2024-1
기초 스터디

2. 조건문, 반복문

복습

- 자료형: 정수형, 실수형, 문자열
- 자료형 변환: **int()**, **float()**, **str()** 함수 사용
- 변수와 대입: **변수명 = 데이터**

복습

- 출력: **print(출력할 데이터)**
- 이스케이프 문자: **\n**, \", \\ 등

복습

- (문자열) 입력 : A = **input()**
- 정수 하나 입력 : A = **int(input())**
- 정수 여러 개 입력 : A, B, ... = **map(int, input().split())**

목차

조건문

- 무엇을 할 지 **‘조건’**에 따라 **‘선택’** 하는 것

- 피자 vs 치킨
- 만약 ‘바삭한 식감이 좋다’ → **치킨**
- 만약 ‘여러가지 맛을 한 입에 느끼고 싶다’ → **피자**

조건문

만약 ‘아보카도가 있다’

→ 우유를 6개 산다

아니라면(아보카도가 없다면)

→ 우유를 1개 산다

조건문 - if

```
1  if 조건식:
2      "조건식이 성립하는 경우 실행할 코드"
3      "반드시 같은 크기의 들여쓰기를 유지해야 합니다."
4
5  "이 코드는 조건식의 성립 여부에 상관없이 실행됩니다."
```

- 조건식에 맞는 경우 : 1 – 2 – 3 – 5
- 조건식에 안 맞는 경우 : 1 – 5

조건문 - if

- 들여쓰기 크기가 다르면 에러 발생 (반복문 포함)

```
1 if 조건식:
2     "조건식이 성립하는 경우 실행할 코드"
3       "반드시 같은 크기의 들여쓰기를 유지해야 합니다."
4
5 "이 코드는 조건식의 성립 여부에 상관없이 실행됩니다."
```

```
>>> %Run -c $EDITOR_CONTENT
 Traceback (most recent call last):
   File "<string>", line 3
     "반드시 같은 크기의 들여쓰기를 유지해야 합니다."
 IndentationError: unexpected indent
```

조건문 - 조건식

- 조건식에는 '참', '거짓' 을 갖는 **명제**로서 **bool형 데이터**가 들어간다.

- **bool형**에는 **True**, **False** 2가지 데이터가 있다.

조건문 - 조건식

- 조건식에는 아래와 같은 코드가 들어갈 수 있다.

1. bool 값	:	**True**	**False**
2. 비교 연산	:	1 **==** 2	2 **!=** 4
3. 포함 검사	:	1 **in** [1, 2]	3 **not in** [5, 6]

조건문 – True / False

- 조건식에 **True, False** 를 직접 사용할 수 있다.
- (키워드이므로 대소문자를 정확하게 지켜야 한다.)

```
1  if True:
2      "이 코드는 언제나 실행됩니다."
3      "이 코드도 언제나 실행됩니다."
4
5  "이 코드는 조건식의 성립 여부에 상관없이 실행됩니다."
```

```
1  if False:
2      "이 코드는 언제나 실행되지 않습니다."
3      "이 코드도요."
4
5  "이 코드는 조건식의 성립 여부에 상관없이 실행됩니다."
```

조건문 – True / False

- 조건식에는 **bool형**이 아닌 데이터도 넣을 수 있다.
- 그 경우 파이썬은 그 데이터를 **bool형으로 바꾸고** True / False를 판단한다.

- 정수형 데이터의 경우
  0이 아닌 숫자는 모두 **True**, 0은 **False**로 변환한다.

조건문 – True / False

- 직접 정수형 데이터를 넣어보며 확인해보자.

```
1  if 3:
2      print("print 3")
3  else:
4      print("print nothing")
```

```
>>> %Run -c $EDITOR_CONTENT
 print 3
```

```
1  if -5:
2      print("print -5")
3  else:
4      print("print nothing")
```

```
>>> %Run -c $EDITOR_CONTENT
 print -5
```

조건문 – 값 비교

- **비교연산자** 이용

- 같다 : ==
- 다르다 : !=
- 대소비교 : >, <, >=, <=

조건문 – 값 비교

- 비교연산자의 연산 결과는 **bool형** 데이터이다.

