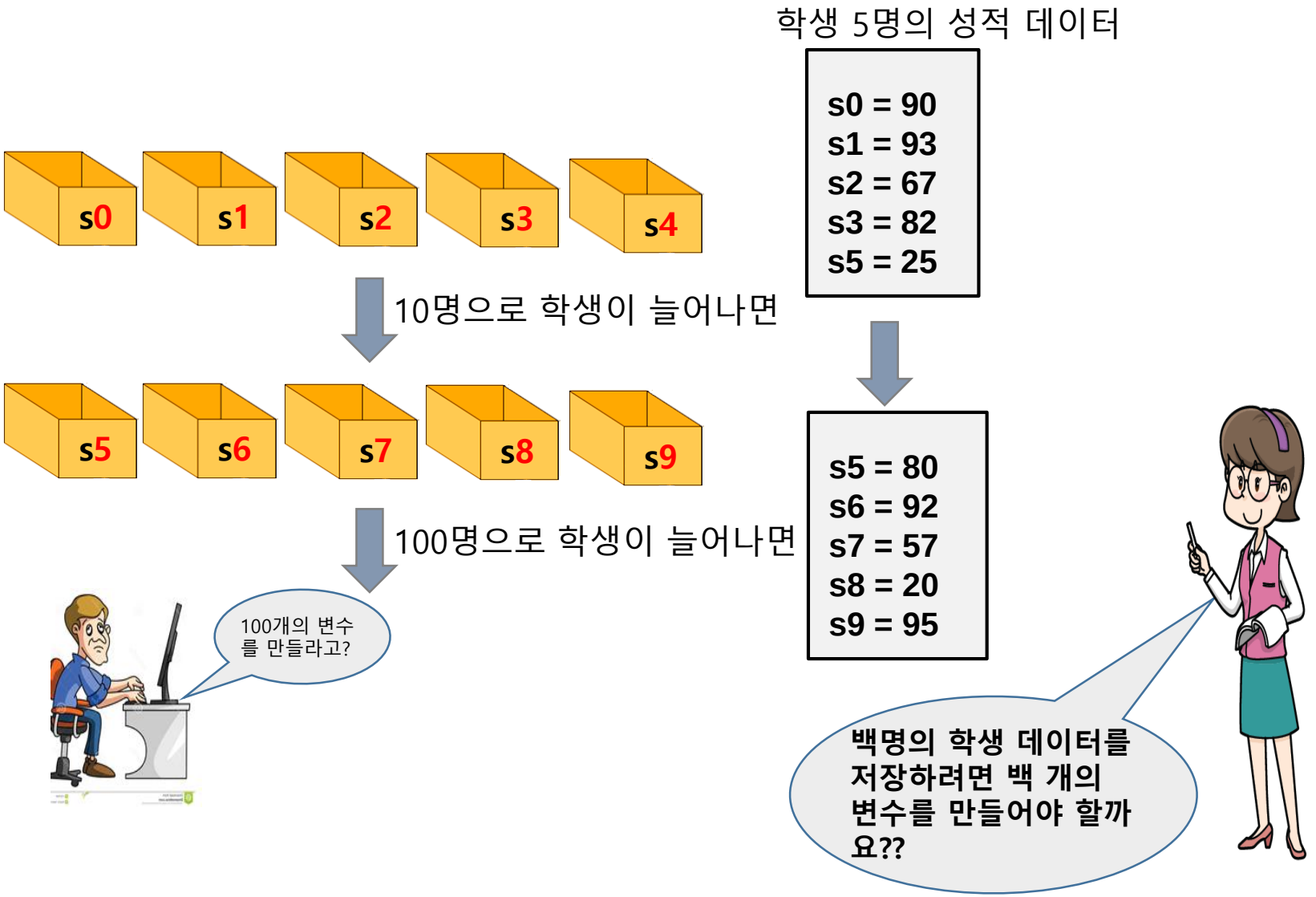
# Container

- list
- tuple
- range
- set

- 필요성


학생 5명의 성적 데이터
s0 = 90
s1 = 93
s2 = 67
s3 = 82
s5 = 25
s0
s1
s2
s3
s4
10명으로 학생이 늘어나면
s5
s6
s7
s8
s9
100명으로 학생이 늘어나면
s5 = 80
s6 = 92
s7 = 57
s8 = 20
s9 = 95
100개의 변수를 만들라고?
백명의 학생 데이터를 저장하려면 백 개의 변수를 만들어야 할까요??

