2편 생선을 잡아라

1. 캣의 다짐
2. 캣의 생선팔기
3. 대입 연산자(할당 연산자)
4. 비교 연산자
5. 논리 연산자
6. 비트 연산자
7. 멤버 연산자
8. 식별 연산자
9. 연산자의 우선순위

1. 캣의 다짐

어느날 생선가게를 운영하며 열심히 살아가던 캣에게 슬픔이 찾아왔습니다. 아프던 캣의 어머니의 병세가 더 위독해진 것이었어요.

당장 병원에 가야했지만 '위니브 월드'에서의 병원은 왕족인 사자와 부자에게만 허락된 공간이었습니다. 평민이었던 캣의 어머니는 병원에 가지 못하였고 캣에게 생선가게를 부탁하였습니다.

"생선가게를 부탁하마..."

"엄마.."

캣은 다짐했습니다.

> 평민도 이용할 수 있는 병원을 세울꺼다냥!!

캣은 병원을 세우기 위해 자신의 스킬을 활용하여 더 열심히 생선을 잡고 팔았습니다.

- **라이캣의 현재 상태창!**

```python
#[In]

이름 = '캣'
설명 = '위니브 월드 외각에 살고 있는 생선가게 주인 캣(cat)'
나이 = 13
오늘_잡은_물고기 = '10'
키 = '45cm'
몸무게 = 1.2
육식 = True
초식 = True
돈 = 5000
훈장 = ['백상아리를 잡은 고양이']
기술 = ['고기잡이', '고기팔기']
```

Python

캣은 장사를 할 때 캣만의 규칙이 있었습니다. 잡은 물고기를 당일 다 팔아야 하고 다음날 매출은 오늘 매출의 2배로 목표를 잡습니다.

```python
#[In]

# 캣의 규칙
오늘_잡은_물고기 = 0
오늘_판_물고기 = 0
매출 = 0

# 물고기를 10마리 잡았습니다.
# 모든 물고기는 100원입니다.

오늘_잡은_물고기 = 오늘_잡은_물고기 + 10
오늘_판_물고기 = 오늘_잡은_물고기
재고 = 오늘_잡은_물고기 - 오늘_판_물고기
매출 = 100*10

#f-string 출력 방법을 사용하셔도 좋습니다.
print('나는 오늘', 오늘_잡은_물고기, '마리를 잡았고', 오늘_판_물고기,'마리를 팔았다냥')
print('재고는', 재고,'마리다냥!')
print('오늘', 매출,'원을 벌었으니 내일은', 매출*2,'원을 벌테다!')
```

Python

`오늘_잡은_물고기 + 10` , `오늘_판_물고기 = 오늘_잡은_물고기` , `오늘_잡은_물고기 - 오늘_판_물고기` , `100*10` 에서 볼 수 있는 +, =, -, * 와 같이 값을 계산하거나 대입하는 것을 연산자라고 합니다.

연산자의 종류로는 **산술 연산자**, **대입 연산자(할당연산자)**, **비교 연산자**, **논리 연산자**, **비트 연산자**, **멤버 연산자**, **식별 연산자**가 있습니다.

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

💡 종류가 많기 때문에 어려움을 느끼실 수 있지만 이 연산자들을 한번에 알려고 하는 것은 큰 의미가 없습니다. 필요할 때 사용해 보면서 익히는 것이 중요합니다.

---

2. 캣의 생선 팔기

캣은 매출을 올리기 위해 모든 물고기를 같은 가격이 아닌 등급에 따라 다르게 책정하기로 합니다.

> 물고기 등급에 따라 가격을 다르게 팔아볼까냥?

```python
#[In]

# A등급 : 1000원, B등급 : 500원, C등급 : 100원
# 오늘 잡은 물고기 : A등급 5마리, B등급 7마리, C등급 10마리

A등급 = 5
B등급 = 7
C등급 = 10
매출 = 0
```

Python

등급에 따라 가격을 책정하여 장사를 시작한 캣의 생선 가게에 손님이 찾아왔습니다.

```python
#[In]

# 손님의 주문
# A등급 2마리, B등급 3마리 주세요.
# 받은 돈은 4000원 입니다.

A등급 = A등급 - 2
B등급 = B등급 - 3

합계 = 1000*2 + 500*3
받은_돈 = 4000

print(f'감사합니다. 여기 거스름 돈 {받은_돈 - 합계}원 입니다.')

재고 = A등급 + B등급 + C등급
매출 = 매출 + 합계

print(f'현재 매출 : {매출}, 재고 : {재고}')
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

여기서 `Alt + Enter` 를 눌러주면 아래와 같이 출력됩니다.

> 💡 물고기의 등급과 가격을 dict로 구현해보시고, 계산해보세요!

