파이썬 심화

파이썬 기초 학습 점검

파이썬 자료형

변수

- 변수
  - 프로그램에서 값을 저장하기 위한 저장 공간을 지칭
  - 변수는 알파벳 대소문자, 숫자, _를 이용하여 이름을 지정
  - 같은 이름의 변수는 정해진 범위내에서 같은 저장공간을 사용
- 변수 이름
  - 숫자가 앞에 와서는 안 됨
  - 예약어(keyword)가 사용되서는 안 됨
  - 올바른 변수 이름 : _ abc _def c0o r123
  - 올바르지 않은 변수 이름 : 3x74 um%b for

변수의 활용

- <변수> = <값>
  - <변수>에 <값>을 저장한다.
  - 예) abc = 3 → abc란 변수에 3이란 값을 저장한다.
- <변수> 수식
  - <변수>에 저장된 값을 사용한다.
  - 예) print(abc) → abc에 저장된 값을 출력한다.
- <변수1>, <변수2> = <값1>, <값2>
  - <변수1>에는 <값1>을 <변수2>에는 <값2>를 저장한다.
  - 예) a, b = 3, 4 → a 변수에는 3이란 값을, b 변수에는 4란 값을 저장한다.

자료형의 종류

- 기본 자료형
  - 숫자형 : 1, 10, 3.14, …
  - 문자열 : “한글”, “Korea”, …
  - 불(bool)형 : True, False
- 나열 자료형
  - 리스트 : [1, 2, 3, 4], [“한글”, “한국”], …
  - 튜플 : (1, 2, 3, 4), (“한글”, “한국”), …
  - 집합 : { 1, 2, 3, 4 }, { “한글”, “한국” }, …
  - 딕셔너리 : { “one” : 1, “two” : 2 }, { 12 : “열둘”, 17 : “열일곱” }, …

연산자

- 파이썬의 연산자는 총 7가지가 존재하며 두개의 항이 필요한 이항연산자
- 연산자는 자료형에 따라 역할이 다르며, 용법에 주의가 필요
- 연산자 우선순위가 있으므로 ( ) 을 통해서 연산자 우선순위 조절

연산자	설명	우선순위
+	덧셈	낮음
-	뺄셈	낮음
*	곱셈	중간
/	나눗셈	중간
**	거듭제곱	높음
//	정수 나누기	중간
%	정수 나머지	중간

사용자 입력과 출력문

사용자 입력문 input()

- 사용자 입력문은 input(<프롬프트>)을 사용
  - <프롬프트>은 생략가능하며, 생략한 경우 <프롬프트>가 출력되지 않음
  - Input()의 결과물은 문자열로 한줄의 입력을 받음
- 한줄로 입력하기 : input()
- 입력된 문자열을 공백으로 분리 : input().split()
- 여러 개의 같은 자료형 입력 : list(map(int, input().split()))
- 정해진 개수의 같은 자료형 입력 : a, b = map(int, input().split())

출력문

- 파이썬 3.x에서는 print(…) 형태로 사용
  - print(<문자열>) → <문자열> 출력
  - print(<문자열1> <문자열2>) → <문자열1><문자열2> 출력(두 문자열 사이에 공백이 없는 것에 유의)
  - print(<문자열1>, <문자열2>) → <문자열1> <문자열2> 출력(두 문자열 사이에 공백이 존재)
  - print(<문자열1>, end=<문자열2>) → <문자열1>을 출력하고 줄바꿈을 하지 않고 대신 <문자열2>를 출력 (여러줄의 print() 문장으로 한줄로 출력을 위해 사용)

문자열 형식화란?

- 문자열의 형식을 지정하여 문자열로 만드는 방법
  - 숫자의 자릿수가 필요한 경우
  - 다양한 형식의 표현이 필요한 경우
  - 위치 맞춤이 필요한 경우
- 문자열 형식 문법
  - <형식 문자열> % (인수 리스트)

문자열 형식

- %을 이스케이프 문자로 사용하는 형식 언어 사용 (C/C++ 스타일)

형식	설명
%s	문자열 (모든 인수에 사용 가능)
%d	정수
%f	실수
%o	8진수 표현
%x	16진수 표현
%c	문자
%%	% 자체를 표현하기 위해 사용

% 타입 형식 지정

- %s, %d 등에서 %와 s 또는 %와 d 사이에 형식을 지정할 수 있다
  - a, b가 정수인 경우 a.b 형태로 지정
  - 예) %5.2s → 총 공간은 5칸을 차지하고, 문자열의 2글자만 표시
  - 예) %4.2f → 소수점 두째자리까지 표현하고, 전체 길이가 4 이하인 경우엔 4글자로 표현, 그렇지 않으면 해당 숫자 표현
  - 예) %-4d → 글자가 왼쪽부터 시작하도록 표시 (-가 붙으면 왼쪽 맞춤)
  - 예) %04d → 숫자를 표현할때 공백대신 0으로 채움

