모든 계열을 위한 기초 교육

지능형로봇프로그램밍의 이해

5주: 반복문 – while

한양대학교 ERICA 소프트웨어학부
조상욱 교수
swcho3@hanyang.ac.kr

목차

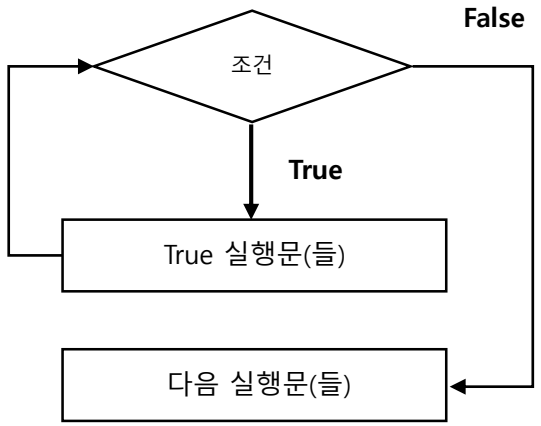
1. 반복문의 이해_while

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

1.1 while문 구조

- 흐름을 제어하는 문의 한 종류로 특정 조건을 만족하는 동안 반복을 계속 수행하는 구조이다. 그림은 while문의 흐름도를 나타낸다. while문내 조건의 참 거짓에 따라 실행을 반복한다. 실행문을 포함하고 있는 몸체는 단일 및 복수개의 문장들로 구성될 수 있다. 조건을 평가 후 거짓일 경우, while문 밖에 있는 실행문으로 제어가 넘겨지게 된다.

while 조건: True일 때 실행할 문장(들) 다음 실행할 문장	"조건"이 True이면 'True 일 때 실행할 문장(들)'을 반복적으로 실행한 후, '다음 실행할 문장'을 실행한다. "조건"이 False이면 바로 '다음 실행할 문장'을 실행한다.



1. 반복문의 이해_while

1.1 while문 구조

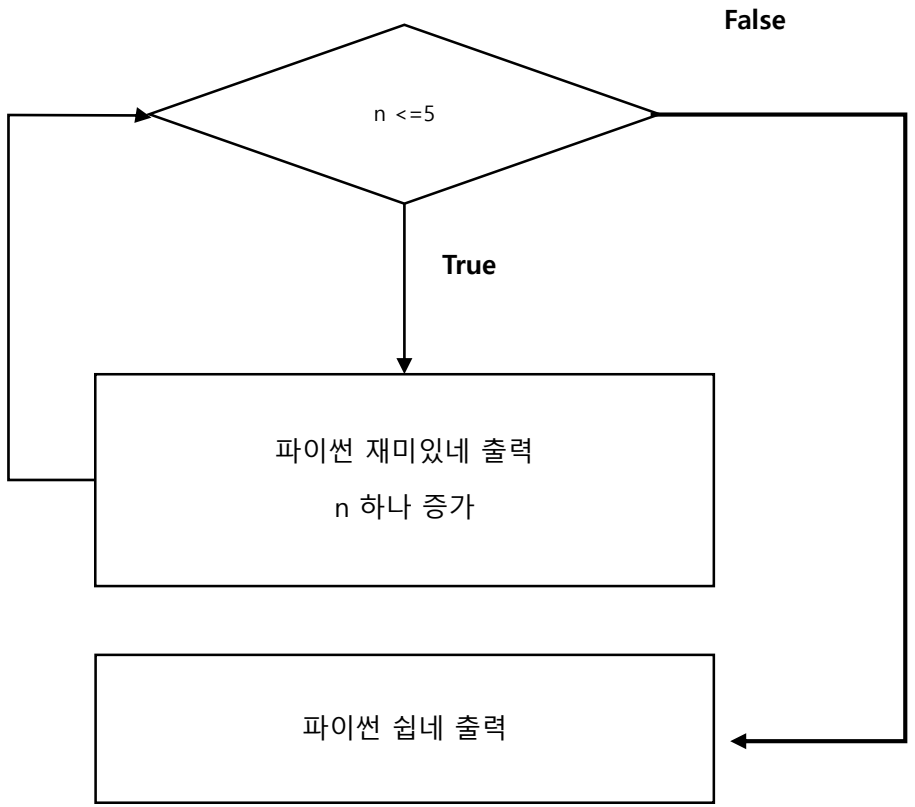
- 다음 구조는 조건이 참이면 True인 실행문을 수행하고나서 While문 밖의 실행문을 수행하게 된다. 간단한 예제 코드와 더불어 확인할 수 있다. 아래 예제에서 반복문을 이용하여 출력하는 문을 5번씩이나 작성할 필요가 없다. 단지 반복 출력할 횟수만 알려주면 된다. 변수 n은 1로 초기화 되어 있다. 조건문 n이 5이하인지를 검사한다. 만약 참이면 “파이썬 재미있네” 문자열을 출력하고 n을 하나 증가시킨다. 즉 n은 프로그램 수행 시 반복처리가 필요한 실행문을 몇 회 반복했는지 기억하기 위한 변수이다. 이를 반복횟수변수라고 한다. 다음 반복횟수변수가 계속 증가하다가 5될때까지 실행문을 수행하고나서, 6이되면 반복을 종료하고 실행문 “파이썬 쉽네” 문을 출력한다. 여기서 주의할 사항은 while문 안에 실행되는 문은 안쪽으로 들여쓰기이다. 이는 가독성과 연관 되어있다. 아래 예제의 “n=n+1”이 while문과 같이 왼쪽 정렬로 구성되었다면 조건문은 항상 참으로 판단되기 때문에 무한 반복을 실행하게 될 것이다.