```
#[Out]

감사합니다. 여기 거스름돈 500 원입니다.
현재 매출 : 3500 재고 : 17
```

Python

오늘도 잡은 물고기를 다 팔아 재고를 0으로 장사를 마무리 하는 캣. 하지만 위니브 월드에서는 당일 매출의 4분의 1을 세금으로 납부해야 한다고 합니다.

> 에휴 오늘도 세금을 납부해야겠구냥... 오늘은 또 얼마를 내야하냥?

```
#[In]

# 세금 납부하기
총매출 = 10000
세금 = 총매출/4
순이익 = 총매출-세금

print(총매출, 세금, 순이익)
# 10000 2500.0 7500.0
```

Python

+, -, *, / 와 같은 연산자를 **산술연산자**라고 합니다. 산술연산자는 가장 많이 접할 수 있고, 자주 사용되는 연산자입니다. 사칙연산과 더불어 제곱, 나머지 연산자 등이 있습니다.

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

산술 연산자

Aa 기호	이름	설명	예제
+	덧셈	연산자의 왼쪽과 오른쪽 값을 더합니다.	a + b
-	뺄셈	연산자의 왼쪽 값에서 오른쪽 값을 뺍니다.	a - b
*	곱셈	연산자의 왼쪽 값과 오른쪽 값을 곱합니다.	a * b
**	제곱	연산자의 왼쪽 값에서 오른쪽 값만큼 제곱합니다.	a ** b
/	나눗셈	연산자의 왼쪽 값을 오른쪽 값으로 나눕니다.	a / b
//	몫	연산자의 왼쪽 값을 오른쪽 값으로 나눈 몫을 구합니다. (따라서 결과는 정수)	a // b
%	나머지	연산자의 왼쪽 값을 오른쪽 값으로 나눈 나머지를 구합니다.	a % b

COUNT 7

```python
#[In]

a = 5
b = 2

print('a + b = ', a + b)    # 7
print('a - b = ', a - b)    # 3
print('a * b = ', a * b)    # 10
print('a / b = ', a / b)    # 2.5
print('a ** b = ', a ** b)  # 25
print('a // b = ', a // b)  # 2
print('a % b = ', a % b)    # 1
```

Python

💡 다른 프로그래밍 언어에서 연산자를 이미 학습하신 분들은 눈치채셨을 수도 있으시겠지만, 파이썬에는 전/후위 연산자(++, --)가 존재하지 않습니다!

자, 성실히 세금을 납부했으니 훈장을 부여하도록 합시다.

```python
#[In]

훈장.append('성실한 납세자')
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

3. 대입 연산자(할당 연산자)

대입 연산자는 **산술 연산의 코드를 축약**해서 사용할 수 있도록 지원하는 기능입니다.

피연산자를 한번만 사용하기 때문에 대입 연산자를 잘 활용한다면 **코드를 좀더 간결하게 사용할 수 있다**는 장점이 있습니다.

위의 예제에서 사용했던 코드를 가져와서 알아봅시다.

```python
#[In]

# 1번 예제
오늘_잡은_물고기 = 오늘_잡은_물고기 + 10
오늘_판_물고기 = 오늘_잡은_물고기
```

Python

```python
#[In]

# 대입 연산자
오늘_잡은_물고기 += 10
오늘_판_물고기 = 오늘_잡은_물고기
```

Python

`오늘_판_물고기 = 오늘_잡은_물고기` 는 대입 연산자로 오른쪽 값을 왼쪽 변수에 할당하였습니다.
`오늘_잡은_물고기 = 오늘_잡은_물고기 + 10` 는 덧셈 대입을 사용하여 `오늘_잡은_물고기 += 10` 로 축약하여 사용할 수 있고 연산 결과는 동일합니다.