- 여러 개의 데이터를 하나의 변수에 저장하고자 할 때 매우 유용함
- 리스트에 들어있는 데이터를 아이템이라고 부름
- 리스트는 [ ]로 묶어주고, 리스트의 아이템은 ,(컴마)로 구분함
- 리스트의 아이템에 접근하고자 할 때에는 인덱스를 사용함
- 인덱스는 0부터 시작함!!

```
>>> shape= ['turtle', 'arrow', 'circle', 'square']
>>> shape[0]
'turtle'
>>> shape[1]
'arrow'
>>> shape[2]
'circle'
>>> shape[3]
'square'
```

**Quiz**

```
>>> shape[4]
```

- 아이템 수정

```
>>> shape= ['turtle', 'arrow', 'circle', 'square']
>>> shape[1] = 'ARROW'
>>> shape
['turtle', 'ARROW', 'circle', 'square']
```

- 아이템 추가

```
>>> shape= ['turtle', 'ARROW', 'circle', 'square']
>>> shape.append('heart')
>>> shape
['turtle', 'ARROW', 'circle', 'square', 'heart']
>>> shape.append(27)
>>> shape
['turtle', 'ARROW', 'circle', 'square', 'heart', 27]
```

- 아이템 삽입

```
>>> shape= ['turtle', 'ARROW', 'circle', 'square', 'heart', '27']
>>> shape.insert(1, 'heart')
>>> shape
['turtle', 'heart', 'ARROW', 'circle', 'square', 'heart', '27']
>>> shape.insert(3, 'triangle')
['turtle', 'heart', 'ARROW', 'triangle', 'circle', 'square', 'heart', '27']
```

- 아이템 검색
  - **리스트 안에 검색 값이 있으면 True**
  - **리스트 안에 검색 값이 없으면 False**

```
>>> shape= ['turtle', 'heart',  'ARROW', 'triangle', 'circle', 'square', 'heart', '27']
>>> 'heart' in shape
True
>>> 'rectangle' in shape
False
```

- 아이템 삭제

```
>>> shape= ['turtle', 'heart',  'ARROW', 'triangle', 'circle', 'square', 'heart', '27']
>>> del shape[1]
>>> shape
['turtle',  'ARROW', 'triangle', 'circle', 'square', 'heart', '27']
>>> del shape[1]
>>> shape
['turtle', 'triangle', 'circle', 'square', 'heart', '27']
```

- 리스트 슬라이싱(리스트의 일부를 자를 때 사용)

```
>>> shape = ['turtle', 'triangle', 'circle', 'square', 'heart', '27']
>>> shape[0:3] #0부터 3이전까지: 0~2
['turtle', 'triangle', 'circle']
>>> shape[2:4] #2부터 4이전까지: 2~3
['circle', 'square']
>>> shape[:2] #처음부터 2이전까지: 0~1
['turtle', 'triangle']
>>> shape[3:] #3부터 끝까지: 3~4
['square', 'heart', '27']
```

```
>>> a=[1,6,2,4,8,5,3,10,9,7]

>>> a[ : :2]   #2씩 건너뛴 요소를 슬라이싱한다.
[1,2,8,3,9]
>>> a[ : :3]   #3씩 건너뛴 요소를 슬라이싱한다.
[1,4,7,3]
```

- Quiz

```
>>> shape = ['turtle', 'arrow', 'circle', 'square' , 'heart']
>>> shape[0:0] #답이 나올까요? 에러일까요?
>>> shape[:] #답이 나올까요? 에러일까요?
```

- 리스트 연결

```
>>> shape = ['turtle', 'arrow', 'circle', 'square' , 'heart']
>>> shape[0]  + shape[2]
'turtlecircle'
>>> [1, 2, 3, 4]+['Seoul', 'Kwangju']+['Alice', 'Bob']
[1, 2, 3, 4, 'Seoul', 'Kwangju', 'Alice', Bob']
```

