Ch04 연산자

> 연산자(Operator)를 활용하여 피연산자(Operand)에 대한 연산(Operation)을 수행하는 방법을 익힌다.

연산자의 종류

> 일련의 연산을 결합하여 수식(Expression)으로 표현하여 컴퓨팅 작업 명령을 구성함.

(대입 연산) a = 3; b = 4;

(산술 연산) (a + b) / 2

(비교 연산) a >= b

(논리 연산) (a > b) or (a == b)

(비트 연산) a = a << 1

In [5]:
```
## 연산의 종류: 연산 결과 예측
a = 3; b = 4;  #대입 연산
c = (a + b) / 2  #산술 연산
d = a >= b  #비교 연산
e = (a > b) or ( a == b)  #논리 연산
a = a << 1; b = ~b + 1;  #비트 연산
```

In [7]:
```
## 연산 결과 확인
print(a, b, c, d, e)
```

```
6 -4 3.5 False False
```

산술 연산자

> 연산자 종류: +, -, *, /, %, //, **

연산 우선순위를 지정하기 위해서 ()를 적절히 사용함.

숫자 자료형과 문자 자료형 간의 연산은 불가능함.

산술 결과 자료형은 자료의 손실을 최소화 하는 방향으로 결정된다.

In [35]:
```
## 연산 결과 자료형 결정: 연산 결과 자료형 예측
a = 4; b = 2;
c = a * b
d = a / b
e = a // b
f = a % b
g = 4 * 2.0
```

In [37]:
```
## 연산 결과 자료형 결정: 연산 결과 자료형 확인
print(a, b, c, d, e, f, g)
print(type(c), type(d), type(e), type(f), type(g))
```

```
4 2 8 2.0 2 0 8.0
<class 'int'> <class 'float'> <class 'int'> <class 'int'> <class 'float'>
```

[실습] 원의 면적을 구해주는 코드 작성

반지름은 키보드로 입력을 받는다.

원의 면적을 계산하여 출력한다.

In [34]:
```
## [실습] 원의 면적을 구하는 연산
pi = 3.14159
r = float(input(">반지름(cm) 입력: "))
```

>> 반지름이 3.000000cm 원의 면적은 28.274310cm2입니다.

원주율 값 받아쓰기

math나 **numpy** 라이브러리를 임포트하여 사용

In [31]:
```
## 원주율 값 사용 방법[A]
import math

pi = math.pi
print("원주율 값:", pi)
```

```
원주율 값: 3.141592653589793
```

In [32]:
```
## 원주율 값 사용 방법[B]
import numpy as np

pi = np.pi
print("원주율 값:", pi)
```

원주율 값: 3.141592653589793

[실습] n일 후의 요일 알아보기 코드 작성

오늘로부터 **n**일이 경과한 날의 요일을 알려주는 코드 작성

> 오늘의 요일 값을 키보드로 입력을 받는다.

In [40]:
```
## [실습] n일 후의 요일 알아보기: 요일 index 값으로 해결하기
wd = int(input(">오늘의 요일 값(0:일,1:월,2:화,3:수,4:목,5금,6:토) 입력: "))
n = int(input(">몇일 후인가요? "))
```

>> 7일 후는 6요일입니다.

(알아두면 좋아요)

---

datetime 모듈 사용하기

.now() .weekday()

In [49]:
```
## datetime 모듈 사용하기
import datetime

# 현재 날짜와 시간을 가져오기
now = datetime.datetime.now()
# 년, 월, 일을 추출합니다.
year = now.year
month = now.month
day = now.day
print(str(year)+"/"+str(month)+"/"+str(day))

# 현재 요일의 인덱스를 가져오기. (월요일: 0, 화요일: 1, ..., 일요일: 6)
weekday_index = now.weekday()
print(weekday_index)
```

2024/3/22
4

In [44]:
```
## [실습] n일 후의 요일 알아보기: 요일 문자열 값으로 해결하기
weekday = "0123456"
weekname = "일월화수목금토"

import datetime
# 현재 날짜와 시간을 가져옵니다.
now = datetime.datetime.now()
# 현재 요일의 인덱스를 가져옵니다. (월요일: 0, 화요일: 1, ..., 일요일: 6)
weekday_index = now.weekday()

wd = (weekday_index + 1) % 7  #요일 index 조정
n = int(input(">몇일 후인가요? "))
result = (wd + n) % 7
print(">> %d일 후는 %c요일입니다." %(n, weekname[result]))
```

>> 1일 후는 토요일입니다.