다른 산술 연산자도 모두 대입 연산자로 축약하여 사용할 수 있습니다.

```python
#[In]

a = 10
b = 2

a += b   # 12
a -= b   # 10
a *= b   # 20
a /= b   # 10.0
a **= b  # 100.0
a //= b  # 50.0
a %= b   # 0.0
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

대입 연산자(할당 연산자)

Aa 기호	이름	설명	예제	동일
=	대입	연산자의 오른쪽 값을 왼쪽 변수에 할당합니다.	a = b	a = b
+=	덧셈 대입	연산자의 왼쪽 변수의 값과 오른쪽 값을 더한 결과를 왼쪽 변수에 할당합니다.	a += b	a = a + b
-=	뺄셈 대입	연산자의 왼쪽 변수의 값에서 오른쪽 값을 뺀 결과를 왼쪽 변수에 할당합니다.	a -= b	a = a - b
*=	곱셈 대입	연산자의 왼쪽 변수의 값과 오른쪽 값을 곱한 결과를 왼쪽 변수에 할당합니다.	a *= b	a = a * b
**=	제곱 대입	연산자의 왼쪽 변수의 값에서 오른쪽 값만큼 제곱한 결과를 왼쪽 변수에 할당합니다.	a **= b	a = a ** b
/=	나눗셈 대입	연산자의 오른쪽 변수의 값을 오른쪽 값만큼 나눈 결과를 왼쪽 변수에 할당합니다.	a /= b	a = a / b
//=	몫 대입	연산자의 왼쪽 변수의 값을 오른쪽 값만큼 나눈 몫을 왼쪽 변수에 할당합니다.	a //= b	a = a // b
%=	나머지 연산 대입	연산자의 왼쪽 변수의 값을 오른쪽 값만큼 나눈 나머지를 왼쪽 변수에 할당합니다.	a %= b	a = a % b

COUNT 8

익숙하지 않고, 혼란의 여지가 있을 경우에는 산술 연산자인 형태로 쓰셔도 됩니다. 실질적으로 연산의 결과값은 동일하기 때문에 상관 없습니다.

하지만 간결한 코드를 중요하게 여겨지는 순간이 온다면 그 때 대입 연산자로 활용하는 훈련을 조금씩 하면 됩니다. 물론 함께 일하는 팀의 컨벤션 등이 정해진다면 그것에 따르는 것이 좋습니다.

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

4. 비교 연산자

비교 연산자는 왼쪽 피연산자와 오른쪽 피연산자의 값을 비교하여 boolean 값(True, False)으로 반환해주는 역할을 합니다. 간단히 말하면 **값을 비교하여 참과 거짓을 구분합니다.**

오늘 잡은 물고기와 판매한 물고기의 수를 비교해 봅시다.

```python
#[In]

오늘_잡은_물고기 = 10
오늘_판_물고기 = 5

print(오늘_잡은_물고기 > 오늘_판_물고기)  # True
print(오늘_잡은_물고기 < 오늘_판_물고기)  # False
print(오늘_잡은_물고기 >= 오늘_판_물고기) # True
print(오늘_잡은_물고기 <= 오늘_판_물고기) # False
print(오늘_잡은_물고기 == 오늘_판_물고기) # False
print(오늘_잡은_물고기 != 오늘_판_물고기) # True
```

Python

비교 연산자는 `if` 조건문과 함께 가장 많이 사용됩니다. `if` 문은 뒤의 챕터에서 자세히 다룰 예정이니 이번 챕터에서는 어떻게 실행되는지만 간단히 알아봅시다.

`if`문은 조건문이 참일 경우에만 실행문이 실행되며 거짓일 경우에는 실행되지 않습니다.

```python
if 조건문 :
    실행문
```

Python

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

앞의 예제에서 캣의 생선 가게에는 오늘 잡은 물고기를 당일 다 판매해야 하는 규칙이 있었죠?

```python
오늘_잡은_물고기 = 10
오늘_판_물고기 = 5

if 오늘_잡은_물고기 == 오늘_판_물고기 :
    print('오늘 물고기 판매 끝!')

if 오늘_잡은_물고기 != 오늘_판_물고기 :
    print('아직 물고기를 다 팔지 않았습니다.')
```

Python

이렇게 잡은 물고기의 수와 판매한 물고기의 수를 비교하여 값이 같지 않을 경우와 같을 경우에 따라 다른 문구를 출력할 수 있습니다.

위의 코드는 `if-else` 문을 통해 간결하게 표현할 수도 있습니다.