- 리스트 안에 리스트(list of list)

```
>>> ID = [1, 2, 3, 4]
>>> ID = list (range(1,5))
>>> area = ['Seoul', 'Kwangju']
>>> name = ['Alice', 'Bob']
>>> student = [ID, area, name]
>>> student
[ [1, 2, 3, 4], ['Seoul', 'Kwangju'], ['Alice', Bob'] ]
```

- Quiz

```
>>> ID = [1, 2, 3, 4]
>>> area = ['Seoul', 'Kwangju']
>>> name = ['Alice', 'Bob']
>>> student = [1, ID, area, name, 'pass' ] #가능할까요?
```

- 아이템 정렬1

```
>>> shape = ['turtle', 'arrow', 'circle', 'square' , 'heart']
>>> shape.sort()
>>> shape
['arrow', 'circle', 'heart', 'square', 'turtle']
```

```
>>> numbers = [ 8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
>>> numbers.sort()
>>> numbers
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- 아이템 정렬2

```
>>> numbers = [ 8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
>>> sorted_numbers = sorted(numbers)
>>> sorted_numbers
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> numbers
[8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
```

- 인덱스 번호를 알고 싶으면

```
>>> numbers = [ 8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
>>> numbers.index(10)
3
>>> numbers.index(8)
1
```

- 리스트에서 인덱스를 음수로 준 경우, 반대로 시작 하며 -1인경우, 맨 마지막 위치에 있는 요소를 가져옴

```
>>> numbers = [ 8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
>>> numbers[-1]
```

# split() 메소드



- 문자열의 단어를 분리하여 리스트 생성

| 사용 방법 | 의미 | 예시 |
|---|---|---|
| split() | 공백을 기준으로 문자열을 나눠 리스트 생성 | a = '1 2 3 4 5'<br>b = a.split()<br>→ b = ['1', '2', '3', '4', '5'] |
| split(문자) | 문자를 기준으로 문자열을 나눠 리스트 생성 | a = '1:2:3:4:5'<br>b = a.split(':')<br>→ b = ['1', '2', '3', '4', '5'] |

# split() 메소드

- split() 메소드

```
>>> wordlist = '''ant baboon badger bat bear beaver camel cat
clam cobra cougar coyote crow deer dog donkey duck eagle
ferret fox frog goat goose hawk lion lizard llama mole
monkey moose mouse mule newt otter owl panda parrot pigeon
python rabbit ram rat raven rhino salmon seal shark sheep
skunk sloth snake spider stork swan tiger toad trout turkey
turtle weasel whale wolf zebra'''.split()
```

```
>>> wordlist
['ant', 'baboon', 'badger', 'bat'.....'zebra' ]
```

# len() 함수



- len() 함수
  - ❖ 리스트의 개수, 문자열의 길이를 반환하는 함수

```
>>> len(wordlist)
63
>>> len('hello')
5
```

# list() 함수



- list 함수: list형으로 형변환

| 사용 방법 | 의미 | 예시 |
|---|---|---|
| list(문자열) | 문자열을 한 글자씩 분리하여 리스트 생성 | list('12345') → ['1', '2', '3', '4', '5'] |

```
>>> list('hello')
['h', 'e', 'l', 'l', 'o']
```

# range() 함수

- range 함수: 규칙적인 숫자열로 구성된 범위 생성

| 사용 방법 | 의미 | 리스트생성 |
|---|---|---|
| range(끝) | [0, 1, 2, 3, …, 끝-1] | list(range(4)) → [0, 1, 2, 3] |
| range(시작, 끝) | [시작, 시작+1, 시작+2, …, 끝-1] | list(range (3, 5)) → [3, 4] |
| range(시작, 끝, 스텝) | [시작, 시작+스텝, 시작+스텝×2, …, 시작+스텝×k] | list(range(2, 11, 2)) → [2, 4, 6, 8, 10] |
| range(시작, 끝, -스텝) | [시작, 시작-스텝, 시작-스텝×2, …, 시작-스텝×k] | list(range(9, 1, -2)) → [9, 7, 5, 3] |