- 1 == 2 → **False**
- 1 != 2 → **True**
- 3 < 5 → **True**
- 6 < 8 < 7 → **False**

조건문 – 값 비교

- 비교연산자의 연산 결과는 **bool형** 데이터이다.

```
1  A = 3
2  B = 6
3
4  if A == B:
5      print("A, B는 같다.")
6  else:
7      print("A, B는 서로 다르다.")
```

```
>>> %Run -c $EDITOR_
  A, B는 서로 다르다.
```

조건문 – 포함 검사

- **in, not in** 키워드 사용

- 데이터 **in** [데이터 그룹] # 포함 되면 True
- 데이터 **not in** [데이터 그룹] # 포함 안되면 True

조건문 – 포함 검사

- **in, not in** 키워드 사용

- 3 **in** [3, 4, 5] → True
- 6 **not in** [1, 2, 3] → True
- “hi” **not in** [“hi”, “bye”] → False

조건문 – 포함 검사

- **in, not in** 키워드 사용

```
1  A = [1, 2, 3]
2  B = 2
3
4  if B in A:
5      print("YES")
6  else:
7      print("NO")
```

```
>>> %Run -c $EDI

  YES
```

조건문 – 논리 연산자

- 조건은 서로 연결될 수 있다.

- Ex) 빨갛고 **(and)** 둥글다면 → 사과이다.
  가볍거나 **(or)** 싸면 → 구매한다.

조건문 – 논리 연산자

- 조건식(bool형 데이터)을 대상으로 하는 연산자를 **'논리 연산자'** 라고 한다.
- 크게 **and**, **or**, **not** 3가지가 있다.

- cf) **xor 연산**을 수행하는 **^** 연산자도 있다.
  위 연산자들에 익숙해지면 나중에 공부해보자.

조건문 – 논리 연산자

- 조건식을 연결할 때는 **and, or** 연산자 사용

- 조건식1 **and** 조건식2
  → 두 조건식 **모두** True면 True / 아니면 False
- 조건식1 **or** 조건식2
  → 두 조건식 중 **하나라도** True면 True / 아니면 False

조건문 – 논리 연산자

- 조건식을 연결할 때는 **and, or** 연산자 사용

- 3 < 5 **and** 6 < 9 → True **and** True → True
- 3 < 5 **or** 2 > 5 → True **or** False → True
- 3 < 5 **and** 2 > 5 → True **and** False → False
- 3 > 5 **or** 2 > 5 → False **or** False → False

조건문 – 논리 연산자

- **or** 은 **왼쪽** 조건식부터 확인하다가 True를 만나면 나머지 조건은 확인하지 않는다.

- if 3 > 2 **or** 4/0 > 0:
  → 4/0은 연산이 불가능하므로 에러가 발생해야 한다.
  하지만 3 > 2 가 True 이므로 이후 조건은 확인하지 않아 에러가 발생하지 않는다.

조건문 – 논리 연산자

- **or** 은 **왼쪽** 조건식부터 확인하다가 True를 만나면 나머지 조건은 확인하지 않는다.

```
1  if 3 > 2 or 4/0 > 0:
2      print("wow")
```

```
>>> %Run -c $EDITOR_CONTENT
wow
```

```
1  if 4/0 > 0 or 3 > 2:
2      print("wow")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ZeroDivisionError: division by zero
```

조건문 – 논리 연산자

- 조건을 반전시킬 때는 **not** 키워드 사용

- **not** True → False
- **not** False → True

조건문 – 논리 연산자

- 조건을 반전시킬 때는 **not** 키워드 사용

- **not** 2 < 5 → **not** True → False
- **not** 3 == 4 → **not** False → True

조건문 – 논리 연산자

- 조건을 많이 연결할 때는 괄호로 묶자

```
1  if not 1 == 2 and 3 < 5 or 14:
2      print("YES")
3  else:
4      print("NO")
```

```
1  if not (1 == 2) and (3 < 5 or 14):
2      print("YES")
3  else:
4      print("NO")
```

조건문 - elif

- if 로 조건을 체크한 상황에서
  추가적으로 조건을 확인하고자 할 때 사용한다.
  → 만약 A 라면 a 하는데, **그렇지 않고** B **라면** b 한다.