1. 반복문의 이해_while

1.1 while문 구조

```
print(“파이썬 재미있네”)
print(“파이썬 재미있네”)
print(“파이썬 재미있네”)
print(“파이썬 재미있네”)
print(“파이썬 재미있네”)
Print(“파이썬 쉽네”)
```

```
n = 1
while n <= 5:
    print("파이썬 재미있네")
    n = n + 1
print("파이썬 쉽네")
```



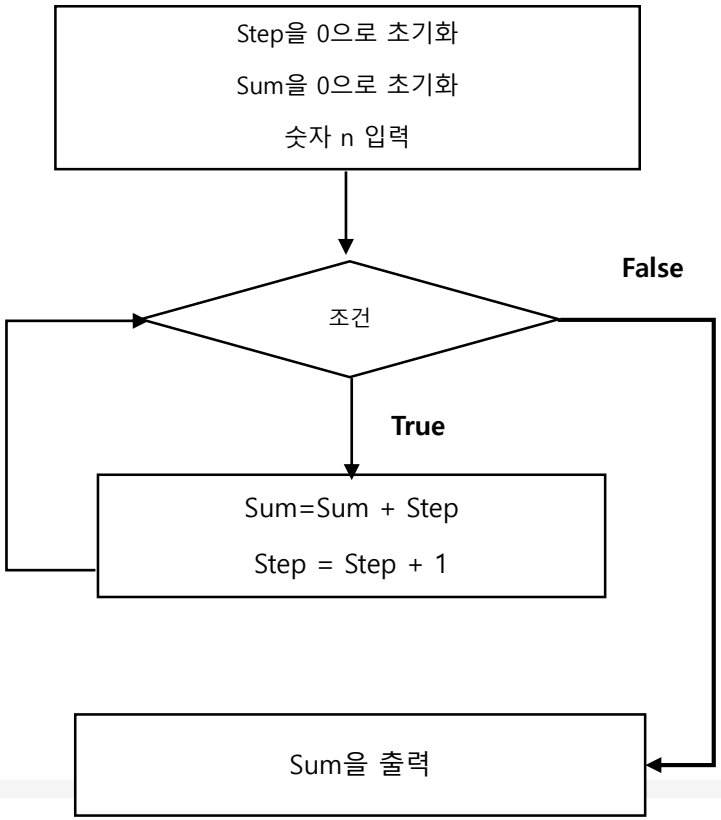
1. 반복문의 이해_while

1.2 while문의 적용하기1

- 다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자
- 1부터 시작하여 사용자가 입력한 값까지 모든 정수를 합한 결과를 출력하는 프로그램을 작성해보자
  - ➢ 입력 값으로 20을 넣어서 합한 결과를 출력해보자
  - ➢ 입력 값으로 30을 넣어서 합한 결과를 출력해보자

■ 문제 해결 알고리즘

- ➢ 하나의 정수를 입력이 받아 1부터 입력 받은 정수까지 합한 결과를 출력하는 프로그램이다.
- ➢ 입력 받은 숫자를 초과할 때까지 조건을 비교하여 반복한다.
  - ① 합계를 위한 변수, 단계를 나타내는 변수 초기화한다.
  - ② 하나의 정수 n을 입력한다.
  - ③ 조건문을 이용하여 정수 n을 초과하는지 비교한다.
  - ④ n보다 작거나 같을 동안 1부터 n까지 단계를 증가하면서 합계를 구한다.
  - ⑤ n보다 초과한 경우 반복문을 빠져나온다.
  - ⑥ 합계를 출력한다.



1. 반복문의 이해_while

■ 프로그램 작성

```
step = 0
sum = 0
n = int(input('Enter the number : '))
while(step <= n):
    sum = sum + step
    step = step + 1
print(sum)
```

