반복문

while

같은 코드를 while문을 이용하여 여러 번 실행할 수 있음
while 루프 문은 주어진 조건이 참일 경우 목표 문장을 반복적으로 실행

In [1]:

```
x = 0
j = 3
while j>0:
    x += 5
    j -= 1
print(x)
```

15

while문 문법

while < test statement >:
  < test 결과가 True일 경우 해야 할 일 >

if문과 비슷한 구조를 지님

- < test statement >가 True일 경우 해야 할 일을 수행하고 다시 while문으로 돌아가 가 참인지 체크
- < test statement >가 False일 경우 while문 종료
- if 문과 마찬가지로 들여쓰기 규칙을 잘 지켜야 한다.

while문 내의 코드를 잘 작성하여 무한 루프(infinite loop)가 발생하지 않도록 주의

무한 루프의 예

j의 값에 변화가 없기 때문에 무한히 "Hello world"를 출력

In [ ]:

```
j = 3
while j>0:
    print("Hello world")
```

while문 내에서 j의 값을 변화하도록 수정하여 무한루프 문제 해결

- j의 값이 매 loop마다 1씩 감소하며 세 번 "Hello world"를 출력

In [2]:

```
j = 3
while j>0:
    j -= 1
    print("Hello world")
```

```
Hello world
Hello world
Hello world
```

- 반복문 다시 살펴 보기

In [3]:

```
x = 0
j = 3
while j>0:
    x += 5
    j -= 1
    print(x)
```

```
5
10
15
```

- while문 검사에 사용하는 변수 j는 while문 바깥에서 미리 초기화
- while문 내에서 j의 값을 변화시키면서 무한 루프를 방지

예제

In [4]:

```
x = 3
ans = 0
itersLeft = x
while (itersLeft != 0):
    ans = ans + x
    itersLeft = itersLeft - 1
print(str(x) + '*' + str(x) + ' = ' + str(ans))
```

```
3*3 = 9
```

반복문 내의 반복문

In [5]:

```
x = 3
counter = 0
while x>0:
    y = 4
    x -= 1
    while y>0:
        counter += 1
        y -= 1
print(counter)
```

```
12
```

break를 통한 while 벗어나기

break를 이용하면 현재의 loop를 벗어날 수 있음

- 일반적으로 if와 함께 사용되며 if의 조건이 True이면 현재 loop 종료

In [6]:

```
count = 0
while True:
    count += 1
    if count >= 10:
        break
print(count)
```

```
10
```

while문을 통해 여러 번 입력 받기

숫자를 다섯 개 입력 받아 총 합을 출력

In [8]:

```
counter = 5
summation = 0
while counter > 0:
    num = input("Input any number : ")
    summation += float(num)
    counter -= 1
print(summation)
```

```
Input any number : 1
Input any number : 2
Input any number : 3
Input any number : 4
Input any number : 5
15.0
```

다음 코드는 입력된 문자열을 출력하는 단순 프로그램이며 q를 입력하여 빠져나올 수 있다.

In [4]:

```
while True:
    inpt = input("If type q, you can quit : ")
    if inpt == 'q':
        break
    else: print("You just typed :", inpt)
```

```
If type q, you can quit : a
You just typed : a
If type q, you can quit : v
You just typed : v
If type q, you can quit : q
```

while/else

Python에 있는 독특한 문법
while문의 반복 조건이 False일 경우 else문을 실행

- while문에 아예 진입하지 않았거나 정상적으로 종료될 경우 else문 실행

break를 통해 while문을 강제로 빠져나오면 else문은 실행되지 않음

예제 : x의 값을 바꿔가며 실험해 보자.

In [9]:

```
x = 8
while x%2 == 0:
    x = x/2
    if x == 1:
        print("x is a power of 2")
        break
else:
    print("x is not a power of 2")
```

```
x is a power of 2
```

반복문의 일반적 형태

반복문을 설계하여야 할 때 다음 사항들을 생각해 보자.

- counter로 사용할 변수 설정
- counter 변수를 반복문 바깥에서 초기화
- 반복문이 언제 끝나야 하는지 결정하는 end-test 설정
- 반복문 내에서 실행할 코드를 작성
  - counter를 변화시키는 작업 포함
- 반복문이 종료되었을 때 이를 무엇을 할지 결정

제곱근을 찾는 단순 반복문 프로그램

다음의 프로그램은 x가 제곱수일 경우 정수인 제곱근을 찾아준다.

- 단, 한 가지 잘못된 점이 있다. 찾아보자.

In [11]:

```
x = 16
ans = 0
while ans*ans <= x:
    ans = ans + 1
print(ans)
```

5

다음은 제곱근을 찾는 단순 반복문 프로그램이다.

- 왼쪽의 프로그램이 어떻게 작동하는지 생각해 보자.
- 이 프로그램을 어떻게 하면 향상시킬 수 있는지 생각해 보자.