```
>>> range(10)
range(0,10)     => 의미 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
>>> range(10, 20)
range(10,20)    => 의미 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
>>> list(range(10, 20))
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
>>> a= range(1,100)
>>> a[2:5]  # 인덱스2부터 5까지 슬라이싱으로 range를 만듦
range(3,6)
>>> b=a[2:5]
[3,4,5]
>>> b[0]=5   #range도 불변객체이므로 재할당할 수 없음
에러
```

# 문자 리스트 생성 예제



- 예제

```
>>> a = 'Hello world'
>>> b = list(a)
>>> c = a.split()
>>> d = a.split('o')
>>> a
'Hello world'
>>> b
['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
>>> c
['Hello', 'world']
>>> d
['Hell', ' w', 'rld']
```

# 숫자 list 생성 예제



- 예제

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(3, 7))
[3, 4, 5, 6]
>>> list(range(10, 1, -2))
[10, 8, 6, 4, 2]
>>> list(range(1, 7, -1))
[]
```

# 실습1. 오늘의 명언



- 리스트에 여러개의 속담을 저장한 후, 속담중에서 하나를 랜덤하게 골라서 제공하는 프로그램을 작성하시오.

```
----------------------------------
          오늘의 명언
----------------------------------
고생없이 얻을 수 있는 진실로 귀중한 것은 하나도 없다.
>>>
```

- 속담리스트

```
꿈을 지녀라. 그러면 어려운 현실을 이길 수 있다.
고생없이 얻을 수 있는 진실로 귀중한 것은 하나도 없다.
사람은 사랑할 때 누구나 시인이 된다.
시작이 반이다.
나는 사랑으로 내가 이해하는 모든 것들을 이해한다.
```

# 실습2-1. 상품재고관리



- 리스트에 상품목록을 저장한 후, 상품을 삽입,삭제,추가,검색하는 프로그램을 작성하시오.
  - 상품리스트

```
커피, 우유, 바나나, 감귤, 화장지, 장갑, 이불, 베개, 장난감, 음료수, 빵
```

  - 상품 삽입

```
----------------------------------------
삽입하고자 하는 상품을 입력하세요. 사과
삽입하고자 하는 위치를 입력하세요. 4
----------------------------------------
[커피, 우유, 바나나, 감귤, 사과, 화장지, 장갑, 이불, 베개, 장난감, 음료수, 빵]
>>>
```

- 상품 삭제

```
----------------------------------------
삭제하고자 하는 상품을 입력하세요. 장갑
----------------------------------------
[커피, 우유, 바나나, 감귤, 사과, 화장지, 이불, 베개, 장난감, 음료수, 빵]
>>>
```

- 상품 추가

```
----------------------------------------
추가하고자 하는 상품을 입력하세요. 물
----------------------------------------
[커피, 우유, 바나나, 감귤, 사과, 화장지, 이불, 베개, 장난감, 음료수, 빵, 물]
>>>
```

# 실습2-4. 상품재고관리



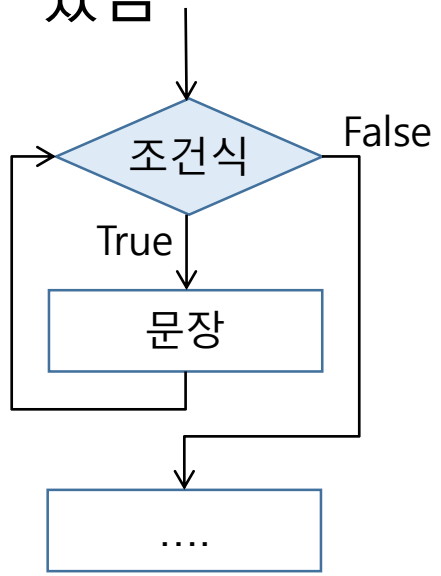
- 상품 검색

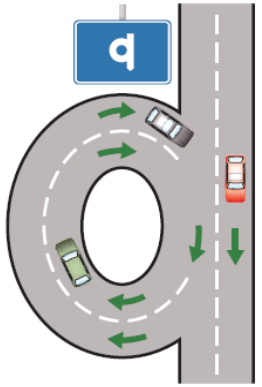
```
----------------------------------------
검색하고자 하는 상품을 입력하세요. 화장지
----------------------------------------
True
>>>
```