■ 테스트

```
Enter the number : 20
210
>>>
Enter the number : 30
465
```

1. 반복문의 이해_while

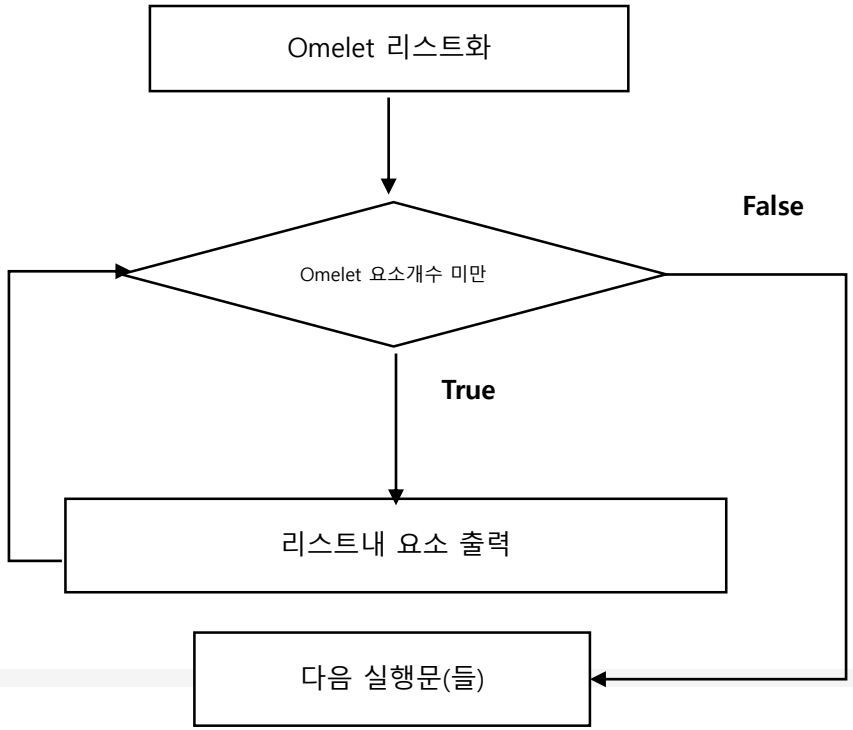
한양대학교 ERICA
Education Research Industry Cluster @ Ansan

1.3 while문의 적용하기2

- 다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자
- 오믈렛 재료의 일부분으로 구성된 list가 다음과 같이 omelet=['egg', 'meat', 'onion', 'carrot'] 구성되어 있다.
  - ➢ Omelet에 있는 모든 재료를 반복문을 이용하여 한 번씩 출력해본다.

■ 문제 해결 알고리즘

- ➢ 요구조건으로 입력값 없이 리스트로 초기화 되어있다. 단지 출력 값만 있다. 이 문제를 해결하기위한 문제의 흐름도는 다음과 같다.



1. 반복문의 이해_while

■ 프로그램 작성

```
omelet=['Egg', 'Meat', 'Onion', Carrot']
index = 0
while(index < len(omelet)):
    ingredient= omelet[index]
    print(ingredient)
    index = index + 1
```

■ 테스트

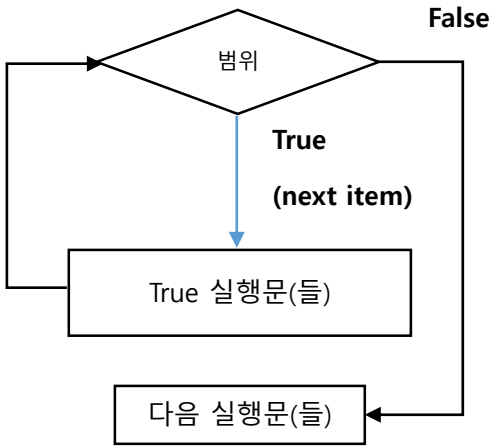
```
>>>
Egg
Meat
Onion
carrot
```

2. 반복문의 이해_for

1.1 for문 구조

•while문은 반복의 횟수가 명확하게 제시되지 않은 경우에 유용하다. 반면에for문은 반복의 횟수나 범위를 미리 알고 있을 경우 사용하는 것이 바람직하다. 시작 값과 종료 값이 주어지면 그 범위 내에서 반복을 수행한다. 그러므로 while문에서 사용한 반복을 벗어나기위한 증감 변수는 for문에서는 사용할 필요가 없다. 그림은 for문의 흐름도를 나타낸다. 시퀀스에 포함된 각각의 요소를 통해 반복된다. 범위지정으로는 range()함수, 시퀀스(sequence), 문자열 등을 사용할 수 있다.

For item in 범위: True 실행문(들)	범위에 포함된 각각의 연속되는 요소가 item에 저장되고 각 요소에 대해 True 실행문(들)이 한번씩 수행



2. 반복문의 이해_for

1.2 범위함수: range()

Range()함수는 반복을 단순하게 만들 수 있도록 도와주는 함수이다. 즉 특정범위를 지정하여 범위 내에서 반복을 수행하게 한다.

range([start], stop [, step]):	start부터 step만큼씩(증가/감소)하면서 stop전까지 범위를 지정하는 함수

- start값은 포함하지만 stop값은 포함되지 않음
  - range(1,9,2)의 의미는 1,3,5,7의 요소를 가진다.
- Range()함수를 사용할 때는 다음과 같은 사항을 주의해야한다.
  - start와 stop에는 정수(int형)만 쓸 수 있음
  - start와 step은 생략될 수 있음
    - range(4)의 의미는 0,1,2,3의 요소를 가짐.
    - range(4,8)의 의미는 4,5,6,7의 요소를 가짐.
  - step은 0이 될 수 없음
    - step이 생략되는 경우는 step 값은 1이라고 간주함.
    - range(4,8)에서처럼 4부터 1씩 증가하여 7까지 요소를 가짐
  - stop은 생략될 수 없음

2. 반복문의 이해_for

1.3 for문 적용하기1

- **range()를 활용한 for문**
- **range() 범위에 포함된 각각의 요소들에 대해 반복을 수행한다**

for var in range(num): True 실행문(들)	range범위에 포함된 요소 각각이 var에 저장되고 각각의 요소에 대해서 True 실행문(들)을 반복적으로 수행

2. 반복문의 이해_for

- 다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자. “You are so good student” 라는 문자열을 사용자로부터 입력 받은 횟수만큼 출력해본다.
- 문제 해결 알고리즘
- 요구조건으로 입력 값은 문자열을 반복적으로 출력하기 위한 횟수이고, 출력은 그 횟수만큼 반복해서 출력하는 것이다. 이 문제를 해결하기위한 문제의 흐름도는 다음과 같다.