In [ ]:
```
```

비교 연산자

> 연산자 종류: <, >, <=, >=, ==, !=

두 개의 피연산자를 비교하여 참(True 또는 1)이나 거짓(False 또는 0)으로 판별

비교 연산의 결과는 True 또는 False로 사용되지만 시스템 내부에서는 1과 0 정수값으로 관리된다.

In [183…
```
## 비교 연산자
a = 3; b = 4
c = a <= b  #연산 결과는 True 또는 False (boolean형)

print(c)
print(type(c))
```

True
<class 'bool'>

In [184…
```
## 비교 연산자
a = 3; b = 4
c = (a < b) + (a == b)  #연산 결과는 정수 값으로 저장
```

```
print(c)
print(type(c))
```

```
1
<class 'int'>
```

In [188…
```
## 비교 연산자
a = 4; b = 4
c = (a < b) + (a == b)  #연산 결과는 정수 값으로 저장

if c:
    print("b가 크거나 같음.")
```

```
b가 크거나 같음.
```

[예제] 나이 비교

In [ ]:
```
## 키보드로 나이를 입력받은 다음 누가 나이가 많은지를 출력
while True : #무한 반복
    age = int(input(">나이 입력: "))
    print("아~ %d살..." % age)
    myage = 30
    print("나하고 %d살 차이가 나네요." % abs(myage - age))

    if myage > age :
        print("그럼 내가 위네요.")  #True 시 수행
    elif myage < age :
        print("그럼 당신이 위네요.")
    else :
        print("그럼 동갑이네요.")

    yn = input(">>그만 하려면 x를 입력? ")
    if yn == 'x' or yn == 'X' :
        break  #반복 탈출
```

(알아두면 좋아요)

---

datetime 모듈 사용하기

.daytime() #날자 자료형 만들기

In [70]:
```
## import datetime: 내 나이 계산하기
import datetime
# 현재 날짜와 시간을 가져오기
now = datetime.datetime.now()

# 출생일로 생년월일 객체를 생성
mybirth = datetime.datetime(2000, 3, 25)  #My birthday
# 나이를 계산
myage = now.year - mybirth.year - ((now.month, now.day) < (mybirth.month, mybirth.day))
print("나이:", myage)
```

나이: 23

In [72]:
```
## import datetime: 날짜 (월, 일) 비교하기
import datetime
# 현재 날짜와 시간을 가져오기
now = datetime.datetime.now()

# 출생일로 생년월일 객체를 생성
mybirth = datetime.datetime(2000, 3, 25)  #My birthday
# 날짜 (월, 일) 비교
(now.month, now.day) < (mybirth.month, mybirth.day)
```

Out[72]: True

[실습] 내 나이를 환산해주는 방식으로 코드 변경

오늘 날자를 기준으로 내 나이(myage)를 결정해주 방식으로 변경

In [67]:
```
## [실습] 내 나이를 환산해주기


myage = 30

while True : #무한 반복
    age = int(input(">나이 입력: "))
    print("아~ %d살..." % age)
    print("나하고 %d살 차이가 나네요." % abs(myage - age))

    if myage > age :
        print("그럼 내가 위네요.")
    elif myage < age :
```

```
        print("그럼 당신이 위네요.")
    else :
        print("그럼 동갑이네요.")

    yn = input(">>그만 하려면 x를 입력? ")
    if yn == 'x' or yn == 'X' :
        break  #반복 탈출
```

```
아~ 33살...
나하고 10살 차이가 나네요.
그럼 당신이 위네요.
```

In [ ]:

논리 연산자

> 연산자 종류: and, or, not

두 개 이상의 조건(True 또는 False)을 결합하여 최종 조건(True 또는 False)을 판단하는 연산

In [189…

```
##논리 연산
a = 3; b = 4;
c = (a < b) or (a == b)  #논리 연산

print(c)
```

```
True
```

In [190…

```
##논리 연산: A or B
a = 3; b = 4;
if (a < b) or (a == b):  #논리, 비교 연산
    imax = b
else:
    imax = a
print(imax)
```

```
4
```

In [191…

```
##논리 연산: not(A or B) = not(A) and not(B)
a = 3; b = 4;
if not((a < b) or (a == b)):  #논리, 비교 연산
    imax = a
```

```
else:
    imax = b
print(imax)
```

