Variable & Operator

Python Basic

- 파이썬의 구성 요소
  - ✓ Literal: 개발자가 직접 입력한 데이터
  - ✓ Variable(변수): 데이터를 재사용하기 위해서 데이터를 저장한 공간에 붙인 이름
  - ✓ Function(함수 – Method): 자주 사용하는 코드를 이름만으로 사용할 수 있도록 묶어 놓은 것
  - ✓ Class(클래스)와 Instance(객체 – Object): 동일한 목적을 위한 데이터 와 기능은 묶어 놓은 것으로 이 안에 선언된 데이터 와 기능은 클래스와 객체를 통해서만 사용
  - ✓ Module(모듈): 여러 구성 요소들을 묶어 놓을 수 있는 것으로 하나의 파일
  - ✓ Package(패키지): 관련 있는 모듈의 모임으로 압축된 형태나 디렉토리 형태로 제공되는데 파이썬의 라이브러리 배포 및 설치 단위
  - ✓ 주석: 번역하지 않는 문장으로 # 다음에 입력하면 되는데 대부분의 용도는 코드에 대한 설명

Python Basic

- 파이썬 코딩
  - ✓ 라인 단위로 번역해서 실행하므로 한 줄에 하나의 명령이 있다면 종료 부호는 필요하지 않음
  - ✓ 하나의 라인에 2개 이상의 실행 문장을 사용할 때 문장을 구분하기 위한 용도로 ;을 사용
  - ✓ 코드의 구조(Block)를 정의하기 위해 들여쓰기를 사용
  - ✓ 코드의 정렬과 구성이 엄격하게 제한되는데 네 칸를 들여쓰는 것이 표준이지만 프로그래머가 들여쓰기의 규칙을 정할 수 있음
  - ✓ 하위 레벨에 코드를 작성할 때는 반드시 앞 절의 마지막에 :을 추가
  - ✓ console에 내용을 출력하고자 하는 경우는 print(내용)의 형태로 출력할 수 있으며 여러 내용을 출력할 때는 ,로 구분해서 출력

Python Basic

❖ 들여쓰기

```
num = 20
#블록마다 들여쓰기를 다르게 설정해도 되지만 권장하지 않음
if num >= 10:
    print("10보다 크다")
else:
  print("10보다 작다")
```

10보다 크다

Python Basic

- 주석(Comment)
  - ✓ 컴파일러나 인터프리터가 해석하지 않는 문장
  - ✓ 파이썬에서 한 줄 주석은 # 기호 뒤에 작성하는데 # 기호 뒤에 있는 문자열은 실행을 할 때 무시
  - ✓ 어떤 특정 구역을 문자열로 만들어서 실행되지 않도록 하려면 " " " 에서 " " " 까지 큰 따옴표(쌍 따옴표) 3개를 사용해서 묶으면 되며 작은 따옴표도 가능한데 이 구문은 주석을 만드는 것이 아니고 문자열 상수를 만들어서 아무 일도 하지 않는 것처럼 보이도록 함
  - ✓ #! 기호는 주석이 아니고 유닉스 Shebang(프로그램으로서 실행)
  - ✓ # -*- coding: cp949 -*- 는 Encoding 지정문(윈도우에서는 cp949, ms949 나 euc-kr 그 이외의 운영체제에서는 utf-8로 지정)
  - ✓ Encoding & Decoding
    - Encoding: 문자를 컴퓨터에 저장할 수 있는 코드로 변경하는 작업
    - Decoding: 컴퓨터에 저장된 코드를 출력하기 위해서 문자로 변경하는 작업

Python Basic

- 파이썬 주석

```
# 이 줄은 라인 코멘트입니다
print("Hello World!")

print("Hello World!")   # 이것도 라인 코멘트입니다


"""
이것은 블럭 주석처럼 사용할 수 있음
큰따옴표 나 작은 따옴표 3개를 연속으로 적으면 주석이 아니고 여러 줄 문자열
"""
```

```
Hello World!
Hello World!
Hello World!
```

Python Basic

- 출력 과 함수의 도움말
  - ✓ print(dir(자료형 또는 데이터))를 호출해서 자료형이나 데이터가 사용 가능한 속성이나 함수의 목록을 확인
  - ✓ 함수나 클래스의 도움말을 확인하고자 하는 경우에는 help(도움말을 얻고자 하는 함수나 클래스 또는 객체 이름)

Python Basic

- 출력 과 함수의 도움말
  - ✓ 가능한 작업 확인과 함수의 도움말 확인

```
#정수를 가지고 할 수 있는 작업 확인
print(dir(100))

#print에 도움말
help(print)
```