문자열 형식

- format() 함수를 통한 문자열 형식
  - <문자열 형식>.format(인수 리스트)
  - <문자열 형식>에 {<숫자>} 형태로 인수의 위치 지정
  - 인수에 <인수명>을 지정하여 {<인수명>}으로 지정

문자열 형식

- 출력 형식 지정 { <인수 위치 또는 인수명>:<정렬 및 형식>}
  - < : 왼쪽 맞춤
  - > : 오른쪽 맞춤
  - ^ : 가운데 맞춤
  - 숫자 : 글자수
  - 소수f : 소수점을 표현

조건문

조건식

- 조건식은 참, 거짓을 판별할 수 있는 연산을 의미하며, 결과는 불 자료형으로 반환
- 조건 연산자의 종류
  - 비교 연산자 : 두 값을 비교해서 참 거짓을 반환
  - 논리 연산자 : 논리(and, or, not) 을 통해 연산
  - in 연산자 : 나열형 자료에 해당 자료가 있는지 검사

비교 연산자

- x < y : x가 y보다 작은 경우 참
- x <= y : x가 y보다 작거나 같은 경우 참
- x > y : x가 y보다 큰 경우 참
- x >= y : x가 y보다 크거나 같은 경우 참
- x == y : x가 y와 같은 경우 참
- x != y : x가 y와 같지 않은 경우 참

논리 연산자

- and : 논리곱 연산자로 두개의 항목이 모두 참인 경우에만 참
  - True and True → True, True and False → False, False and False → False
- or : 논리합 연산자로 두개의 항목 중 참이 있으면 참
  - True or True → True, True or False → True, False or False → False
- not : 논리 부정 연산자로 현재의 논리값의 부정
  - not True → False, not False → True
- 예) 3 > 4 and 4 > 3 → 3 > 4 는 False, 4 > 3 은 True, False and True 는 False

in 연산자

- 나열형 자료형에 해당 원소가 있으면 참을 반환
- x in <나열형 자료> : <나열형 자료>에 x 값이 존재하면 참
- x not in <나열형 자료> : <나열형 자료>에 x 값이 존재하지 않으면 참
    - 예) 3 in [1, 2, 3, 4] → True
    - 예) 3 in [1, 2, 4, 5] → False
    - 예) 3 not in [1, 2, 3, 4] → False
    - 예) 3 not in [1, 2, 4, 5] → True

if 조건문

- if 문

```
if <조건식> :
    <조건식이 참인 경우 수행문>  # 같은 블록의 <수행문>은 같은 수준 들여쓰기 필수
```

- if - else 문

```
if <조건식> :
    <조건식이 참인 경우 수행문>  # 같은 블록의 <수행문>은 같은 수준 들여쓰기 필수
else :
    <조건식이 거짓인 경우 수행문>
```

if 조건문

- if - elif - else 문

```
if <조건식1> :
    <조건식1이 참인 경우 수행문> # 같은 블록의 <수행문>은 같은 수준 들여쓰기 필수
elif <조건식2>
    <조건식1이 거짓이고 조건식2가 참인 경우 수행문>
...
else :
    <앞의 모든 조건식이 거짓인 경우 수행문>
```

반복문

while 문

- 전형적인 반복문의 형태
- while 문 구조
  - <조건식> : <조건식>이 참인경우 <반복 수행문> 수행 거짓인 경우 반복문을 빠져나옴
  - <반복 수행문> : 반복 실행될 수행문으로 동일한 형태의 들여쓰기가 된 여러줄의 수행문으로 구성

```
while <조건식>:
    <반복  수행문>
    …
…
```

for 문

- 형식이 있는 반복문으로 정형적인 반복문을 사용
- for 문 구조
  - <반복자> : for문을 반복할 때, 변화하는 변수
  - <나열형 자료> : 리스트, 튜플 등 나열형 자료
  - <반복 함수> : range(…)와 같은 반복하여 값을 내는 함수

```
for <반복자> in <나열형 자료>/<반복 함수>:
    <반복 수행문>
    …
…
```

반복 함수 range()

- for문에서는 반복 함수를 사용 가능
- range(<값들>)
  - 1개의 값 n만 주어진 경우 : 0부터 n-1까지 차례대로 반복
  - 2개의 값 a, b가 주어진 경우 : a부터 b-1까지 차례대로 반복
  - 3개의 값 a, b, c가 주어진 경우
    - c가 양수인 경우 : a부터 b-1까지 c간격으로 반복자값 반복
    - c가 음수인 경우 : a부터 b+1까지 c간격으로 반복자값 반복

제어문

- 제어문은 반복문 구조에서 반복 조건외에 반복문의 흐름을 조절할 때 사용
  - continue : 남아있는 반복 수행문을 수행하지 않고 맨 마지막으로 간다.
  - break : 현재 반복문을 빠져나간다.
  - continue문을 사용할 경우 조건식에 주의가 필요

continue 제어문

- 일반적으로 if문과 함께 continue문 사용

```
while(<조건문>):
    <수행문1>
    if <조건문>:
        continue
    <수행문2>
    # 반복문의 끝
<수행문3>
```