- else if 를 줄여서 **elif** 라고 쓴다.

조건문 - elif

- **if** 다음에 사용하며, 여러 번 쓸 수도 있다.

```
1 if 조건식A:
2     "이 코드는 A일 때 실행됩니다."
3 elif 조건식B:
4     "이 코드는 A는 아니면서, B일 때 실행됩니다."
5
6 "이 코드는 A, B 상관없이 항상 실행됩니다."
```

조건문 – “if + if” vs “if + elif”

```
1  A = 10
2  if A > 5:
3      print("hello")
4  if A > 3:
5      print("world")
```

```
>>> %Run -c $EDITOR_CONTENT
 hello
 world
```

```
1  A = 10
2  if A > 5:
3      print("hello")
4  elif A > 3:
5      print("world")
```

```
>>> %Run -c $EDITOR_CONTENT
 hello
```

조건문 - else

- **if**, **elif** 조건에 모두 해당하지 않을 때 사용

```
1 if 조건식A:
2     "이 코드는 A일 때 실행됩니다."
3 elif 조건식B:
4     "이 코드는 A가 아니면서 B일 때 실행됩니다."
5 else:
6     "이 코드는 A도 아니고, B도 아닐 때 실행됩니다."
7
8 "이 코드는 항상 실행됩니다."
```

조건문 – else

- **else**는 **elif**가 없어도 사용할 수 있다.

```
1 A = 10
2 if A > 3:
3     print("hello")
4 else:
5     print("world")
```

```
>>> %Run -c $EDITOR_CONTENT
 hello
```

조건문 – 연습 문제

5 1330번 두 수 비교하기 성공

문제

두 정수 A와 B가 주어졌을 때, A와 B를 비교하는 프로그램을 작성하시오.

입력

첫째 줄에 A와 B가 주어진다. A와 B는 공백 한 칸으로 구분되어져 있다.

조건문 – 연습 문제

- 한번 먼저 풀어보세요

조건문 – 연습 문제

1. 두 정수를 입력 받기
2. 두 정수의 크기를 비교하여 결과 분류하기
3. 분류에 따라 다르게 출력하기

조건문 – 연습 문제

- 같이 풀어봅시다

조건문 – 연습 문제

- 정답코드

```
a, b = map(int, input().split())
if a > b:
    print(">")
elif a < b:
    print("<")
else:
    print("==")
```

If 표현식 (심화)

- if문 코드를 아래와 같이 쓸 수 있다

```
1 if A > 3:
2     print("hello")
```

→

```
if A > 3: print("hello")
```

- if문 내부 코드가 한 줄 일 때,
  들여쓰기 없이 한 줄로 사용할 수 있다.

If 표현식 (심화)

- if문으로 값을 대입할 때 아래와 같이 줄일 수 있다
- 이런 표현식을 **'if 표현식'** 이라고 한다.

```
1  if A > 3:
2      B = 3
3  else:
4      B = 5
```

→

```
6  B = 3 if A > 3 else 5
```

If 표현식 (심화)

- if표현식은 **'데이터'**를 나타내므로 print 문에 직접 넣을 수 있다.

```
1  A = 4
2  print(3 if A > 3 else 5)
```

```
>>> %Run -c $EDITOR_CONTENT
 3
```

If 표현식 (심화)

- (참고) 1330 문제를 if 표현식으로도 풀 수 있다.

```
1  a, b = map(int, input().split())
2  print(">" if a > b else "<" if a < b else "==")
```

- 이 내용은 심화 내용이니 이해가 안가도 괜찮습니다!
  이런 문법도 있구나~ 정도로 넘어가도 됩니다.