Python Basic

- 출력 과 함수의 도움말
  - ✓ 가능한 작업 확인과 함수의 도움말 확인

```
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delattr__', '__dir__', '__divmod__', '__doc__',
'__eq__', '__float__', '__floor__', '__floordiv__', '__format__', '__ge__', '__getattribute__', '__getnewargs__', '__gt__',
'__hash__', '__index__', '__init__', '__init_subclass__', '__int__', '__invert__', '__le__', '__lshift__', '__lt__',
'__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__',
'__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__round__',
'__rpow__', '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__sizeof__', '__str__',
'__sub__', '__subclasshook__', '__truediv__', '__trunc__', '__xor__', 'bit_length', 'conjugate', 'denominator', 'from_bytes',
'imag', 'numerator', 'real', 'to_bytes']
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Python Basic

- python의 예약어
  - ✓ 예약어(Reserved Words - keyword): 파이썬이 문법적인 용도로 사용되고 있는 단어 또는 문자
  - ✓ 예약어에 사용자가 기능을 설정하면 에러
  - ✓ 예약어 확인

```
#파이선 예약어 확인
import keyword
print(keyword.kwlist)
```

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Python Basic

- 모듈을 찾는 순서 확인

```
#현재 사용 중인 파이썬에서 모듈을 찾는 순서 확인
import sys
print(sys.path)
```

```
========== RESTART: //Mac/Home/Documents/source/python/second.py
==========
['//Mac/Home/Documents/source/python',
'C:₩₩Users₩₩munseokpark₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python37₩₩Lib₩₩idlelib',
'C:₩₩Users₩₩munseokpark₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python37₩₩python37.zip',
'C:₩₩Users₩₩munseokpark₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python37₩₩DLLs',
'C:₩₩Users₩₩munseokpark₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python37₩₩lib',
'C:₩₩Users₩₩munseokpark₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python37',
'C:₩₩Users₩₩munseokpark₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python37₩₩lib₩₩site-packages']
```

Data

❖ 파이썬에서 제공하는 기본 데이터의 종류

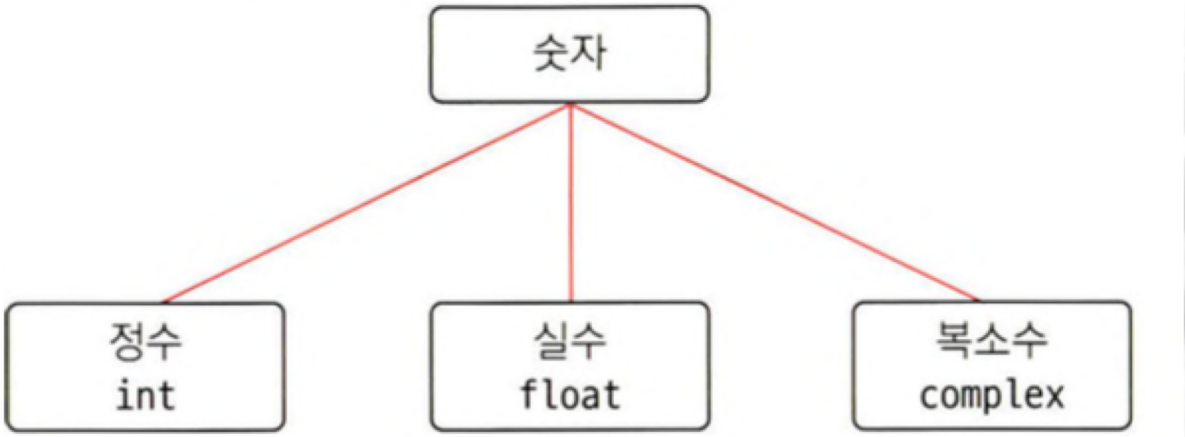
	자료형	저장 모델	변경 가능성	접근 방법
수치형	int, float, complex	Scala	Immutable	Direct
문자열	str	Container		Sequence
튜플	tuple			
리스트	list		Mutable	
사전	dict			Mapping
집합	set			Set

Data

- 파이썬에서 제공하는 기본 데이터의 종류
  - ✓ 저장 모델
    - Scala: 하나의 데이터
    - Container(Vector, Collection): 데이터의 모임
  - ✓ 변경 가능성
    - Immutable: 변경 불가
    - Mutable: 변경 가능
  - ✓ 접근 방법
    - Direct: 직접 할당
    - Sequence: 순서 중시
    - Mapping: 순서 무관
    - Set: 중복 불가

Data

- Literal
  - ✓ 소스 코드의 고정된 값을 나타내는 표기법
  - ✓ Scala Data 표현
    - 숫자
      - 정수(int): 10진수(107), 8진수(0o107), 16진수(0x107, 0X107), 2진수(0b101), long(107L)