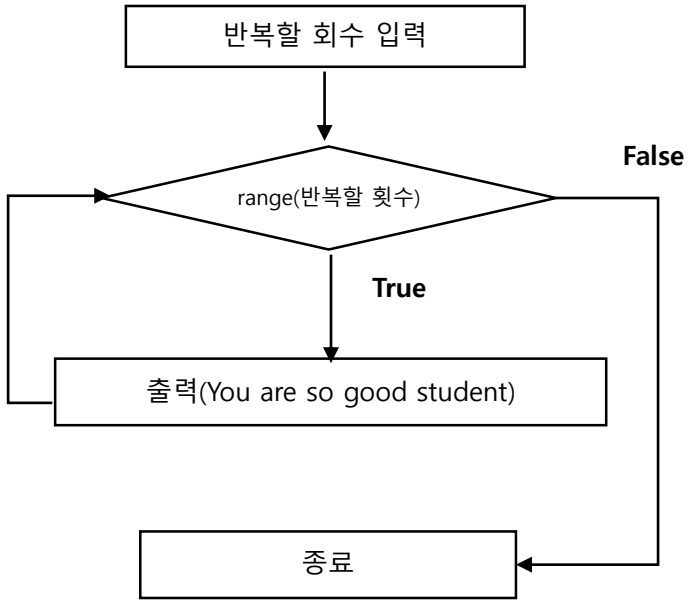
① 반복할 횟수 num을 입력한다.

② 범위내에서 정수 num을 초과하는지 비교한다.

③ num-1 횟수만큼 문자열을 출력한다.

④ 반복 횟수가 num일 경우 반복문을 빠져나온다.

⑤ 종료한다.



2. 반복문의 이해_for

■ 프로그램 작성

```
num=int(input("enter the repeat number :")
for i in range (num):
        print("You are so good student")
```

■ 테스트

```
>>>
enter the repeat number :4
You are so good student
You are so good student
You are so good student
You are so good student
>>>
```

2. 반복문의 이해_for

1.4 for문 적용하기2

- 문자열을 활용한 for문
- 문자열의 모든 문자에 대해 실행문을 반복 수행한다.

for chr in 문자열: True 실행문(들)	문자열 범위에 포함된 문자 각각에 대해 chr에 저장되고 각 문자에 대해서 True 실행문(들)을 반복적으로 수행

2. 반복문의 이해_for

- 다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자. 사용자로부터 n개의 문자열을 입력 받아 1번째 알파벳부터 n번째 알파벳까지 알파벳 순으로 출력해 보자. 예를 들어 문자열 'abcdef' 를 입력 받았으면 다음과 같이 출력 결과를 출력해본다.
- *출력결과*

1번째 알파벳: a

2번째 알파벳: b

3번째 알파벳: c

4번째 알파벳: d

5번째 알파벳: e

6번째 알파벳: f

2. 반복문의 이해_for

■ 프로그램 작성

➢ 요구조건으로 입력은 n개의 문자열이다. 입력된 문자열을 대상으로 개별적인 문자에 접근하여 각각 출력하는 프로그램이다. 이 문제를 해결하기위한 문제의 흐름도는 다음과 같다.

① 문자열에 포함된 개별적인 위치변수를 초기화한다.

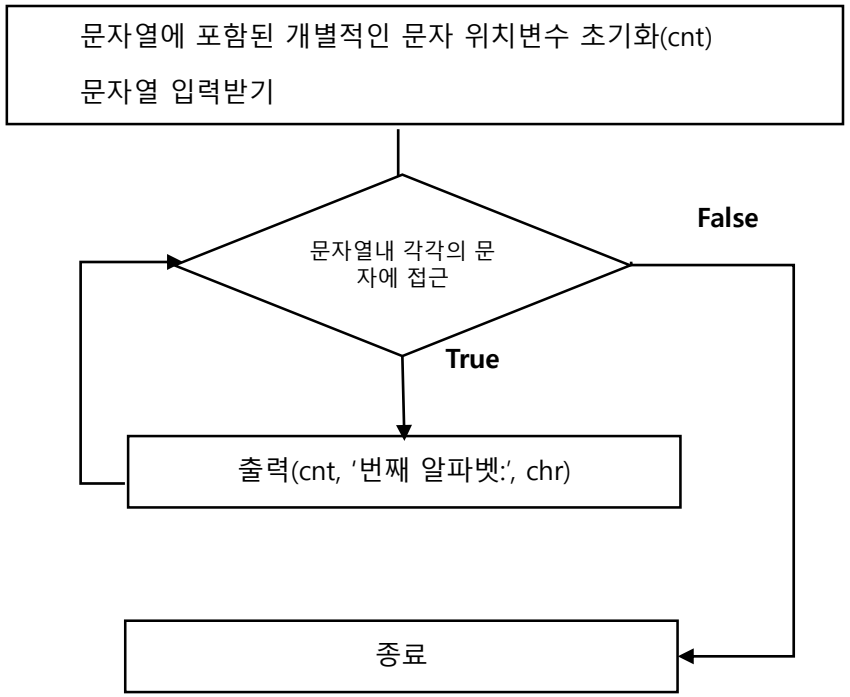
② 문자열을 입력한다.