조건문 – 논리 연산자 연습 문제

2753번 윤년 성공

문제

연도가 주어졌을 때, 윤년이면 1, 아니면 0을 출력하는 프로그램을 작성하시오.

윤년은 연도가 4의 배수이면서, 100의 배수가 아닐 때 또는 400의 배수일 때이다.

예를 들어, 2012년은 4의 배수이면서 100의 배수가 아니라서 윤년이다. 1900년은 100의 배수이고 400의 배수는 아니기 때문에 윤년이 아니다. 하지만, 2000년은 400의 배수이기 때문에 윤년이다.

조건문 – 논리 연산자 연습 문제

- 같이 풀어봅시다.

조건문 – 논리 연산자 연습 문제

- 정답 코드

```
year = int(input())
if ((year % 4 == 0) and (year % 100 != 0)) or year % 400 == 0:
    print(1)
else:
    print(0)
```

반복문

- 특정 코드를 반복해서 실행하는 구문
- **규칙적**으로 반복되는 데이터를 다룰 때 자주 사용

ex) 1, 3, 5, 7, 9, ...
ex) “a”, “b”, “c”, ...

반복문

1. **조건**에 따라 반복 → while 문
2. **정해진 횟수**에 따라 반복 → for 문

반복문 - while

- 주어진 조건을 만족하는 동안 반복 (if문과 동일)

```
1 while 조건식:
2     "조건식이 성립하는 동안 반복되는 코드"
3 "반복이 끝나면 실행되는 코드"
```

- 실행 순서 : (1) – (2) – (1) – (2) - .... – **(1)** – (3)

반복문 – break, continue

- **Break** : 현재 반복하고 있는 반복문 종료
- **Continue** : 즉시 다음 차례 반복으로 건너뜀

- 보통 조건문과 함께 사용한다.

반복문 – break, continue

```
1  A = 0
2  while A < 5:
3      A += 1
4      print(A)
```

```
1
2
3
4
5
```

```
1  A = 0
2  while A < 5:
3      A += 1
4      if A < 3:
5          continue
6
7      print(A)
```

```
3
4
5
```

```
1  A = 0
2  while A < 5:
3      A += 1
4      if A == 3:
5          break
6
7      print(A)
```

```
1
2
```

반복문 – while 연습 문제

10952번 A+B - 5 성공

문제

두 정수 A와 B를 입력받은 다음, A+B를 출력하는 프로그램을 작성하시오.

입력

입력은 여러 개의 테스트 케이스로 이루어져 있다.

각 테스트 케이스는 한 줄로 이루어져 있으며, 각 줄에 A와 B가 주어진다. (0 < A, B < 10)

입력의 마지막에는 0 두 개가 들어온다.

반복문 – while 연습 문제

- 같이 풀어봅시다.

반복문 – while 연습 문제

- 정답 코드

```
while True:
    a, b = map(int, input().split())
    if a == 0 and b == 0:
        break

    print(a+b)
```

반복문 - for

- 데이터 그룹의 모든 데이터에 한번씩 접근

```
for 변수 in 데이터그룹:
    """
        각 반복마다 데이터그룹 안의 데이터가
        하나씩 차례대로 변수에 담깁니다.
    """
```

반복문 - for

- 데이터 그룹의 모든 데이터에 한번씩 접근

```
1  for number in [1, 2, 3]:
2      print(number)
```

- 반복할 때마다 number 변수에 1, 2, 3 이 차례대로 담긴다.

반복문 - for

- 일정 횟수 반복할 때는 **range 함수**를 이용

```
1  for i in range("반복할 횟수"):
2      "반복할 코드"
```

- range 함수는 **0부터 ‘ 반복할 횟수 - 1’ 까지의 정수**로 구성된 데이터 그룹을 만들어준다.

반복문 - for

```
1 for i in range(5):
2     print(i)
```

- range 함수는 **0부터 '4' 까지 정수**로 구성된 **5개 정수**의 데이터 그룹을 만들어준다.