In [12]:

```
x = 18
ans = 0
increment = 1
while increment >= 0.0001:
    ans += increment
    if ans**2 == x:
        break
    elif ans**2>x:
        ans -= increment
        increment = increment/10.0
print(ans)
```

4.242599999999998

약수를 찾는 프로그램

다음은 약수를 찾는 프로그램이다. 어떻게 향상시킬 수 있을까?

In [13]:

```
x = 100
divisor = 1
while divisor <= 100:
    if x % divisor == 0:
        print(divisor)
    divisor += 1
```

1
2
4
5
10
20
25
50
100

for를 이용한 반복문

In [14]:

```
for x in [1,2,3,4]:
    print(x)
```

```
1
2
3
4
```

문법은 다음과 같음:

for < variable > in < some collection >:
  < something to do >

조사할 특정 범위가 있다면 for문이 while문보다 간결히 이용될 여지가 많다.

In [15]:

```
for character in "python":
    print(character)
```

```
p
y
t
h
o
n
```

다음의 range() 함수 사용 결과는 0 이상 5미만의 숫자 리스트를 나타낸다.

In [16]:

```
for j in range(0,5):
    print(j)
```

```
0
1
2
3
4
```

예제 : 단어의 순서를 거꾸로 만들어 출력하기

In [19]:

```
string = "abcde"
reversed_string = ""
for s in string:
    reversed_string  = s + reversed_string
print(reversed_string)
```

```
edcba
```

range()의 다양한 활용

Python 2의 range()와 Python 3의 range()는 약간의 차이가 있으나 그 용법은 거의 비슷하다.

In [25]:

```
for j in range(2,6): print(j)
```

```
2
3
4
5
```

In [26]:

```
for j in range(8): print(j)
```

```
0
1
2
3
4
5
6
7
```

In [27]:

```
for j in range(4,10,2): print(j)
```

```
4
6
8
```

In [28]:

```
for j in range(0,-8,-2): print(j)
```

```
0
-2
-4
-6
```

예제

다음은 소수를 찾는 프로그램이다.

다음의 코드는 약간의 수정이 필요하다.

In [29]:

```
for num in range(10, 20):
    for i in range(2, num):
        if num%i == 0:
            j=num/i
            print('%d equals %d * %d' % (num,i,j))
            break
        else:
            print(num, 'is a prime number')
```

10 equals 2 * 5
11 is a prime number
11 is a prime number
11 is a prime number
11 is a prime number
11 is a prime number
11 is a prime number
11 is a prime number
11 is a prime number
11 is a prime number
12 equals 2 * 6
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
13 is a prime number
14 equals 2 * 7
15 is a prime number
15 equals 3 * 5
16 equals 2 * 8
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
17 is a prime number
18 equals 2 * 9
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number
19 is a prime number

연습문제

- 1500에서 2700사이의 숫자 중에서 7로 나누어 떨어지고, 5의 배수인 모든 숫자를 출력하라.

- 킬로미터를 미터로, 미터를 킬로미터로 변환하는 프로그램을 작성하라. 예를 들어 1500m을 입력하면, 1.5km을 출력하고, 2.3km을 입력하면 2300m를 출력하게 하라.

- 다음 패턴을 출력하는 프로그램을 작성하라.

@

@ @

@ @ @

@ @ @ @

@ @ @ @ @

@ @ @ @

@ @ @

@ @

@

- 입력받은 숫자의 약수를 모두 출력하는 프로그램을 만들어라.

- 1부터 100까지 숫자 중 약수의 개수가 5개인 숫자를 모두 찾는 프로그램을 작성하라.

- 자신을 제외한 모든 약수들의 총합이 자기 자신이 되는 수를 완전수라고 한다. 1이상 1000이하의 모든 완전수를 출력하는 프로그램을 작성하라.
  - 예: 6의 약수는 1,2,3,6이며 1+2+3=6이므로 완전수이다.

- 메르센 수(Mersenne number)는 2의 거듭제곱에서 1이 모자란 숫자를 가리킨다.
  - 즉 $2^n - 1$의 형태로 나타난다.
  - $3 = 2^2 - 1$ 이므로 메르센 수이다.
- 1부터 1000까지의 숫자 중 메르센 숫자를 모두 찾는 프로그램을 작성하라.

- 메르센 소수는 메르센 수이면서 소수인 숫자를 나타낸다. 1부터 1000까지의 숫자 중 메르센 소수를 모두 찾는 프로그램을 작성하라.

- 거꾸로 해도 똑같은 숫자를 palindrom이라 한다. 1부터 200까지의 숫자 중 palindrom이면서 소수인 수를 모두 찾는 프로그램을 작성하라.

- 다음 방정식의 해는 모두 양의 정수로 이루어져있다고 한다. 모든 정수해를 찾아라.
  - $x^5 - 25x^4 + 227x^3 - 923x^2 + 1620x - 900 = 0$

- 다음 방정식의 모든 정수해를 찾아라. 해는 음수일 수도 있음
  - $x^5 - x^4 - 33x^3 + 61x^2 + 32x - 60 = 0$