      - 실수(float): 1.2, 0.12e1, 0.12E1 ->지수 부분은 정수일 수 있으나 실수 일 수 없음
      - 복소수(complex): 허수 부분 뒤에 j 나 J 사용 4+5j, 7-2J
    - bool: True(1), False(0)



Data

- Literal
  - ✓ Vector Data 표현
    - str: 0개 이상의 문자 집합으로 '문자열' 또는 "문자열" 의 형태로 기재하며 여러 줄의 문자열을 하나의 문자열로 만들 때는 ''' 문자열''' 또는 """문자열"""
    - bytes: byte의 집합으로 b'문자열' 또는 b'₩코드₩코드...' 으로 생성
    - list: 비교 가능한 데이터의 모임으로 [데이터 나열]의 형태로 표현
    - tuple: 비교 불가능한 여러 데이터의 모임으로 (데이터 나열)의 형태로 표현
    - set: 중복 없는 데이터의 모임으로 {데이터 나열}의 형태로 표현
    - dict: 데이터에 이름을 부여한 모임으로 {키:값, 키:값, ...}의 형태로 표현
    - list, tuple, set, str, dict 등의 데이터 집합을 for를 이용해서 순차적으로 접근할 수 있다고 해서 iterable 이라고도 함
  - ✓ 제어 문자: 하나의 문자가 의미를 갖는 명령어가 되는 것으로 앞에 ₩를 붙여서 ₩n(줄바꿈), ₩t(탭), ₩₩의 형태로 표현하는데 예전 윈도우즈에서는 ₩r₩n으로 줄 바꿈을 할 수 있음
  - ✓ None: Types.NoneType 의 유일한 값으로 값이 존재하지 않는 변수에 대입하여 이 변수에 아무런 값이 없다는 것을 나타내기 위한 목적으로 활용하는데 NaN이라고도 하고 null, nil 또는 결측치 라고도 함

Data

- Identifier(식별자)
  - ✓ 프로그래머가 기능을 정하는 명령어
  - ✓ 식별자는 변수, 함수, 클래스, 객체, 모듈, 패키지를 식별하는데 사용되는 이름
    - 영문 문자 와 숫자를 사용할 수 있으며 한글도 가능
    - 대소문자 구별
    - 식별자는 문자 A~Z 또는 a~z과 언더바(_)로 시작해야 하며 숫자부터 시작할 수 없음
    - 특수문자(+, -, *, /, @, $, % 등)는 식별자로 사용할 수 없음
      다음과 같은 것은 식별자가 될 수 없음 : 1abc, @file, %x
    - 파이썬의 키워드는 식별자로 사용할 수 없음
      SyntaxError: invalid syntax
    - 기존에 존재하는 이름들과 동일한 이름의 식별자를 사용하면 식별자의 기능이 변경됨
    - 블럭이 다르면 동일한 이름으로 다른 기능을 수행할 수 있도록 할 수 있음

Data

- Variable
  - ✓ 데이터를 저장해두고 다시 사용할 수 있도록 저장 공간에 붙여놓는 이름
  - ✓ python은 변수를 생성할 때 Data Type을 기재하지 않고 이름에 값을 대입해서 생성
  - ✓ 영역 내에 동일한 이름이 없을 때는 변수를 생성하고 동일한 영역에 동일한 이름의 변수가 존재한다면 변수의 값을 수정
  - ✓ 변수에 값을 할당 할 때 Data Type을 자동으로 설정
  - ✓ 등호 (=)는 데이터에 변수 이름과 데이터를 매핑해서 참조하도록 함
  - ✓ 변수가 = 의 왼쪽에 사용된 경우에는 변수의 의미를 갖지만 그 이외의 경우는 변수가 저장하고 있는 참조를 의미
  - ✓ 파이썬에서 대입

    a = 1 #1이라는 값을 저장한 공간의 참조를 a에 대입
  - ✓ 변수 명명 규칙은 식별자 규칙을 적용
  - ✓ 변수의 삭제는 **del 변수명**: 데이터가 삭제되는 것이 아니고 변수의 이름이 삭제되서 데이터를 변수를 이용해서 참조할 수 없는 상태가 됨
  - ✓ 여러 개를 한 번에 생성 가능

    x, y, z = 10, 20, 30

Data

- 변수를 생성하고 값을 대입하고 수정한 후 출력

```
#변수를 생성해서 값을 대입
counter = 100          # 정수 할당
miles   = 1000.0       # 부동 소수점
name    = "홍길동"     # 문자열

#변수의 값을 콘솔에 출력
print(counter)
print(miles)
print(name)

#반복문을 이용해서 counter 변수에 새로운 값을 대입
for counter in [1,2,3]:
    print(counter)

#기존에 저장된 값 대신에 마지막에 저장한 값을 출력
print(counter)
```

```
100
1000.0
홍길동
1
2
3
3
```

Variable

- 예약어에 데이터를 할당

```
and = 1
print(and)
```

```
 File "/var/folders/42/fxl3_n31121f16yj69fbrb_m0000gn/T/ipykernel_1445/3701783566.py",
     line 1
    and = 1
    ^
SyntaxError: invalid syntax
```

Variable

- 기존 존재하는 이름에 데이터를 할당