4

In [192…

```
##논리 연산: not(A or B) = not(A) and not(B)
a = 3; b = 4;
if not(a < b) and not(a == b):  #논리, 비교 연산
    imax = a
else:
    imax = b
print(imax)
```

4

[실습] 할인 적용 여부 판단

> "할인" 조건: "국가유공자"인 경우 또는 "현역 군인"인 경우

In [78]:

```
## [실습] 할인 적용 여부 판단
print(">신분이 어떻게 되는지  y나 기타 키로 답하세요.")
status1 = input(">>국가유공자이신가요(y)? ").strip().lower()
status2 = input(">>현혁군인이신가요(y)? ").strip().lower()
```

>신분이 어떻게 되는지 y나 기타 키로 답하세요.

>>>할인 대상입니다.

[실습] "합격"과 "불합격" 판단

"합격" 조건은 다음과 같이 조건-A와 조건-B를 모두 만족시켜야 한다.

> 각 과목에서 40점 미만이 없어야 한다.

> 모든 과목의 평균 점수가 60점 이상이어야 한다.

In [92]:

```
## [실습] '합격', '불합격' 판단
score = [80, 63, 38, 88, 90]

avg = sum(score) / len(score)
smin = min(score)
print("Avg: %5.1f, Min: %d" %(avg, smin))
```

```
Avg:  71.8, Min: 38
불합격
```

In [ ]:

대입 연산자

> 연산자 종류: =

변수에 오른쪽의 값이나 연산 결과를 저장하는 연산자

산술 결과 자료형은 자료의 손실을 최소화 하는 방향으로 결정된다.

In [94]:
```
## 대입 연산자 사용: 기본 사용
a = 3
b = 4
c = a + b

print(a, b, c)
```

```
3 4 7
```

In [96]:
```
## 대입 연산자 사용: 목합 사용
a = 3; b = 4
c = a + b

print(a, b, c)
```

```
3 4 7
```

In [97]:
```
## 대입 연산자 사용: 목합 사용
a = b = c = 0

print(a, b, c)
```

```
0 0 0
```

In [98]:
```
## 대입 연산자 사용: 목합 사용
a, b, c = 3, 4, 5

print(a, b, c)
```

```
3 4 5
```

In [106…
```
## 대입 연산자 사용: 목합 사용
a = 3; b = 4
a, b = b, a  #교환 효과

print(a, b)
```

4 3

In [112…
```
## 대입 연산자 사용: 목합 사용
a = 3; b = 4; c = 5
a, b, c = c, b, a  #교환 효과

print(a, b, c)
```

5 4 3

복합 대입 연산자

> 연산자 종류: +=, -=, *=, /=, //=, %=

대입 연산자와 산술 연산자를 함께 표기한 연산자

앞에 연산을 한 후에 뒤의 연산(= 대입)을 하는 복합 대입 연산자

> a += 1 는 a + 1 연산 후에 그 결과를 a에 대입(=)하는 복합 연산

> a =+ 1 는 a = 1 연산 후에 a + 1 연산하는 복합 연산

In [204…
```
## 복합 대입 연산자
a = 3
a = a + 1
print(a)
```

4

In [205…
```
## 복합 대입 연산자: +=
a = 3
a += 1  # + 연산 후에 대입(=) 연산 수행
print(a)
```

4

In [206…
```
## 복합 대입 연산자: =+
a = 3
```

```
a =+ 1  # 대입(=) 연산 후에 + 연산 수행
print(a)
```

1

In [ ]:

비트 연산자

> 연산자 종류: &, |,^, ~, <<, >>

2진수(binary)의 각 비트 간의 논리 연산, 비트 이동**(shift)** 연산

In [116…
```
## 비트 연산자
a = 3
b = -3

print(bin(a), bin(b))
```

0b11 -0b11

In [138…
```
## 비트 AND 연산자: &
a = 0b1010
b = 0b0111
c = a & b

print(bin(a), bin(b), bin(c))
```

0b1010 0b111 0b10

In [137…
```
## 비트 OR 연산자: |
a = 0b1010
b = 0b0111
c = a | b

print(bin(a), bin(b), bin(c))
```

0b1010 0b111 0b1111

In [139…
```
## 비트 XOR 연산자: ^
a = 0b1010
b = 0b0111
c = a ^ b
```

```
print(bin(a), bin(b), bin(c))
```