- 반복문
  - 반복(iteration)은 동일한 문장을 여러 번 반복시키는 구조
  - 조건식이 **True**인 동안 실행문장을 반복함
  - If문과 마찬가지로 반복실행 되어야 할 문장은 들여쓰기가 되 있어야 함
  - 컴퓨터는 반복작업을 빠르게 실행 할 수 있음

- **반복문 형식1**

```
while 조건식:
    실행문장
```




- **반복문 형식2**

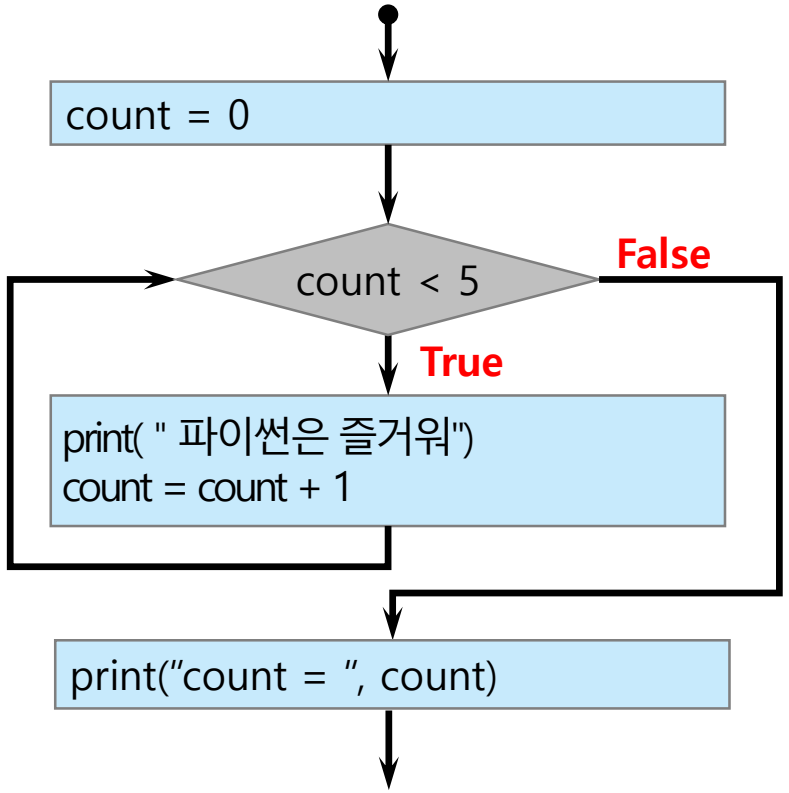
```
while 1:
    ...
    break
```

조건이 항상 True인 반복문
1은 True의 의미를 가짐
이 반복문을 빠져나오기 위해 break를 꼭 써야함

- 반복 조건식에 사용할 변수
  - while 조건식에 사용할 변수는 반드시 먼저 정의되어 있어야 함

```
count = 0

while count < 5:
    print ('파이썬은 즐거워')
    count = count + 1
print ('count= ', count)
```



- 반복문

```
for 변수 in 범위:
    실행문장1
    실행문장2
    실행문장3
```

  - 범위의 처음부터 끝까지 반복함
  - while과의 차이점
    - while: 조건비교결과가 참인 동안 실행문장들이 반복해서 실행됨
    - for : 범위의 처음부터 끝까지 반복해서 실행됨

- 범위에 숫자 리스트를 준 예제

```
for count in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    print ('나무를', count, '번 찍었습니다.', '딱! ' *count)
print ('넘어가즈아~~~')
```

- 동일한 표현

```
for count in range(1, 11):
```

```
나무를 1 번 찍었습니다. 딱!
나무를 2 번 찍었습니다. 딱! 딱!
나무를 3 번 찍었습니다. 딱! 딱! 딱!
나무를 4 번 찍었습니다. 딱! 딱! 딱! 딱!
나무를 5 번 찍었습니다. 딱! 딱! 딱! 딱! 딱!
나무를 6 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱!
나무를 7 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱!
나무를 8 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱!
나무를 9 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱!
나무를 10 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱!
넘어가즈아~~~
>>>
```