```
print(abs(-3))
abs = 1
print(abs(-3))

3
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/42/fxl3_n31121f16yj69fbrb_m0000gn/T/ipykernel_1445/1040105075.py in
      <module>
       1 print(abs(-3))
       2 abs = 1
----> 3 print(abs(-3))

TypeError: 'int' object is not callable
```

Operator

- 할당 연산자
  - ✓ =: 오른쪽의 데이터를 저장하고 그 공간에 왼쪽의 이름을 설정

Operator

- 산술 연산자
  - ✓ +
    - 동일한 종류의 데이터에 사용할 수 있는 기호로 숫자의 경우는 덧셈을 하고 데이터의 모임은 결합을 수행
    - 서로 다른 종류의 데이터 연산은 불가능
  - ✓ -: 뺄셈
  - ✓ *: 숫자 데이터끼리는 곱셈이고 데이터의 모임과 정수에 사용하면 정수 개수만큼 반복
  - ✓ **: 거듭제곱(4**3 은 4의 3제곱)
  - ✓ /: 나눗셈을 한 결과 – 결과는 실수로 리턴
  - ✓ //: 나눗셈을 한 후 소수를 버리고 몫을 정수로 리턴
  - ✓ %: 나머지를 구해주는 연산자로 주기적으로 일정한 패턴 작업을 반복해야 하는 작업을 만들 때 이용

Operator

- 산술 연산자

```
a = 11
b = 2

add = a + b
sub = a - b
mul = a * b
div = a / b
mok = a //b
mod = a % b

print("a+b=",add)
print("a-b=",sub)
print("a*b=",mul)
print("a/b=",div)
print("a//b=",mok)
print("a%b=",mod)
```

```
a+b= 13
a-b= 9
a*b= 22
a/b= 5.5
a//b= 5
a%b= 1
```

Operator

❖ 나머지 연산자

```
#소녀시대 레드벨벳 블랙핑크 ITZY를 번갈아 보여주기
import time
i = 0
while i < 12:
    result = i % 4
    i = i + 1
    time.sleep(1)

    if result == 0:
        print("소녀시대")
    if result == 1:
        print("레드벨벳")
    if result == 2:
        print("블랙핑크")
    if result == 3:
        print("ITZY")
```

```
소녀시대
레드벨벳
블랙핑크
ITZY
소녀시대
레드벨벳
블랙핑크
ITZY
소녀시대
레드벨벳
블랙핑크
ITZY
```

Operator

- 비교 연산자
  - ✓ 소유하고 있는 데이터의 크기나 동일성 여부를 판단하는 연산자로 결과가 True 또는 False
  - ✓ a = 10, b = 20 이라 가정
    - 숫자 데이터 와 bool 데이터 그리고 문자열 데이터에서 사용 가능한 연산자
      - >: 왼쪽 값이 오른쪽 값보다 큼 (a > b) → False
      - <: 왼쪽 값이 오른쪽 값보다 작음 (a < b) → True
      - >=: 왼쪽 값이 오른쪽 값보다 크거나 동일 (a >= b) → False
      - <=" 왼쪽 값이 오른쪽 값보다 작거나 동일 (a <= b) → True
    - 모든 종류의 데이터에서 가능
      - ==: 값이 동일함 (a == b) → False
      - !=: 값이 동일하지 않음 (a != b) → True
  - ✓ bool 데이터를 숫자 데이터와 비교하면 True는 1로 간주하고 False는 0 으로 간주
  - ✓ 산술 연산자 와 같이 사용하면 산술 연산자를 먼저 수행

Operator

❖ 비교 연산자

```
#크기 확인
pan = 10 > 3;
print("pan:",pan);

#동일성 확인
pan = 10 == 3;
print("pan:",pan);

date = '2019-01-01'
print(date == '2019-01-01')
print(date == '2018-12-31')

#문자의 크기 비교는 첫글자부터 순서대로 비교
#알파벳은 대문자와 소문자를 구분하며 소문자가 대문자보다 큼
print("레드벨벳" > "블랙핑크")
print("BlackPink" > "blackpink")