  → (0, 1, 2, 3, 4) 로 구성된 숫자 데이터 그룹

반복문 - for

```
1 for i in range(5):
2     print(i)
```

- 총 **5**번 반복하고, 각 반복마다 i 변수에는 0, 1, 2, 3, 4 가 차례대로 담긴다.

반복문 – range

- **range()** 함수는 아래와 같이 사용할 수 있다.

- **range(a)** → 0부터 a-1까지 정수 데이터 그룹
- **range(a, b)** → a부터 b-1까지 정수 데이터 그룹
- **range(a, b, c)** → a부터 b-1까지 c씩 변화하는 정수 데이터 그룹

반복문 – range

```
1 for i in range(5):
2     print(i, end=' ')
```

```
>>> %Run -c $
 0 1 2 3 4
```

```
1 for i in range(2,5):
2     print(i, end=' ')
```

```
>>> %Run -
 2 3 4
```

```
1 for i in range(6, 3, -1):
2     print(i, end=' ')
```

```
>>> %Run
 6 5 4
```

반복문 – for 연습문제

10950번 A+B - 3 성공

문제

두 정수 A와 B를 입력받은 다음, A+B를 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다.

각 테스트 케이스는 한 줄로 이루어져 있으며, 각 줄에 A와 B가 주어진다. (0 < A, B < 10)

반복문 – for 연습문제

- 먼저 풀어봅시다.

반복문 – for 연습문제

- 같이 풀어봅시다.

반복문 – for 연습문제

- 정답 코드

```
1 T = int(input())
2 for _ in range(T):
3     a, b = map(int, input().split())
4     print(a+b)
```

반복문 - 중첩 반복문

- 반복문 안에서 반복문을 실행하는 것

- 반복문 안의 반복문이 끝나야 다음 반복으로 넘어간다.
- 중첩된 반복문 안에서 사용하는 **break**, **continue** 는 내부 반복문에 대해서만 동작한다.

반복문 - 중첩 반복문

```
1 for i in range(3):
2     for j in range(3):
3         print(i, j)
4         if j == 0:
5             break
```

```
>>> %Run -c $EDITOR_CONTENT
 0 0
 1 0
 2 0
```

반복문 – 중첩 반복문 연습문제

2438번 별 찍기 - 1 성공

입력

첫째 줄에 N(1 ≤ N ≤ 100)이 주어진다.

출력

첫째 줄부터 N번째 줄까지 차례대로 별을 출력한다.

예제 입력 1 복사

```
5
```

예제 출력 1 복사

```
*
**
***
****
*****
```

반복문 – 중첩 반복문 연습문제

- 먼저 풀어봅시다.

- 중첩 반복문을 사용한다면
  print(출력할 데이터, end=‘’) 형식으로 출력해주세요.

```
1  for i in range(5):
2      print(i, end='')
```

```
>>> %Run -c $ED
 01234
```

반복문 – 중첩 반복문 연습문제

- 같이 풀어봅시다.

반복문 – 중첩 반복문 연습문제

- 정답 코드

```
N = int(input())
for i in range(N):
    for j in range(i+1):
        print("*", end='')

    print()
```

시간 제한 & 빠른 입력

- 많은 알고리즘 문제들은 입력을 반복해서 받는다.
- 이때 반복 횟수가 많으면 100만까지도 간다.

- input 함수는 느린 함수라서 입력 횟수가 많아지면 제한시간을 넘어간다.
  → 빠른 입력을 위해 **외부 함수**를 사용한다.

시간 제한 & 빠른 입력

- 빠른 입력 함수는 **sys.stdin.readline()** 이다.
- 일반적으로 아래와 같이 사용한다.

```
1  import sys
2  input = sys.stdin.readline
3
4  A = input().rstrip()
```

빠른 입력

```
1 import sys
2 input = sys.stdin.readline
3
4 A = input().rstrip()
```

1. **readline** 이라는 빠른 입력 함수를 이용한다.
이 함수는 **sys** 라는 외부 파일에 저장되어 있으므로,
해당 외부 파일을 import (불러오기) 해준다.