```python
if 오늘_잡은_물고기 == 오늘_판_물고기 :
    print('오늘 물고기 판매 끝!')
else :
    print('아직 물고기를 다 팔지 않았습니다.')
```

Python

`if` 문의 조건이 거짓일 경우에는 if문의 실행문이 실행되고 거짓을 경우에는 `else` 문의 실행문이 실행됩니다. 이 내용도 뒤에서 자세히 배울 내용이니 간단하게 알아보고 넘어가시길 바랍니다.

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

비교 연산자

Aa 기호	이름	설명	참(True)	거짓(Fal...
>	~보다 큰	왼쪽의 피연산자가 오른쪽의 피연산자보다 크면 참(True)를 반환하고 그렇지 않으면 거짓(False)를 반환합니다.	3 > 0	0 > 3
<	~보다 작은	왼쪽의 피연산자가 오른쪽의 피연산자보다 작으면 참(True)를 반환하고 그렇지 않으면 거짓(False)를 반환합니다.	3 < 0	3 > 0
>=	~보다 크거나 같은	왼쪽의 피연산자가 오른쪽의 피연산자보다 크거나 같으면 참(True)를 반환하고 그렇지 않으면 거짓(False)를 반환합니다.	3 >= 3 3 >= 0	0 >= 3
<=	~보다 작거나 같은	왼쪽의 피연산자가 오른쪽의 피연산자보다 작거나 같으면 참(True)를 반환하고 그렇지 않으면 거짓(False)를 반환합니다.	3 <= 3 3 <= 0	0 <= 3
==	같은	피연산자들이 같으면 참(True)를 반환하고 그렇지 않으면 거짓(False)를 반환합니다.	3 == 3	3 == 0
!=	다른	피연산자들이 다르면 참(True)를 반환하고 그렇지 않으면 거짓(False)를 반환합니다.	0 != 3	3 != 3

COUNT 6

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

5. 논리 연산자

논리 연산자는 boolean과 함께 쓰이며, 결과값으로 boolean 값(True, False)을 반환합니다. 논리 연산자에는 **and, or, not** 연산이 있습니다.

```python
#[In]

a = True
b = False

print(a and a) # True
print(a and b) # False
print(a or a)  # True
print(a or b)  # True
print(not a)   # False
print(not b)   # True
```

Python

and 연산은 피연산자 모두 참일 경우에만 True를 반환하며, **or 연산**은 피연산자 중 하나라도 참이면 True를 반환합니다. **not 연산**은 True일 때는 False를 False일 때는 True로 값을 반전시켜 반환합니다.

논리 연산자는 앞서 배웠던 비교 연산자와의 조합을 통해 조건을 더 까다롭고 명확하게 작성할 수 있습니다.

캣의 생선 가게에는 또 다른 규칙이 있었죠? 바로 오늘 매출은 전날 매출의 2배를 목표로 한다는 규칙입니다. 앞의 예제에 규칙을 추가해 봅시다.

```python
#[In]

오늘_잡은_물고기 = 10
오늘_판_물고기 = 10
전날_매출 = 10000
오늘_매출 = 20000

if (오늘_잡은_물고기 == 오늘_판_물고기) and (전날_매출*2 <= 오늘_매출):
    print('오늘 물고기 판매 끝!')
    print('매출 목표 달성!')
else :
    print('오늘의 목표를 달성하지 못했습니다.')
```

Python

and 연산을 사용하여 모두 참일 경우에는 판매 종료와 매출 목표 달성을 알리고 둘 중 하나라도 거짓일 경우에는 "오늘의 목표를 달성하지 못했습니다."라는 문구가 출력됩니다.

or 연산을 사용할 경우에는 둘 중 하나만 조건을 만족하면 실행문이 실행됩니다.

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO

```python
#[In]

if (오늘_잡은_물고기 == 오늘_판_물고기) or (전날_매출*2 <= 오늘_매출):
    print('오늘 물고기 판매 끝!')
    print('매출 목표 달성!')
```

Python

이번에는 not 연산을 사용해 봅시다. boolean 값을 변수에 저장하여 사용할 수도 있습니다.

```python
#[In]

결과값 = 오늘_잡은_물고기 == 오늘_판_물고기

if 결과값 :
    print('오늘 물고기 판매 끝!')

if not 결과값 :
    print('아직 물고기를 다 팔지 않았습니다.')
```

Python

논리 연산자

Aa 기호	이름	설명	참(True)	거짓(False)
and	그리고	양 쪽의 피연산자들 모두 참일 때만 참(True)을 반환하고 피연산자들 중 하나라도 거짓이면 거짓(False)를 반환합니다.	True and True	True and False False and True False and False
or	또는	양 쪽의 피연산자들 중 하나라도 참인 경우에 참(True)을 반환하고 피연산자들이 모두 거짓인 경우에만 거짓(False)를 반환합니다.	True or True True or False False or True	False or False
not	부정	오른쪽 피연산자의 논리 상태를 반전시킵니다.	not False	not True

COUNT 3

Project name :

DATE .

비고	Line	File name :				
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					

MEMO