#연산 순서
print(10 == 10 + 1)
```

```
pan: True
pan: False
True
False
False
False
False
```

Operator

- 산술 비트 연산자
  - ✓ 정수 데이터를 2진수로 변환해서 비트 단위로 연산을 수행 한 후 결과를 10진 정수로 리턴하는 연산자로 시스템 프로그래밍이나 데이터 분석 분야에서 많이 이용
  - ✓ &: AND 연산으로 둘 다 1일 때만 1
  - ✓ |: OR 연산. 둘 중 하나만 1이여도 1
  - ✓ ^: XOR 연산. 두 개의 데이터가 다르면 1
  - ✓ ~: 1의 보수 연산. (~a) = -61 → 1100 0011
  - ✓ <<: 왼쪽 시프트 연산자. 변수의 값을 왼쪽으로 지정된 비트 수 만큼 이동해주는 연산자로 한 번 이동할 때 마다 곱하기 2 한 효과
  - ✓ >>: 오른쪽 시프트 연산자. 변수의 값을 오른쪽으로 지정된 비트 수 만큼 이동해주는 연산자로 한 번 이동할 때마다 나누기 2한 효과

Operator

- 정수의 숫자 표현 과 비트 연산

  a = 20; 라고 변수를 초기화하게 되면 실제로 컴퓨터에 저장된 형태는 다음과 같음

  00000000 00010100

  b = -13 이렇게 초기화하면 변수 b는 다음과 같이 저장

  음수는 양수의 2의 보수로 표현하고 십진수로 변경할 때는 2의 보수를 구한 후 양수 값을 구하고 - 기호를 앞에 추가

  11111111 11110011

  a & b의 결과

  00000000 00010000 연산의 결과는 10진수 16

  a | b의 결과

  11111111 11110111 연산의 결과는 10진수 -9

  a ^ b의 결과

  11111111 11100111 연산의 결과는 10진수 -25

  b<<2 의 결과

  11111111 11001100 연산의 결과는 10진수 -52

Operator

- 비트 연산자

```
a = 20 #00000000 00000000 00000000 00010100
b = 12 #00000000 00000000 00000000 00001100

print("a&b=",a&b)
print("a|b=",a|b)
print("a^b=",a^b)
print("a<<2=", a<<2)
print("a>>2=", a>>2)

print()
b = -13 #11111111 11111111 11111111 11110011
print("a&b=",a&b)
print("a|b=",a|b)
print("a^b=",a^b)
print("b<<2=", b<<2)
print("b>>2=", b>>2)
```

```
a&b= 4
a|b= 28
a^b= 24
a<<2= 80
a>>2= 5

a&b= 16
a|b= -9
a^b= -25
b<<2= -52
b>>2= -4
```

Operator

- 논리 비트 연산자
  - ✓ bool 식을 연산해서 결과를 bool 타입으로 리턴하는 연산자
  - ✓ a = True, b = False
    - and: 논리 AND 연산. 둘 다 참 일 때 만 참 (a and b) = False
    - or: 논리 OR 연산. 둘 중 하나만 참 이여도 참 (a or b) = True
    - not: 논리 NOT 연산. 논리 상태를 반전 not(a and b) = True
  - ✓ and는 앞의 결과가 False 이면 뒤의 결과가 무엇이든 간에 False 가 되므로 뒤의 결과를 확인하지 않으며 or는 앞의 결과가 True 이면 뒤의 결과가 무엇이든 간에 True가 되므로 뒤의 결과를 확인하지 않음
  - ✓ 조건의 순서를 변경해도 결과는 변하지 않지만 조건을 확인하는 횟수는 다름

Operator

❖ 윤년 판별

```
#윤년은 2개의 조건 중 하나를 만족하면 됩니다.
#1. 4의 배수이고 100의 배수는 아닌 경우
#2. 400의 배수인 경우

year = 2019
leafcheck = year % 4 == 0 and year % 100 != 0 or year % 400 == 0
print(year, "의 윤년 여부:", leafcheck)

year = 2020
leafcheck = year % 4 == 0 and year % 100 != 0 or year % 400 == 0
print(year, "의 윤년 여부:", leafcheck)
```

```
2019 의 윤년 여부: False
2020 의 윤년 여부: True
```

Operator

- 12의 배수 구하기

```
#1-100까지에서 3의 배수이고 4의 배수 개수 구하기
cnt = 0
loop = 0
for i in range(1,101):
    loop = loop + 1
    if i % 3 == 0:
        loop = loop + 1
        if i % 4 == 0:
            cnt = cnt + 1
print("12의 배수 개수: ", cnt, "개")
print("루프 횟수: ", loop, "번")
print("==================")

cnt = 0
loop = 0
for i in range(1,101):
    loop = loop + 1
    if i % 4 == 0:
        loop = loop + 1
        if i % 3 == 0:
            cnt = cnt + 1
print("12의 배수 개수: ", cnt, "개")
print("루프 횟수: ", loop, "번")
```

```
12의 배수 개수:  8 개
루프 횟수:  133 번
==================
12의 배수 개수:  8 개
루프 횟수:  125 번
```

Operator

- 논리 함수
  - ✓ all(목록): 목록 내의 모든 원소가 True이면 True 리턴
  - ✓ any(목록): 목록 내의 원소 중 하나라도 True이면 True 리턴

```
L1 = [True, True, False]
print(L1)
print(all(L1))
print(any(L1))

L2 = [1, 1, 0]
print(L2)
print(all(L2))
print(any(L2))