반복문의 끝으로 진행

break 제어문

- 일반적으로 if문과 함께 사용

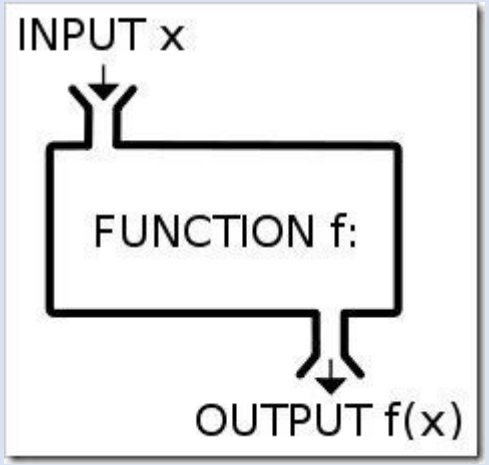
```
while(<조건문>):
    <수행문1>
    if <조건문>:
        break
    <수행문2>
    # 반복문의 끝
<수행문3>
```

반복문을 빠져나옴

함수

함수

- 반복수행할 수행문들을 이름을 지정하여 실행할 수 있는 구조문
  - 함수이름 : 함수에 들어있는 수행문을 함수이름으로 대표해서 호출
  - 매개변수 : 함수를 호출할 때, 함수의 수행문이 사용할 수 있도록 전달
  - 반환값 : 함수가 종료될 때, 함수를 호출한 곳으로 값을 전달



def 문

- 함수를 정의하기 위해서는 def라는 예약어를 사용
- def 문 구조
  - <함수이름> :함수를 호출하기 위해서 사용되는 대표 이름
  - <매개변수 리스트> : 함수를 호출할 때 전달되는 값들에 대응할 변수 리스트
  - <수행문> : 함수가 호출되는 실행되는 수행문으로 같은 수준의 들여쓰기가 필수

```
def <함수이름>(<매개변수 리스트>):
    <수행문>
    ...
...
```

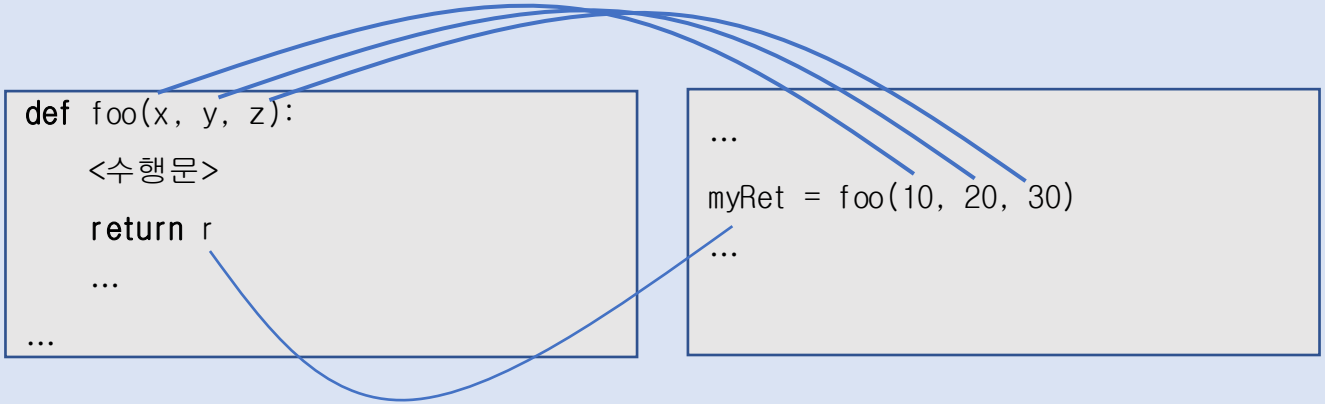
반환값을 전달하기 위한 return

- 함수의 결과값을 함수를 호출한 곳으로 전달할 수 있는데, 이 때 return이란 예약어를 사용
  - return 뒤에 전달할 값을 나열하면 해당 값을 호출한 곳으로 전달
  - return만 쓰인 경우, 바로 함수를 종료하고 호출한 곳으로 복귀

```
def <함수이름>(<매개변수 리스트>):
    <수행문>
    return <반환값>
    …
…
```

매개변수와 반환 값

- 함수를 호출한 곳과 함수를 정의한 곳의 매개변수는 1:1로 매칭
- 반환값은 함수를 호출한 곳으로 반환



반복함수

- for문에서 반복을 계속할 수 있도록 제작된 함수 (예 : range() 함수)
- 반환값을 전달할 때, return 대신 yield 예약어를 사용
  - 값을 반환하지만 함수는 현재 상태에서 보류상태
  - 다음 함수가 호출되면 계속 실행

```
def myrange(n):
    for k in range(n):
        yield k

for x in myrange(4):
    print(x)
```

```
0
1
2
3
```