파이썬 프로그래밍

10차시

논리 자료와 다양한 연산

⚠ 학습개요

… 논리 자료와 bool 자료형

… 관계 연산자

… 비트 논리 연산자

⚠ 학습목표

… 논리 자료를 활용할 수 있다.

… 관계 연산자와 비트 논리 연산자를 활용할 수 있다.

… 비트 배타적 논리합 ^을 사용하여 암호화를 할 수 있다.

Chapter 1.

논리 자료와 bool 자료형

PYTHON PROGRAMMING

⚠ 논리 유형 bool과 함수 bool( )

+ 논리 값으로 참과 거짓을 의미하는 True와 False를 키워드로 제공

- 클래스 bool

+ 내장 함수 bool(인자)

- 인자의 논리 값을 각각 1(True)과 0(False)으로 반환

⚠ 논리 유형 bool과 함수 bool( )

```
>>> print(True, False)
True, False
>>> type(True)
<class 'bool'>
```

```
>>> bool(0), bool(0,0), bool('')
(False, False, False)
>>> bool(10), bool(3.14), bool('python')
(True, True, True)
```

```
>>> int(True), int(False)
(1, 0)
```

끝

Chapter 2.

관계 연산자

PYTHON PROGRAMMING

⚠ 관계 연산

\+ > < >= <= == !=

```
>>> ord('a'), ord('A'), ord('\0'), ord('B')
(97, 65, 0, 66)
>>> 8 > 2,'a' > 'A'
(True, True)
>>> 8 >= 2,'a' >= 'A'
(True, True)
```

```
>>> 8 < 2,'a' < 'aB'
(False, True)
>>> 8 <= 2,'a' <= 'aB'
(False, True)
>>> 8 == 2,'a' == 'aB'
(False, False)
>>> 8 != 2,'a' != 'aB'
(True, True)
```

⚠ 관계 연산

+ [표] 대소 비교 관계 연산자

연산자	연산 사용	의미	문자열 관계 연산
>	a > b	크다(greater than).	사전 순서(lexicographically)에서 뒤에(코드 값이 크다)
>=	a >= b	크거나 같다 (greater than or equal to).	사전 순서에서 뒤에(코드 값이 크다) 있거나 동일
<	a < b	작다(lesser than)	사전 순서에서 앞에(코드 값이 작다)
<=	a <= b	작거나 같다 (less than or equal to).	사전 순서에서 앞에(코드 값이 작다) 있거나 동일
==	a == b	같다(equal to).	사전 순서에서 동일
!=	a != b	다르다(not equal to).	사전 순서와 다름

⚠ 관계 연산을 이용한 비만도 측정

✚ 체질량 지수(BMI: Body Mass Index)

- 키가 t 미터(m), 체중 w 킬로그램(kg)일 때
  - BMI = w / t^2
  - 여기서 키의 단위가 센티미터라면 계산식 $w / (t/100)^2$

⚠ 관계 연산을 이용한 비만도 측정

+ [표] BMI 기준과 관계 연산 표현

기준	판정	관계 연산 표현(BMI)
40이상	고도 비만	40 <= bmi
35 ~ 39.9	중등도 비만	35 <= bmi < 40
30 ~ 34.9	비만	30 <= bmi < 35
25 ~ 29.9	과체중	25 <= bmi < 30
18.5 ~ 24.9	정상	18.5 <= bmi < 25
18.5 미만	저체중	bmi < 18.5

⚠ 관계 연산을 이용한 비만도 측정

[코딩실습] 키와 몸무게로 비만도 지수 BMI 판정 난이도 응용

```
1. h, w = input('당신의 키(cm)와 몸무게(kg)는? >> ').split()
2. height = float(h)
3. weight = float(w)
4. bmi = weight / (height/100)**2
5. print('키:%6.1f(cm), 몸무게:%5.1f(km), BMI:%5.1f' % (height, weight, bmi))
6. print('{} {}'.format('고도 비만', 40 <= bmi))
7. print('{} {}'.format('중등도 비만', 35 <= bmi < 40))
8. print('{} {}'.format('비만', 30 <= bmi < 35))
9. print('{} {}'.format('과체중', 25 <= bmi < 30))
10.print('{} {}'.format('정상', 18.5 <= bmi < 25))
11.print('{} {}'.format('저체중', bmi < 18.5))
```

⚠ 관계 연산을 이용한 비만도 측정

결과

```
당신의 키(cm)와 몸무게(kg)는? >> 171.2 67.5
키: 171.2(cm), 몸무게: 67.5(kg), BMI: 23.0
고도 비만 False
중등도 비만 False
비만 False
과체중 False
정상 True
저체중 False
```

⚠ 관계 연산을 이용한 전기 기본 요금 계산

+ 논리 값 True와 False를 각각 1과 0으로 산술 연산에 활용

우리나라 가정용 전기 요금은 누진제이며, 전기 사용량에 따라 기본요금이 부과된다. 가정의 전기 사용량(kWh)을 입력 받아 기본 요금을 출력하는 프로그램을 작성하자.
누진제의 어느 구간에 속하는지 알려면 변수 3개에 저장한 후 다음 연산식을 사용해야 한다.