L3 = [3 > 0, 3 > 2, 2 == 2]
print(L3)
print(all(L3))
print(any(L3))
```

```
[True, True, False]
False
True

[1, 1, 0]
False
True

[True, True, True]
True
True
```

Operator

- 복합 할당 연산자
  - ✓ 연산자 축약
  - ✓ 변수명 연산자= 데이터: 왼쪽의 변수에 저장된 데이터와 오른쪽의 데이터를 연산한 후 다시 왼쪽의 변수에 저장하는 연산자
    - += 왼쪽 변수의 데이터에 오른쪽 값을 더하고 결과를 왼쪽변수에 할당 c += a → c = c + a
    - -= 왼쪽 변수의 데이터에서 오른쪽 값을 빼고 결과를 왼쪽변수에 할당 c -= a → c = c - a
    - *= 왼쪽 변수의 데이터에 오른쪽 값을 곱하고 결과를 왼쪽변수에 할당 c *= a → c = c * a
    - /= 왼쪽 변수의 데이터에서 오른쪽 값을 나누고 결과를 왼쪽변수에 할당 c /= a → c = c / a
    - %= 왼쪽 변수의 데이터에서 오른쪽 값을 나눈 나머지의 결과를 왼쪽변수에 할당 c %= a → c = c % a
    - **= 왼쪽 변수의 데이터에 오른쪽 값만큼 제곱을 하고 결과를 왼쪽변수에 할당 c **= a → c = c ** a
    - //= 왼쪽 변수의 데이터에서 오른쪽 값을 나눈 몫의 결과를 왼쪽변수에 할당 c //= a → c = c // a

Operator

- 연산자 축약

```
a = 10
a += 5 # a = 10 + 5
print("a:", a)
a *= 2 # a = 15 * 2
print("a:", a)
```

a: 15
a: 30

Operator

- 기타 연산자
  - ✓ type(데이터): 데이터의 자료형을 문자열로 리턴
  - ✓ id(데이터): 저장 위치를 구분하기 위한 코드 리턴
  - ✓ 자료형 확인 및 저장된 데이터의 참조 코드 확인

```
#자료형 확인 및 저장된 데이터의 참조 코드 확인
a = 20
b = 20
print("a가 가리키고 있는 곳의 id:", id(a))
print("a의 자료형:", type(a))
print("b가 가리키고 있는 곳의 id:", id(b))

a = 10 + 20
print("a가 가리키고 있는 곳의 id:", id(a))
b = 30
print("b가 가리키고 있는 곳의 id:", id(b))

a = 30 + 40
print("a가 가리키고 있는 곳의 id:", id(a))
```

```
a가 가리키고 있는 곳의 id: 4332919696
a의 자료형: <class 'int'>
b가 가리키고 있는 곳의 id: 4332919696
a가 가리키고 있는 곳의 id: 4332920016
b가 가리키고 있는 곳의 id: 4332920016
a가 가리키고 있는 곳의 id: 4333109776
```

Operator

❖ 연산자 우선 순위(Operators Precedence)

연산	의미
**	거듭 제곱
~x, +x, -x	단항 연산
*, @, /, //, %	곱셈 및 나누기 연산
«, »	Bitwise Shift
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
>, >=, <, <=	비교 연산자
==, !=	비교 연산자(평등)
in, not in, is, is not	식별 연산자
not	논리 연산자
and	논리 연산자
or	논리 연산자

Operator

- 데이터 자료형 변환
  - ✓ 정수 변환
    - int(숫자 데이터): 실수의 경우 소수는 버림
    - int(정수로 변환이 가능한 문자열): 실수 문자열은 안됨
    - int(bool 데이터)
  - ✓ 실수 변환
    - float(숫자 데이터)
    - float(숫자로 변환이 가능한 문자열)
    - float(bool 데이터)
  - ✓ bool 변환
    - bool(숫자 데이터)
    - bool(변환이 가능한 문자열)
  - ✓ 문자열 변환은 str(데이터)

Operator

- 데이터 자료형 변환

```
print(int(10.7))
print(int("10"))
#print(int("10.7"))
print(int(True))
print("==========")

print(float("10.7"))
print(float(True))
print("==========")

print(bool(20))
print(bool(0.0))
print(bool(True))
print("==========")

print(str(10.7))
```

```
10
10
1
==========
10.7
1.0
==========
True
False
True
==========
10.7
```

bool

- bool 자료형

```
a = True
b = False

#True 는 상황에 따라 1 False 는 0으로 해석
print(a == 1)     # True가 출력됨
print(b != 0)     # False가 출력됨
print(a + b)      # 1
print(not a)      #False
```