③ 문자열 범위에 포함된 문자 각각에 대해 접근한다.

④ 반복횟수와 개별적인 문자를 출력한다.

⑤ 문자열 범위를 벗어나면 반복문을 빠져나온다.

⑥ 종료한다.



2. 반복문의 이해_for

■ 프로그램 작성

```
cnt = 1
chr = input("enter the string :")
for i in chr:
        print(cnt, '번째 알파벳:', chr[cnt-1])  #print(cnt, '번째 알파벳:', i)
        cnt=cnt+1
```

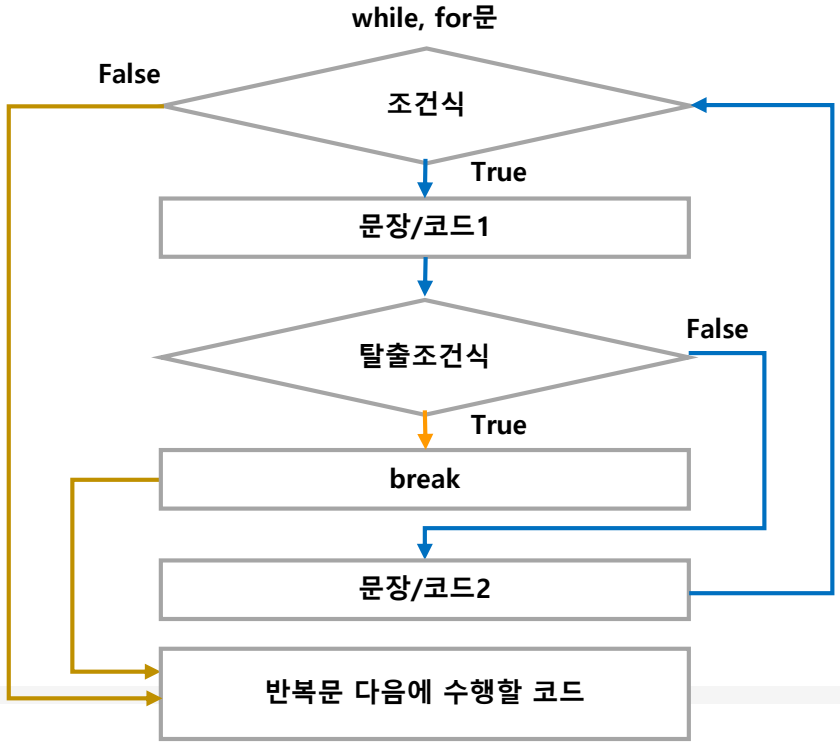
■ 테스트

```
enter the string :abcdef
1 번째 알파벳: a
2 번째 알파벳: b
3 번째 알파벳: c
4 번째 알파벳: d
5 번째 알파벳: e
6 번째 알파벳: f
>>>
```

3. break문 이해

1.1 break문 구조

- break문은 반복문 안에서 사용하면 반복문을 강제종료 시킬 수 있다. 다음 그림에서와 같이 반복 명령문에 추가적인 제어를 위해 사용되었다. 반복문 안에서 탈출 조건식을 판단하여 참이면 반복문의 제어를 반복문 다음 코드로 넘기게 된다. 반면에 탈출 조건식이 거짓인 경우는 반복문안에 있는 문장을 계속 수행한다. 주로 무한 반복문안에서 사용자가 원하는 시점에서 강제적으로 반복을 벗어날 때 사용 한다.



3. break문 이해

- **무한반복**

조건식이 항상 참인 경우를 사용해 보면 반복문은 끝나지 않고 반복문 안의 명령문을 계속 수행

- **while문의 break적용**

while 조건:
  True 실행문(들)
  break조건:
  break
	조건에 만족하는 동안 True 실행문(들)을 반복적으로 수행하고, break조건에 만족하면 반복을 중단하고, while문을 빠져나감

3. break문 이해

■ 다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자. 사용자가 입력한 n이 홀수인지 짝수인지를 판별하는 결과를 출력해본다.