- 구구단 예제

```
for i  in range(2,10):
    for j  in range(1,10):
        print (i, '*', j, '=', i*j)
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
....

9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
>>>
```

- 범위에 문자열 리스트를 준 예제

```
for thing in ['programming', 'pasta', 'spam', 'play park']:
    print ('I really like ' + think + '.')
```

- 실행결과

```
I really like programming.
I really like pasta.
I really like spam.
I really like play park.
>>>
```

- 범위에 변수를 준 예제

```
stuff = ['programming', 'pasta', 'spam', 'play park']
for thing in stuff:
    print ('I really like ' + think + '.')
```

- 실행결과

```
I really like programming.
I really like pasta.
I really like spam.
I really like play park.
>>>
```

- 범위에 문자열을 준 예제

```
for i  in 'Hello world!':
      print (i, end=' ')
```

```
Hello world!
>>>
```

```
for i  in 'Hello world!':
      print (i)
```

```
H
e
l
l
o

w
o
r
l
d
!
>>>
```

# while문/for문 이용하여 나무 찍기

열번 찍어 안 넘어가는 나무없다!!!

나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.

간다~간다~넘어간다!!!

# while문/for문 이용하여 동전모으기

- Scenario of "티끌 모아 태산"

```
티끌 모아 태산!!!

땡그랑, 100원을 모았습니다.
땡그랑, 땡그랑, 동전을 200원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 동전을 300원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전을 400원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전을 500원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전을 600원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전을 700원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전을 800원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전을 900원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전을 1000원을 모았습니다.

산다. 산다. 집 산다!!!
```

# while문/for문 로또번호 생성기

- 로또번호는 1~45사이 random.randint()를 사용
- 6개의 번호가 필요함.
- 중복번호가 나올 수도 있음

```
로또번호 생성기 프로그램
1번째 번호는 1입니다.
2번째 번호는 8입니다.
3번째 번호는 19입니다.
4번째 번호는 41입니다.
5번째 번호는 14입니다.
6번째 번호는 32입니다.
```

```
로또번호 생성기 프로그램
1번째 번호는 40입니다.
2번째 번호는 11입니다.
3번째 번호는 11입니다.
4번째 번호는 38입니다.
5번째 번호는 31입니다.
6번째 번호는 5입니다.
```

# 로그인 프로그램

- 올바른 암호를 입력할 때까지 암호를 입력받는 프로그램을 작성하세요. 올바른 암호를 입력하면 로그인 성공을 출력하세요.
  - 알고리즘
    1. 초기값 password = ''
    2. 암호가 "pythonisfun"이 아니면 반복한다.
       - 암호를 입력하세요.
       - password=raw_input()
    3. 암호가 올바르면 "로그인 성공"을 출력한다.

```
암호를 입력하세요.
pythonis
암호를 입력하세요.
funfunfun
암호를 입력하세요.
pythonisfun
로그인 성공

>>>
```

# 구구단 출력

- 원하는 단을 입력 받아 그 단의 구구단을 출력하는 프로그램을 while문을 이용하여 작성하세요.
  - 알고리즘
    1. 단을 입력 받는다.
    2. while문의 조건을 준다.
       - 출력한다.
       - 조건이 반복되도록 변수의 값을 변경한다.

```
원하는 단은: 9
9*1=9
9*2=18
9*3=27
9*4=36
9*5=45
9*6=54
9*7=63
9*8=72
9*9=81
```

# random.randint()와 반복문 이용

- 입력한 값이 1보다 작은 경우 “잘못 입력하셨습니다.” 메시지를 출력한 후 다시 입력 받도록 작성해보세요.

```
1이상의 숫자를 입력해주세요.
-3
잘못 입력하셨습니다.
1이상의 숫자를 입력해주세요.
100
시작 범위 1부터 끝 범위 100 가운데 선택된 랜덤 넘버는 78입니다.
>>>
```

# 1~100까지 숫자 더하기

- 1~100까지 숫자를 더해 합을 출력하는 프로그램을 while/for문을 사용하여 작성하시오.