```
True
False
1
False
```

콘솔 입출력

- Console 입출력
  - ✓ 콘솔 (Console): Windows에서는 Command 창, 리눅스/맥에서는 Terminal 창이며 각 IDE에서는 별도의 콘솔 창을 제공
  - ✓ print(데이터 나열): 화면으로 데이터를 출력할 때 가장 보편적으로 사용되는 함수로 데이터를 출력하고 줄 바꿈을 수행
  - ✓ 여러 개의 데이터를 ,로 구분해서 출력
  - ✓ end라는 매개변수를 이용해서 줄 바꿈을 대신할 문자열을 설정
  - ✓ sep 매개변수를 이용해서 여러 개의 데이터를 한꺼번에 출력할 때 각 데이터를 구분하는 문자열을 설정

콘솔 입출력

❖ 콘솔 출력

```
print (1,2)
print (3,4)

print(1,2, end= ' ')
print(3,4)

print(1,2,3,4, sep='₩t')
```

```
1 2
3 4
1 2 3 4
1	2	3	4
```

콘솔 입출력

- 콘솔 출력
  - ✓ 출력할 때 서식을 설정하고 뒤에 포맷 서식을 붙여서 출력이 가능
  - ✓ 문자열 안에 서식을 설정하고 이어서 %(데이터 나열)를 추가 설정

서식	설명
%s	문자열 (String)
%d, %i	정수 (Integer)
%f	실수 (Floating-point)
%%	Literal % (문자 % 자체)

- 서식은 % 다음에 정수를 추가해서 전체 자릿수를 확보해서 생성할 수 있는데 데이터의 길이보다 작은 숫자를 설정하면 데이터 전체를 출력
- 실수의 경우는 전체 자릿수와 소수 자릿수를 .으로 구분해서 설정할 수 있는데 소수는 그 이하 자리에서 반올림한 결과를 출력하는데 전체 자릿수는 생략 가능
- 모든 데이터는 %s로 서식 설정 가능

콘솔 입출력

❖ 서식을 이용한 출력

```
txt1 = '자바'
txt2='파이썬'
num1= 5
num2=10
num3=3.141569
print('나는 %s보다 %s에 더 익숙합니다.' %(txt1, txt2))
print('%d + %d = %d' %(num1, num2, num1+num2))
print('작년 세계 경제 성장률은 전년에 비해 %d%% 포인트 증가했다.' %num1)
print('파이는 소수 세째 자리에서 반올림 하면 %.2f 입니다.' %num3)
```

나는 자바보다 파이썬에 더 익숙합니다.
5 + 10 = 15
작년 세계 경제 성장률은 전년에 비해 5% 포인트 증가했다.
파이는 소수 세째 자리에서 반올림 하면 3.14 입니다.

콘솔 입출력

- '{ }'.format()
  - ✓ str.format()이라고도 하는데 %-formatting과 결과는 같지만 좀 더 파이썬 스러운 방법
  - ✓ 중괄호{}를 사용하는 방법으로 .format() 괄호 안에 중괄호에 대입할 문자나 숫자를 입력
  - ✓ 중괄호{}에 들어갈 순서 지정도 가능

```
print("My name is {}".format('Adam'))
print('{}은 {}를 좋아합니다.'.format('Adam', '신촌 라이브 카페'))
print('{1}은 {0}을 보고 만들었습니다.'.format('원빈', 'Adam'))
```

My name is Adam
Adam은 신촌 라이브 카페를 좋아합니다.
Adam은 원빈을 보고 만들었습니다.

콘솔 입출력

- f-Strings
  - ✓ 새로운 python 문자열(strings) formatting 방법으로 '{}'.format()과 사용이 비슷하지만 훨씬 코드가 깔끔
  - ✓ 먼저 변수에 값을 대입한 후 중괄호{}에 변수명을 넣는데 이때 중괄호가 들어가는 코드 처음에 f를 입력함
  - ✓ 대문자 F도 가능

```
name = 'Adam'
birth = '1997년 12월 12일'
print(f"{name}의 생년월일은 {birth}입니다.")

one = 1
two = 2
print(f"{one}집은 제네시스로 어느 정도 성공했고 {two}집은 엑소더스로 폭망했습니다.")
```

Adam의 생년월일은 1997년 12월 12일입니다.
1집은 제네시스로 어느 정도 성공했고 2집은 엑소더스로 폭망했습니다.

콘솔 입출력

- ❖ 콘솔 입력
  - ✓ input('메시지')
    - 문자열을 입력받는 내장함수
    - 매개변수로 하나의 문자열을 받는데 매개변수로 입력된 내용은 키보드의 입력을 알려주는 프롬프트
    - Enter를 누를 때까지 입력을 받아서 하나의 문자열로 리턴
    - 정수나 실수를 입력받고자 할 때는 문자열을 입력받고 난 후 수치형으로 변환
    - 여러 개의 데이터를 한 번에 입력받고자 하는 경우에는 구분 기호를 이용해서 입력받고 split 함수를 이용해서 분할해서 사용

