제어문

while

제어문 (Control Statement)

- 반복 구문 - while

  while 조건문 :

  조건이 참(True)인 동안 수행할 구문1
  조건이 참(True)인 동안 수행할 구문2
  …

  – 조건문을 검사하여 그 결과가 참(True)인 동안 while 블록의 구문들을 순서대로 반복하여 실행한다.
  – 조건문의 결과가 거짓(False)이 되면 while 블록은 더 이상 실행되지 않고 빠져 나온다.

제어문 (Control Statement)

- 반복 구문 - while

  while 조건문 :

  조건이 참(True)인 동안 수행할 구문1
  조건이 참(True)인 동안 수행할 구문2
  . . .

  – 정수 1부터 5까지 화면에 출력하시오.

```
1  a = 1
2
3  while a < 6:
4      print(a)
5      a += 1
```

```
1
2
3
4
5
```

제어문 (Control Statement)

- 반복 구문 - while / else

```
while 조건문 :
        조건이 참(True)인 동안 수행할 구문1
        ...
else :
        조건이 거짓(False)인 경우 수행할 구문1
        ...
```

  - 조건문을 검사하여 그 결과가 참(True)인 동안 while 블록의 구문들을 순서대로 반복하여 실행한다.
  - 조건문의 결과가 거짓(False)이 되면 else 블록 구문들을 순서대로 한번 실행하고 빠져 나온다.

제어문 (Control Statement)

- 반복 구문 - while / else
  - 사용자로부터 정수를 계속 입력 받아서, 0이 아니면 그 수를 화면에 출력하고, 0이면 “종료합니다.”를 출력한 뒤 프로그램을 종료하시오.

```
1  a = int(input("정수를 입력하세요: "))
2
3  while a != 0:
4      print(a)
5      a = int(input("정수를 입력하세요: "))
6  else:
7      print("종료합니다.")
```

```
정수를 입력하세요: 5
5
정수를 입력하세요: -1
-1
정수를 입력하세요: 0
종료합니다.
```

제어문 (Control Statement)

- 반복을 강제로 종료하기 - **break**
  - 명령어 break는 자신이 속해 있는 반복 구문을 빠져 나온다.
  - 조건문이 현재 참이더라도, break를 만나면 반복 구문은 종료된다.

```
1  a = 1
2
3  while a < 6:
4      print(a)
5      break
6      a += 1
```

```
1
```

```
1  a = int(input("정수를 입력하세요: "))
2
3  while a != 0:
4      print(a)
5      a = int(input("정수를 입력하세요: "))
6      break
7  else:
8      print("종료합니다.")
```

```
정수를 입력하세요: 5
5
정수를 입력하세요: -1
```

제어문 (Control Statement)

- 반복을 강제로 종료하기 - break
  - 반복을 수행하는 도중, 원래 조건문 외의 반복을 종료할 다른 추가 조건을 검사할 필요가 있을 때 사용한다.

```
1  a = input("문자열을 입력하세요 (종료하려면 quit): ")
2
3  while True:
4      print("입력한 내용은", a)
5      if a == "quit":
6          break
7      a = input("문자열을 입력하세요 (종료하려면 quit): ")
8
9  print("반복 구문이 끝났습니다.")
```

```
문자열을 입력하세요 (종료하려면 quit): 안녕하세요?
입력한 내용은 안녕하세요?
문자열을 입력하세요 (종료하려면 quit): 반갑습니다
입력한 내용은 반갑습니다
문자열을 입력하세요 (종료하려면 quit): python
입력한 내용은 python
문자열을 입력하세요 (종료하려면 quit): quit
입력한 내용은 quit
반복 구문이 끝났습니다.
```

제어문 (Control Statement)

- 반복의 처음 위치로 돌아가기 - **continue**
  - 명령어 continue는 자신이 속해 있는 반복 구문의 맨 처음 위치로 돌아간다.
  - continue를 만나면 반복 구문의 맨 위의 조건문으로 돌아가서 조건을 다시 검사한다.

```
1  while True:
2      a = int(input("양의 정수를 입력하세요: "))
3      if a <= 0:
4          continue
5      print("입력한 내용은", a)
```

```
양의 정수를 입력하세요: 5
입력한 내용은 5
양의 정수를 입력하세요: -1
양의 정수를 입력하세요: 2
입력한 내용은 2

양의 정수를 입력하세요:
```

제어문 (Control Statement)

- 반복의 처음 위치로 돌아가기 - continue
  - 반복을 수행하는 도중, 다음 구문들을 무시하고 반복을 계속해야 할 필요가 있을 때 사용한다.
  - 대부분의 경우, if / else 구문만으로 동일한 동작을 처리할 수 있다.

```
1  a = 0
2
3  while a <= 10:
4      a += 1
5      if a % 2 == 1:
6          continue
7      print(a)
```

```
2
4
6
8
10
```

※ continue 없이
동일한 동작을 수행한 예

```
1  a = 0
2
3  while a <= 10:
4      a += 1
5      if a % 2 == 0:
6          print(a)
```

```
2
4
6
8
10
```

제어문 (Control Statement)