- 튜플(tuple)
  - 리스트와 동일하지만, 튜플은 불변객체이다는 차이점이 있음

```
>>> a=(1,2,3,4,5)
>>> a
(1,2,3,4,5)
>>> b=tuple([1,2,3,4,5])
>>> b[0]
1
>>> b[0]=0  #불변객체이므로 재할당이 되지 않는다.

>>> c=tuple(range(1,10))
```

- 집합 (set)
  - set은 순서가 없으며, 수정이 가능하고, 값이 중복되지 않음

```
>>> a={1,2,3,4,5}
>>> a
{1,2,3,4,5}
>>> b=set([1,1,4,2,2,6,6,6,6,7,8,10,4,2,8,6,3])
>>> b
{1,2,3,4,6,7,8,10}
```

- set은 집합이므로, 합집합, 차집합, 교집합, 대칭차집합을 표현할 수 있음

```
>>> a={1,4,7}
>>> b={1,3,6,8,10}
>>> a | b   #합집합
{1, 3, 4, 6, 7, 8, 10}
>>> a-b   #차집합
{4, 7}
>>> a&b   #교집합
{1}
>>> a^b   #대칭차집합    #(합집합)-(교집합)
{3, 4, 6, 7, 8, 10}
```

- 딕셔너리
  - key와 value 쌍으로 이루어진 자료구조

```
>>> a={'a' : 1, 'b': 2}
>>>  a
{'a' : 1, 'b': 2}
```

- 키-값 쌍(key-value pair)으로 저장
- 딕셔너리는 중괄호{ }로 생성
- 딕셔너리 생성

```
>>> phone_book={'신사임당': '010-1234-5678', '홍길동': '010-4532-4532'}
>>> phone_book
{'신사임당': '010-1234-5678', '홍길동': '010-4532-4532'}
```

```
>>> phone_book['이순신']= '010-0987-6543'
>>> phone_book
{'신사임당': '010-1234-5678', '홍길동': '010-4532-4532', '이순신': '010-0987-6543'}
```

- 딕셔너리 검색

```
>>> phone_book={'신사임당': '010-1234-5678', '홍길동': '010-4532-4532', '이순신': '010-0987-6543'}
>>> phone_book['이순신']
010-0987-6543
```

```
>>> phone_book.keys()
dict_keys(['신사임당', '홍길동', 이순신'])
```

```
>>> phone_book.values()
dict_values(['010-1234-5678', '010-4532-4532', '010-0987-6543'])
```

- 딕셔너리 항목 삭제

```
>>> del phone_book['홍길동']
>>> phone_book
{'신사임당': '010-1234-5678', '이순신': '010-0987-6543'}
>>> phone_book.clear() #모든 항목 삭제
>>> phone_book
{}
```

# 편의점 재고관리 프로그램

- 편의점 아이템
  - items = {"커피": 7, "펜": 3, "종이컵": 2, "우유": 1, "콜라": 4, "책": 5}

```
----------메뉴 ----------
1. 재고확인    2. 판매    3. 종료
-----------------------------------
1
재고 확인을 위한 물건명을 입력하세요. 커피
7
----------메뉴 ----------
1. 재고확인    2. 판매    3. 종료
-----------------------------------
2
판매한 물건명을 입력하세요. 콜라
콜라의 판매 개수를 입력하세요. 2
----------메뉴 ----------
1. 재고확인    2. 판매    3. 종료
-----------------------------------
1
재고 확인을 위한 물건명을 입력하세요. 콜라
2
----------메뉴 ----------
1. 재고확인    2. 판매    3. 종료
-----------------------------------
3
>>>
```

# 한영사전 프로그램

```
------------------메뉴 ----------------
1. 단어 추가    2. 사전 보기    3. 종료
-----------------------------------------
1
한글 단어를 입력하세요: 사과
영어 단어를 입력하세요: apple
1
한글 단어를 입력하세요: 기차
영어 단어를 입력하세요: train
2
찾고자 하는 단어를 입력하세요:기차
train
3
>>>
```