- `base = 730 * less200 + 1260 * less400 + 6060 * greater400`

기본요금(원/호)	
200kWh이하 사용	730
201~400kWh 사용	1,260
400kWh 초과 사용	6,060

⚠ 관계 연산을 이용한 전기 기본 요금 계산

[코딩실습] 전기 사용량의 기본 요금 계산 난이도 응용

```
1. usage = float(input('가정의 전기 사용량(kWh)은 >> '))
2. less200 = usage <= 200
3. less400 = 200 < usage <= 400
4. greater400 = 400 < usage
5.
6. base = 730 * less200 + 1260 * less400 + 6060 * greater400
7. print('전기 사용량(kw): %d, 기본 요금(원): %d' % (usage, base))
```

결과

```
가정의 전기 사용량(kWh)은 >> 180
전기 사용량(kw): 180, 기본 요금(원): 730
```

```
가정의 전기 사용량(kWh)은 >> 450
전기 사용량(kw): 450, 기본 요금(원): 6060
```

⚠ 검사 in

+ 한 단어를 표준 입력

- 그 단어가 지금까지 배운 파이썬의 키워드인지를 **in문**으로 검사해 결과를 출력

[코딩실습] 멤버십 검사 in으로 배운 파이썬 키워드 검사 난이도 기본

```
1. inkey = input( ' 배운 파이썬 키워드를 입력하세요 >> ' )
2. # inkey = ' key '
3. keywords = ' False ', ' True ', ' and ', ' in ', ' is ', ' not ', ' or '
4. print( ' 입력단어 {}, 키워드인가? {} ' .format(inkey, inkey in keywords))
```

결과

```
배운 파이썬 키워드를 입력하세요 >> and
입력 단어 and, 키워드인가? True
```

```
배운 파이썬 키워드를 입력하세요 >> const
입력 단어 const, 키워드인가? False
```

끝

Chapter 3.

비트 논리 연산자

PYTHON PROGRAMMING

⚠ 비트 논리곱 &, 비트 논리합 |, 비트 배타적 논리합 ^

+ [표]비트 논리 연산 이해

연산식	10진수	2진수 표현								설명
		128	64	32	16	8	4	2	1	
a	23	0	0	0	1	0	1	1	1	23의 2의 진수 00010111
b	57	0	0	1	1	1	0	0	1	57의 2의 진수 00111001
a&b	17	0	0	0	1	0	0	0	1	비트가 모두 1이면 1
a|b	63	0	0	1	1	1	1	1	1	비트가 하나라도 1이면 1
a^b	46	0	0	1	0	1	1	1	0	두 비트가 다르면 1, 같으면 0

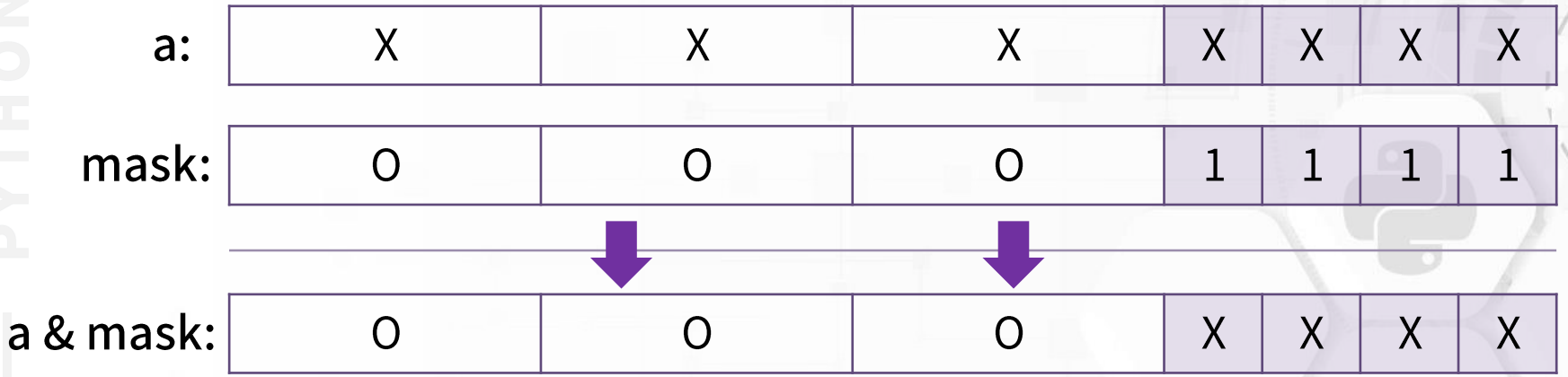
⚠ 비트 논리곱 &, 비트 논리합 |, 비트 배타적 논리합 ^

```
>>> a, b = 23, 57
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(a))
10진수 23, 2진수 00010111
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(b))
10진수 57, 2진수 00111001
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(a & b))
10진수 17, 2진수 00010001
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(a | b))
10진수 63, 2진수 00111111
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(a ^ b))
10진수 46, 2진수 00101110
```