빠른 입력

```
1  import sys
2  input = sys.stdin.readline
3
4  A = input().rstrip()
```

2. 원래 사용하던 input 입력 함수를
   새로운 입력함수 readline 으로 덮어쓴다.

빠른 입력

```
1  import sys
2  input = sys.stdin.readline
3
4  A = input().rstrip()
```

3. input() 이라고 입력 기능을 실행하면, readline 함수가 대신 입력 기능을 수행한다.

빠른 입력

```
1 import sys
2 input = sys.stdin.readline
3
4 A = input().rstrip()
```

- readline 함수는 '엔터'를 칠 때까지 입력한 모든 문자를 **'엔터(\n)' 까지 포함한 문자열로** 돌려준다.

빠른 입력

```
1  import sys
2  input = sys.stdin.readline
3
4  A = input().rstrip()
```

- 마지막 '엔터(\n)'를 제외하기 위해 **rstrip()** 메서드를 사용한다.
- 입력 받은 문자열을 그대로 사용하는 경우에 필요하다.

시간을 줄이는 팁

- 파이썬에서 문제를 제출할 때, Python3 대신 **PyPy3**으로 제출하면 더 빠른 경우가 많다.

빠른 입력 - 연습문제

15552번 빠른 A+B 성공

입력

첫 줄에 테스트케이스의 개수 T가 주어진다. T는 최대 1,000,000이다. 다음 T줄에는 각각 두 정수 A와 B가 주어진다. A와 B는 1 이상, 1,000 이하이다.

출력

각 테스트케이스마다 A+B를 한 줄에 하나씩 순서대로 출력한다.

빠른 입력 - 연습문제

- 정답 코드

```python
import sys
input = sys.stdin.readline

T = int(input())
for _ in range(T):
    a, b = map(int, input().split())
    print(a + b)
```

출석 체크 & 우수 스터디원

- 출석체크 조건
  → 강의 중에 같이 푼 문제 모두 풀기

- 우수 스터디원 선정 기준
  1. 출석을 많이 한 사람
  2. 연습 문제를 많이 푼 사람

우수 스터디원 상품

- 하이아크에서 주는 상품 + 스터디장이 주는 상품

이번주 연습 문제

- 9498 시험 성적
  14681 사분면 고르기
  2884 알람 시계
  2480 주사위 세개
  2739 구구단
  10872 팩토리얼
  2439 별 찍기 - 2
  25304 영수증

연습 문제 풀이 팁

- **2739 구구단**

  2 * 1 = 2 를 출력할 때 print() 함수는 **,** 로 나열한 값을 공백으로 구분해서 출력해준다는 걸 이용해봅시다.

  print(2, '*', 1, '=', 2) 이렇게 출력하면 간단해요!

  문자열을 미리 공부해보고 싶다면 **f-string** 을 공부해보세요.
  f-string을 이용하면 더 깔끔하게 풀 수 있습니다.

연습 문제 풀이 팁

- **2439 별 찍기 - 2**

  힌트: 공백 문자의 개수와 '*' 의 개수 합이 N으로 일정합니다!

  이 문제를 풀었다면 https://www.acmicpc.net/workbook/view/20
  별 찍기 시리즈의 다른 브론즈 문제도 풀어보세요!

  반복문을 완전 정복할 수 있습니다.

연습 문제 풀이 팁

- **25304 영수증**

  합이 일치하면 Yes , 일치하지 않으면 No 출력
  이렇게 조건에 따라 Yes, No 를 출력하는 유형이 꽤 많아요
  이런 유형을 풀 때 **if 표현식 (심화)** 개념을 한번 사용해보세요.

  물건 가격의 합을 sum_cost, 영수증 가격을 receipt_cost 라고 하면

  print(“Yes” if sum_cost == receipt_cost else “No”)

  이렇게 한 줄로 출력 코드를 작성할 수 있어요.