 변환	`"python".upper()` → `"PYTHON"`
`lower()`	소문자로 변환	`"PYTHON".lower()` → `"python"`
`title()`	각 어절의 첫글자만 대문자로 변환	`"hello world".title()` → `"Hello World"`
`capitalize()`	첫 문자만 대문자로 변환	`"hello world".capitalize()` → `"Hello world"`

문자열 검색 메서드

메서드	설명	예시
`count('찾을문자열')`	찾을 문자열의 등장 횟수 반환	`"hello".count("l")` → `2`
`find('찾을문자열')`	왼쪽부터 찾아 첫 번째 인덱스 반환 **없으면 -1 반환**	`"hello".find("l")` → `2`
`rfind('찾을문자열')`	오른쪽부터 찾아 첫 번째 인덱스 반환 **없으면 -1 반환**	`"hello".rfind("l")` → `3`
`index('찾을문자열')`	왼쪽부터 찾아 첫 번째 인덱스 반환 **없으면 오류 발생**	`"hello".index("l")` → `2`

문자열 검증 메서드

메서드	설명	예시
`startswith('문자열')`	특정 문자열로 시작하는지 확인	`"hello".startswith("he")` → `True`
`endswith('문자열')`	특정 문자열로 끝나는지 확인	`"hello".endswith("lo")` → `True`
`isdigit()`	숫자 문자열인지 확인	`"123".isdigit()` → `True`
`islower()`	소문자 문자열인지 확인	`"hello".islower()` → `True`
`isupper()`	대문자 문자열인지 확인	`"HELLO".isupper()` → `True`

문자열 변환 및 공백 처리 메서드

메서드	설명	예시
`replace(old, new)`	old 문자열을 new로 변경	`"hello".replace("l", "r")` → `"herro"`
`split()`	문자열을 분리하여 리스트로 반환 **기본은 공백 단위로 분리**	`"hello world".split()` → `["hello", "world"]` `"a,b,c".split(",")` → `["a", "b", "c"]`
`strip()`	좌우 공백 제거	`"  hello  ".strip()` → `"hello"`
`rstrip()`	오른쪽 공백 제거	`"  hello  ".rstrip()` → `"  hello"`
`lstrip()`	왼쪽 공백 제거	`"  hello  ".lstrip()` → `"hello  "`