⚠ 비트 논리곱 &로 특정 비트의 값 알아내기

✚ 마스크(mask): 원하는 특정 비트를 모두 1로 지정

- a & mask, a의 특정 비트 값만을 뽑아냄
  - a & mask(특정 비트만 1로): a의 특정 비트 값만 표시
  - 다음 mask = 0b1111은 가장 오른쪽 4비트를 추출해 내는 마스크로 사용



⚠ 비트 논리곱 &로 특정 비트의 값 알아내기

[코딩실습] 비트 연산자 &로 정수의 특정 비트 알아내기

난이도 응용

```
1. a = int(input('정수 하나를 입력하세요 >> '))
2. mask = 0b1111 # 0xf도 가능
3. print('정수 {0} 2진수로는 {0:b}'.format(a))
4. print('가장 오른쪽 4비트: {0:04b} 정수로는{0}'.format(a & mask))
```

결과

```
정수 하나를 입력하세요 >> 195
정수 195 2진수로는 11000011
가장 오른쪽 4비트: 0011 정수로는 3
```

```
정수 하나를 입력하세요 >> 57
정수 57 2진수로는 111001
가장 오른쪽 4비트: 1001 정수로는 9
```

⚠ 비트 배타적 논리합 ^을 사용, 암호화

+ 비트 배타적 논리합 특성

- a ^ a == 0, a ^ 0 == a, a ^ -1 == ~a
- a ^ b == b ^ a, (a ^ b) ^ c == a ^ (b ^ C)
- (a ^ b) ^ b == a ^ (b ^ b) == a ^ 0 == a

```
   (orgPwd ^ keyMask) ^ keyMask
=  orgPwd ^ (keyMask ^ keyMask)
=  orgPwd ^ 0
=  orgPwd
```

⚠ 비트 배타적 논리합 ^ 연산자

[코딩실습] 비트 배타적 논리합 ^으로 ID 암호화

난이도 응용

```
1. orgPwd = int(input('ID로 사용할 여덟 자리의 정수를 입력하세요 >> '))
2. keyMask = 27182818 # 키로 사용할 정수 하나를 저장
3. encPwd = orgPwd ^ keyMask # ID를 암호화시켜 저장
4. print('입력한 ID: %d' % orgPwd)
5. print('암호화해 저장된 ID: %d' % encPwd)
6. inPwd = int(input('로그인할 ID를 입력하세요>>'))
7. result = encPwd ^ keyMask # 키로 암호화된 것을 복호화
8. print('로그인 성공: {}'.format(inPwd == result))
```

(orgPwd ^ keyMask) ^ keyMask
= orgPwd ^ (keyMask ^ keyMask)
= orgPwd ^ 0
= orgPwd

결과

```
ID로 사용할 여덟 자리의 정수를 입력하세요
>> 87652877
입력한 ID: 87652877
암호화해 저장된 ID: 78101743
로그인할 ID를 입력하세요 >> 87652877
로그인 성공: True
```

```
ID로 사용할 여덟 자리의 정수를 입력하세요
>> 45678298
입력한 ID: 45678298
암호화해 저장된 ID: 52836408
로그인할 ID를 입력하세요 >> 456789299
로그인 성공: False
```

끝

⚠논리 자료와 bool 자료형

… True False

⚠논리 함수

… bool()

⚠관계 연산자

… < <= > >= == !=

⚠비트 논리 연산자

… & 비트 마스크에 활용

… |

… ^ 암호화 수행