- -1값을 입력하면 “stop”을 출력하며 프로그램이 종료된다

3. break문 이해

■ 프로그램 작성

```
while(True):
    num=int(input("input the number :"))
    if num==-1:
            print("stop")
            break
    if num%2==0:
            print(num, " is even")
    else:
            print(num, " is odd")
```

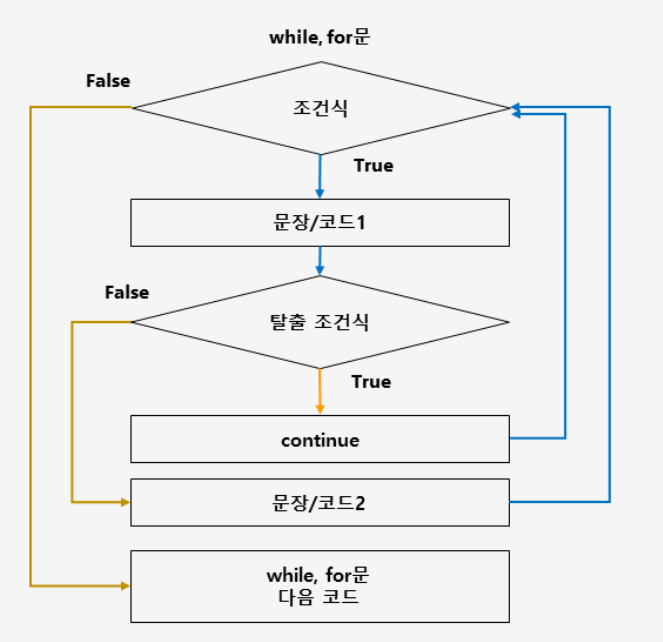
■ 테스트

```
input the number :-1
stop
>>>
```

3. contiune문 이해

1.1 continue문 구조

반복문안에서 continue문을 사용할 수 있다. Continue문을 만나면 현재 반복을 종료하고 무조건 조건식으로 점프한다. if문과 같이 사용되는 경우가 많은데 if조건식에 해당하는 조건이 참이면 스킵된다. 즉 continue는 반복문안에서 남은 부분은 건너 띄고 반복문 처음으로 돌아간다. 반면에 break는 반복을 중단하고 반복문 밖으로 빠져나간다.



3. continue문 이해

■ 무한반복

조건식이 항상 참인 경우를 사용해 보면 반복문은 끝나지 않고 반복문 안의 명령문을 계속 수행

■ for문의 continue적용

for i in 범위: True 실행문(들) (continue) 조건: continue	범위를 만족하는 동안 True 실행문(들)을 반복적으로 수행하고, continue조건을 만족하면 반복을 중단하고, continue문을 빠져나가 다시 for 반복문으로 돌아감.

3. continue문 이해

■ **다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자. 1부터 n까지의 수를 대상으로 3의 배수인 수만 출력해본다.**

- **For 문과 continue문을 이용하여 작성**
- **continue문을 사용하지 않은 코드와 비교**

■ **문제 해결 알고리즘**

① 정수를 입력 받는다.

② 정수 n 범위까지 만족하는 동안 진행한다.

① 조건문을 이용하여 0과 같은 지를 비교한다.

① 조건을 만족 할 경우, '3의 배수'를 출력하며 반복문을 계속한다.

② 조건을 만족하지 않을 경우 건너뛰고 반복문을 계속한다.

③ 정수 n 범위를 벗어나면 종료한다.

3. continue문 이해

■ 프로그램 작성

```
n=int(input('숫자를 입력하세요:'))
for i in range(1,n+1):
    if i % 3 !=0:
        continue
    print(i, ': 3의 배수')
```

```
n=int(input('숫자를 입력하세요:'))
for i in range(1,n+1):
    if i % 3==0:
        print(I, ': 3의 배수')
```

이와 같이 break 와 continue 문을 사용하면 상황에 따라 유용하게 사용할 수 있다. 그러나 부적절하게 많이 사용할 경우 디버깅이 어려워지거나 가독성이 떨어질 수 있다. 이런 점을 주의하여 사용하길 바란다.

■ 테스트

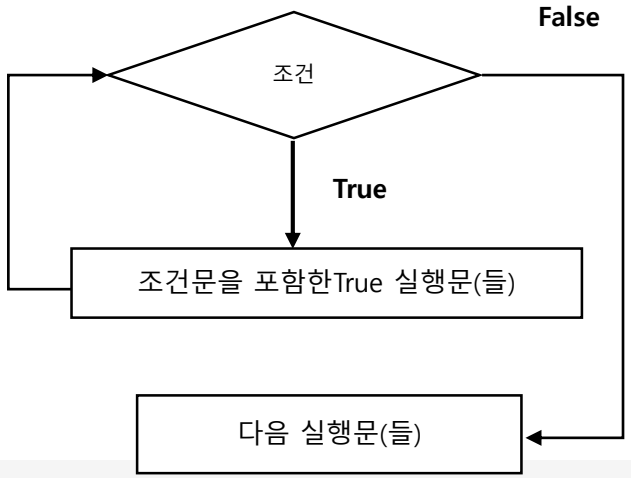
```
숫자를 입력하세요:10
3 : 3의 배수
6 : 3의 배수
9 : 3의 배수
```

4. 조건문과 반복문의 결합

1.1 조건문과 반복문의 결합 구조

단순히 조건문과 반복문을 개별적으로 사용하기 보다는 혼용하여 사용하면 프로그램을 좀 더 효율적으로 작성할 수 있다. 예를 들어 무한 루프가 실행되는 코드에서 반복을 멈추고 빠져나오려면 특정 조건을 부여하여 중단시킬 수 있다. 이때 특정 조건을 나타내는 조건문이 포함되는 것이다. 즉 반복문안에서 특정 조건에 따라 반복 실행되는 조건문을 결합하여 활용한다.