```
name = input('이름:')
print(name)
age = int(input('나이:'))
print(age)
print(type(age))
```

```
이름:박문석
박문석
나이:34
34
<class 'int'>
```

콘솔 입출력

- 콘솔 입력
  - ✓ input('메시지')
    - 여러 개의 데이터를 한 번에 입력 받기

```
#공백으로 분할해서 입력
data = input("입력 (x y z) : ")

L = data.split()
x, y, z, = L[0], L[1], L[2]

print(x)
print(y)
print(z)

#,로 구분해서 입력
x, y, z = input("데이터 입력 (a,b,c) : ").split(',')

print(x)
print(y)
print(z)
```

```
입력 (x,y,z) : 1 2 3
1
2
3
데이터 입력 (a,b,c) : 1,2,3
1
2
3
```

콘솔 입출력

- 콘솔 입출력
  - ✓ 한겨레 신문사에서 kb라는 검색어를 입력해서 뉴스 검색을 하면 아래와 같은 URL을 생성

    http://search.hani.co.kr/Search?command=query&keyword=kb&media=news&submedia=&sort=d&period=all&datefrom=2000.01.01&dateto=2019.06.22&pageseq=0
  - ✓ 검색어 부분을 입력해서 URL을 완성하기

```
url = 'http://search.hani.co.kr/Search?command=query&keyword='
keyword = input('검색어를 입력하세요:')
url = url + keyword
url = url +
'&media=news&submedia=&sort=d&period=all&datefrom=2000.01.01&dateto=2019.06.22&pageseq=0'
print(url)
```

```
검색어를 입력하세요:kb
http://search.hani.co.kr/Search?command=query&keyword=kb&media=news&submedia=&sort=d&period=all&datefrom=2000.01.01&dateto=2019.06.22&pageseq=0
```

알고리즘

- Computational Thinking
  - ✓ 현실 세계의 문제를 분석하여 해결책을 찾는 과학적 사고법을 Computational Thinking 이라 하며 이렇게 설계한 해결책을 컴퓨터의 명령어로 작성하는 것이 컴퓨터 프로그래밍
  - ✓ 구성
    - 분해: 복잡한 문제를 작은 문제로 나눔
    - 패턴 인식: 문제 안에서 유사성을 발견
    - 추상화: 문제의 핵심에만 집중하고 부차적인 것은 제외
    - 알고리즘: 이렇게 정의한 문제를 해결하는 절차(일반화와 모델링은 여기에 포함)
  - ✓ 복잡한 문제를 해결하는 것은 어렵지만 작은 문제를 해결하는 것은 비교적 쉽기 때문에 작은 문제를 해결하다 보면 복잡한 문제를 해결하게 되는데 컴퓨터 공학에서 배우는 알고리즘은 대부분 정형화된 문제에 대해 검증된 해법을 제시하는 것
  - ✓ 현실에서 컴퓨터로 해결하려는 문제는 정형화된 문제가 아니라 비정형화된 문제가 더 많기 때문에 비정형화된 문제를 컴퓨터로 해결하는 과정, 즉 문제를 이해하고 분해, 패턴 인식, 추상화, 알고리즘 작성 까지가 Computational Thinking

연습문제

- 다음 중 변수를 만드는 방법으로 올바른 것을 고르세요.
  - a. int x = 10
  - b. 10 = x
  - C. X = 10
  - d. x <- 10
  - e. x(1
- 다음 중 변수 이름으로 사용할 수 없는 것을 모두 고르세요
  - a. 300
  - b. _53A
  - c. if
  - d. z
  - e. H-80
  - f. _hello3
  - g. 30seconds
  - h. 99%

연습문제

- 평균 구하기
  - ✓ 국어, 영어, 수학, 과학 점수가 입력받아서 평균 점수를 출력하는 프로그램을 만드세요(input에서 안내 문자열은 과목 이름을 출력)
  - ✓ 평균 점수를 출력할 때는 소수점 이하 자리는 버림(정수로 출력).

- 소음이 가장 심한 층수 출력하기
  - ✓ 국립환경과학원에서는 아파트에서 소음이 가장 심한 층수를 구하는 계산식을 발표했는데 소음이 가장 심한 층은 0.2467 * 도로와의 거리(m) + 4.159
  - ✓ 도로와의 거리를 입력받아서 소음이 가장 심한 층수가 출력되게 만드세요.
  - ✓ 층수를 출력할 때는 소수점 이하 자리는 버림(정수로 출력)

- 스킬 공격력 출력
  - ✓ L이라는 게임에서 "왜곡"이라는 스킬이 AP * 0.6 + 225의 피해를 줍니다.
  - ✓ 이 게임에서 AP(Ability Power, 주문력)는 마법 능력치를 뜻합니다.
  - ✓ AP를 입력받아서 스킬의 피해량이 출력되게 만드세요.