- 반복 구문을 사용할 때 주의할 점
  - 의도하지 않는 한, 무한 루프가 되지 않도록 블록 내에서 반복을 종료할 수 있는 구문이 반드시 존재해야 한다.
  - 변수 값 조절을 통해 조건문의 결과를 거짓으로 만들거나, break 구문으로 특정 상황에서 빠져 나오도록 한다.

```
1  a = 1
2
3  while True:
4      print(a)
5      a += 1
6      if a > 10:
7          break
```

```
1  v = int(input("정수를 입력하세요: "))
2
3  while v != 1:
4      print(v)
5      if v % 2 == 0:
6          v /= 2
7      else:
8          v += 1
```

```
정수를 입력하세요: 3
3
4
2.0
```

제어문 (Control Statement)

- 반복 구문의 여러 가지 예

```
a = 0
b = {1, 2, 3}

while a < 3:
    print("a가 3보다 작은 동안 출력됩니다.")
    if a > 1:
        if type(b) == set:
            if "1" in b:
                print("a가 1보다 크고 b가 집합이며 문자열 "1"이 b에 있는 경우 출력됩니다.")
            else:
                print("a가 1보다 크고 b가 집합이며 문자열 "1"이 b에 없는 경우 출력됩니다.")
    a += 1

print("while 구문이 끝났고, 이어서 그 다음 줄부터 실행됩니다.")
print(a)
```

```
a가 3보다 작은 동안 출력됩니다.
a가 3보다 작은 동안 출력됩니다.
a가 3보다 작은 동안 출력됩니다.
a가 1보다 크고 b가 집합이며 문자열 "1"이 b에 없는 경우 출력됩니다.
while 구문이 끝났고, 이어서 그 다음 줄부터 실행됩니다.
3
```

제어문 (Control Statement)

- 반복 구문의 여러 가지 예

```
1  a = 3
2
3  while a:
4      print("값이", a, "입니다.")
5      a -= 1
```

```
값이 3 입니다.
값이 2 입니다.
값이 1 입니다.
```

```
1  while False:
2      print("while 구문의 조건이 참인 동안 출력됩니다.")
3  else:
4      print("while 구문의 조건이 거짓인 경우 출력됩니다.")
```

```
while 구문의 조건이 거짓인 경우 출력됩니다.
```

```
1  while "False":
2      print("while 구문의 조건이 참인 동안 출력됩니다.")
3      break
4  else:
5      print("while 구문의 조건이 거짓인 경우 출력됩니다.")
```

```
while 구문의 조건이 참인 동안 출력됩니다.
```

제어문 (Control Statement)

- 반복 구문의 여러 가지 예

```
1  a = [1, 2, 3]
2
3  while a:
4      print("리스트에 항목이 들어 있는 동안 출력됩니다.")
5      print("항목을 하나씩 꺼내어 출력합니다.", a.pop(0))
6  else:
7      print("리스트가 비어 있으면 출력됩니다.")
```

```
리스트에 항목이 들어 있는 동안 출력됩니다.
항목을 하나씩 꺼내어 출력합니다. 1
리스트에 항목이 들어 있는 동안 출력됩니다.
항목을 하나씩 꺼내어 출력합니다. 2
리스트에 항목이 들어 있는 동안 출력됩니다.
항목을 하나씩 꺼내어 출력합니다. 3
리스트가 비어 있으면 출력됩니다.
```

제어문 (Control Statement)

- 반복 구문의 여러 가지 예

```
1  check = True
2
3  while check:
4      v = int(input("양수를 입력하세요: "))
5
6      if v > 0:
7          check = False
8      else:
9          print("입력한 값이 양수가 아닙니다. 다시 입력하세요.")
```

```
양수를 입력하세요: 0
입력한 값이 양수가 아닙니다. 다시 입력하세요.
양수를 입력하세요: -7
입력한 값이 양수가 아닙니다. 다시 입력하세요.

양수를 입력하세요:
```

제어문 (Control Statement)

- 반복 구문의 중첩 (nested statement)
  - 구문은 필요에 따라서 얼마든지 중첩되어(nested) 사용할 수 있다.
  - 중첩된 하위 블록은 상위 블록에 종속(dependent)된다.

```
1  a = 1
2  b = 0
3
4  while a < 3:
5      print("a가 3보다 작은 동안 출력됩니다. 지금 a는", a, "입니다.")
6      while b < 2:
7          print("그 상황에서 b가 2보다 작은 동안 출력됩니다. 지금 b는", b, "입니다.")
8          b += 1
9      b = 0
10     a += 1
```

```
a가 3보다 작은 동안 출력됩니다. 지금 a는 1 입니다.
그 상황에서 b가 2보다 작은 동안 출력됩니다. 지금 b는 0 입니다.
그 상황에서 b가 2보다 작은 동안 출력됩니다. 지금 b는 1 입니다.
a가 3보다 작은 동안 출력됩니다. 지금 a는 2 입니다.
그 상황에서 b가 2보다 작은 동안 출력됩니다. 지금 b는 0 입니다.
그 상황에서 b가 2보다 작은 동안 출력됩니다. 지금 b는 1 입니다.
```

제어문 (Control Statement)

- 반복 구문의 중첩 (nested statement)

```
a = 1
b = 2
c = 3

while a >= 0:
    print(a)
    while b == "2":
        print("second")
    else:
        while True:
            if c == 3:
                print(c)
                break
            c += 1
        print(b)
    a -= 1
```

```
1
3
2
0
3
2
```