0b1010 0b111 0b1101

In [126…
```
## 비트 NOT 연산자: ~
a = 0x23
b = 0xf0
c = ~a
d = ~b

print(bin(a), bin(b), bin(c), bin(d))
```

0b100011 0b11110000 -0b100100 -0b11110001

비트 마스크 연산

원하는 자리의 비트를 추출하기 위해 해당 위치의 비트 값을 논리곱(&) 1 수행

In [140…
```
## 비트 마스크 연산: a의 하위 4비트 추출
a = 0b01000001  #0x41
b = 0x00001111  #0x0f  #Mask
c = a & b   #Masking
d = a & ~b  #Masking

print(bin(a), bin(b), bin(c), bin(d))
```

0b1000001 0b1000100010001 0b1 0b1000000

비트 시프트 연산

비트열을 좌나 우로 원하는 수만큼 축차 이동시킴

> 빈 자리는 0으로 채움

비트는 위치에 따른 가중치가 있으므로 수의 *= 2 또는 /= 2 효과가 있음

In [148…
```
## 비트 시프트 연산
a = 0b00011010  #0x1a

for _ in range(8+1):
    print(bin(a))
    a = a >> 1
```

```
0b11010
0b1101
0b110
0b11
0b1
0b0
0b0
0b0
0b0
```

In [147…

```python
## 비트 시프트 연산
a = 0b00011010  #0x1a

for _ in range(8+1):
    print(bin(a))
    a = a << 1
```

```
0b11010
0b110100
0b1101000
0b11010000
0b110100000
0b1101000000
0b11010000000
0b110100000000
0b1101000000000
```

In [151…

```python
## 비트 시프트 연산
a = 0b1

for _ in range(10+1):
    print(a)
    a = a << 1
```

```
1
2
4
8
16
32
64
128
256
512
1024
```

In [150…

```python
## 비트 시프트 연산
a = 1024  #0b1000000000

for _ in range(10+1):
    print(a)
    a = a >> 1
```

```
1024
512
256
128
64
32
16
8
4
2
1
```

In [ ]:

```python

```

[실습] 원하는 위치의 비트 값 알아내기

대상 값은 10진숫자로 입력받는다.

> 입력된 대상 값을 2진수로 출력해준다.

추출할 비트의 위치 값을 입력받는다.

> 맨 왼쪽 비트를 0번째로 한 위치 값

추출한 비트 값을 출력한다.

In [157…

```
## 비트 연산자: 원하는 위치의 비트 값 알아내기
a = int(input(">대상 값(10진숫자)): "))
print(bin(a))
pos = int(input(">원하는 위치 값(맨 왼쪽 0번째 기준)): "))
```

0b100001

>> 비트 1

In [ ]:

[과제] 거스름 돈 반환 개수 최소화

키보드로 다음을 입력받아 처리

>청구한 금액 입력(원)?

>받은 금액 입력(원)?

다음 내용을 출력(단, 거스름 돈은 최소 개수가 되도록 하시오.)

>거스를 총 금액:

>50000원권 %d, 10000원권 %d, 5000원권 %d, 1000원권 %d

>500원권 %d, 100원권 %d, 50원권 %d, 10원권 %d, 1원권 %d

In [166…

```
## [예제] 거스름 돈 반환 개수 최소화: 금액 입력 처리
totwon = 0
inwon = 0
outwon = 0

while inwon <= totwon:
    totwon = int(input(">청구한 금액 입력(원)? "))
    inwon = int(input(">받은 금액 입력(원)? "))

outwon = inwon - totwon    ##거스를 금액
print(">>거스를 총 금액 : %d" %outwon)
```

>>거스를 총 금액 : 8766

In [165…

```
## [과제] 거스름 돈 반환 개수 최소화: 금액 입력 처리
totwon = 0
inwon = 0
outwon = 0

while inwon <= totwon:
    totwon = int(input(">청구한 금액 입력(원)? "))
    inwon = int(input(">받은 금액 입력(원)? "))

outwon = inwon - totwon    ##거스를 금액
print(">>거스를 총 금액 : %d" %outwon)
```

```
>>거스를 총 금액 : 8766
5000원권 1
1000원권 3
500원권 1
100원권 2
50원권 1
10원권 1
1원권 6
```