while item in range(num):
  if 조건
  True 실행문(들)
	Range 범위 내의 각 item에 대해 조건을 판단한 후, 만족하면 True 실행문(들)을 수행



4. 조건문과 반복문의 결합

- 다음 예를 들어 1부터 50미만의 정수가 있을 때 반복문과 조건문을 결합하여 짝수만 출력하는 프로그램을 작성하여 보자. 먼저 요구사항을 생각해보면 1부터 50미만의 범위를 지정해야 할 수 있다. 그리고 짝수인지를 확인하다. 짝수를 확인하는 조건은 2로 나누었을 때 0인경우 짝수가 된다. 다음 코드처럼 간단하게 작성될 수 있다

```
for num in range(1,50):
        if num%2==0
    print(num, ' is even' )
```

4. 조건문과 반복문의 결합

1.2 조건문과 반복문의 결합 적용하기

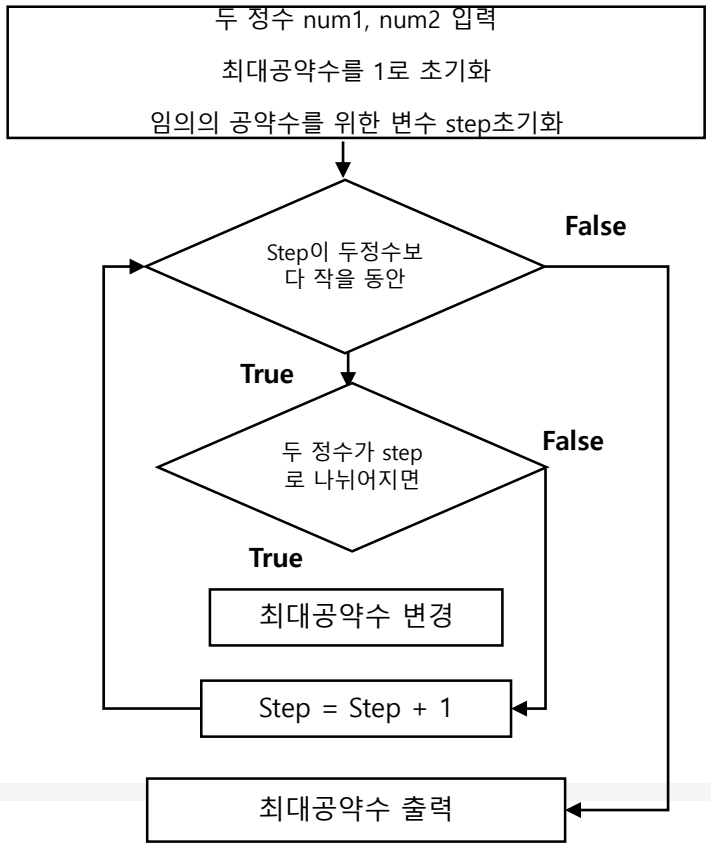
- 다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자
- 두 정수를 입력 받아 최대 공약수를 구하는 프로그램을 작성해보자.
  - 두 정수 num1, num2의 공약수 1은 최대공약수가 아닐 수 있다.
  - 두 정수의 최대공약수가 num1, num2의 공약수인지를 num1, num2보다 작을 때까지 검사한다.

4. 조건문과 반복문의 결합

1.2 조건문과 반복문의 결합 적용하기

■ 문제 해결 알고리즘

➢ 요구사항으로 두 정수를 입력이 받아 최대공약수를 출력하는 프로그램이다. 이 문제를 해결하기위한 알고리즘을 설정해 보도록 한다.



4. 조건문과 반복문의 결합

■ 프로그램 작성

```
gcd = 1
step = 2
num1 = int(input('Enter the number : '))
num2 = int(input('Enter the number : '))

while(step <= num1) and (step <=num2):
    if (num1%step==0) and (num2%step==0):
            gcd = step
    step = step + 1
print("최대공약수 : ", gcd)
```

■ 테스트

```
Enter the number : 6
Enter the number : 4
최대공약수 :  2
>>>
```

4. 중첩반복문

2.1 중첩반복문 구조

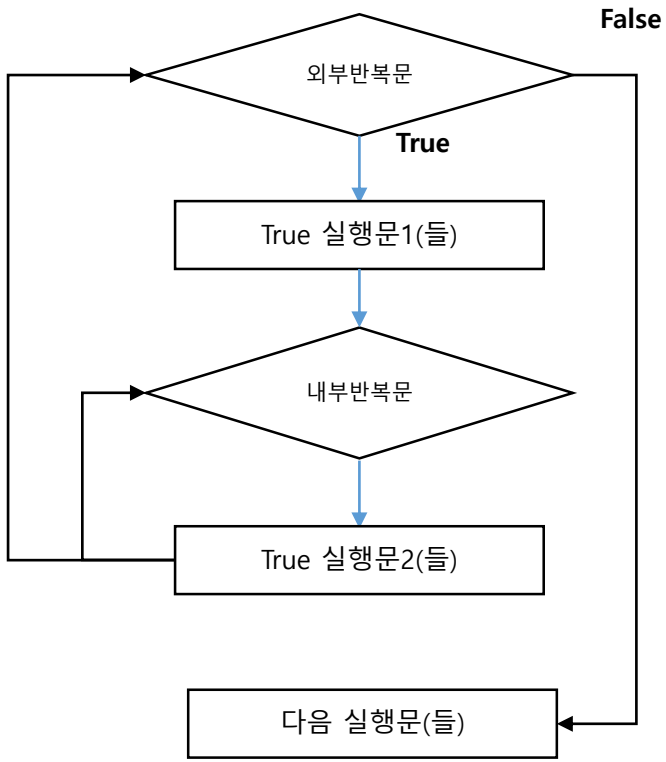
조건문과 마찬가지로 반복문 또한 중첩 가능하다. 중첩반복문은 한 개의 for문 또는 while문안에 한 개 이상의 내부 반복문을 가지는 것이다. 즉 첫번째 한 개의 for문과 while문은 외부 반복문으로 취급될 수 있으며, 외부반복문으로 해결되지 않는 문제 발생 시 내부 반복문을 사용하여 해결한다.

for item1 in sequence1:
  True 실행문1
  for item2 in sequence2:
  True 실행문2
	외부 for 시퀀스1에 포함된 요소 각각이 item1에 저장되고 각각의 요소에 대해서 True 실행문1과 내부 for문 반복적으로 수행, 내부 for문 시퀀스2내의 item2각각에 대해서 True 실행문2을 반복적으로 수행

while 조건1:
  True 실행문1
  while 조건2:
  True 실행문2
	조건1을 만족할 경우 True 실행문1과 다음 내부 while문으로 진입하여 조건2를 만족할 경우 True 실행문2를 반복적으로 수행

4. 중첩반복문

2.1 중첩반복문 구조



4. 중첩반복문

2.2 중첩반복문의 적용하기

■ 다음 조건에 맞는 코드를 작성하여 프로그램을 만들어보자. 구구단을 출력하기위해 중첩반복문을 이용하여 다음과 같이 출력 결과를 출력해본다.

```
2단:  2  4  6  8 10 12 14 16 18
3단:  3  6  9 12 15 18 21 24 27
4단:  4  8 12 16 20 24 28 32 36
5단:  5 10 15 20 25 30 35 40 45
6단:  6 12 18 24 30 36 42 48 54
7단:  7 14 21 28 35 42 49 56 63
8단:  8 16 24 32 40 48 56 64 72
9단:  9 18 27 36 45 54 63 72 81
```

■ 문제 해결 알고리즘

- 요구조건으로 2단에서 9단까지의 구구단을 출력하는 것이다.
- 단의 출력값은 'end' 키워드를 사용하여 가로줄로 각 단을 출력한다.
- 단의 결과값의 출력 형식은 'print('%3d' %(i*j)'를 사용하여 각 자리수가 3자리로 서식을 고정시킨다.

4. 중첩반복문

■ 프로그램 작성

```
for i in range(2,10):
    print(str(i)+'단:',end=' ')
    for j in range(1,10):
        print('%3d' %(i*j), end=' ')
    print()
```

■ 테스트

```
2단:   2   4   6   8  10  12  14  16  18
3단:   3   6   9  12  15  18  21  24  27
4단:   4   8  12  16  20  24  28  32  36
5단:   5  10  15  20  25  30  35  40  45
6단:   6  12  18  24  30  36  42  48  54
7단:   7  14  21  28  35  42  49  56  63
8단:   8  16  24  32  40  48  56  64  72
9단:   9  18  27  36  45  54  63  72  81
>>>
```

연습문제

1. 다음 코드를 실행했을 때의 출력 결과로 알맞은 것은 무엇인가? ①

<코드>

```
step = 2
total = 5
while (step <= 5):
        total += step
        step += 1
print(total)
```

① 19 ② 16 ③ 13 ④ 11

퀴즈

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

2. 다음 range() 함수에 대한 설명으로 틀린 것은 무엇인가? 2

<다음>

range([start], stop, [step])

① start와 step은 생략할 수 있음

② start가 생략되는 경우에 start 값은 항상 1부터 시작함

③ step는 0일 수 없음

④ stop은 생략할 수 없음

퀴즈

3. 반복문을 강제로 탈출할 때 사용하는 제어문은 무엇인가?

① for문

② pass문

③ with~as문

④ break문

퀴즈

4. 주어진 조건이 만족하면 반복문의 시작 부분으로 이동하는 제어문은 무엇인가?

① with~as문

② continue문

③ if~else문

④ pass문

퀴즈

- 5. 단을 입력 받아 해당 단을 출력하는 구구단 프로그램을 실행 결과와 같이 작성하시오.
- <실행결과>
- 출력할 단 입력: 3
- 3 × 1 = 3
- 3 × 2 = 6
- 3 × 3 = 9
- 3 × 4 = 12
- 3 × 5 = 15
- 3 × 6 = 18
- 3 × 7 = 21
- 3 × 8 = 24

감사합니다.