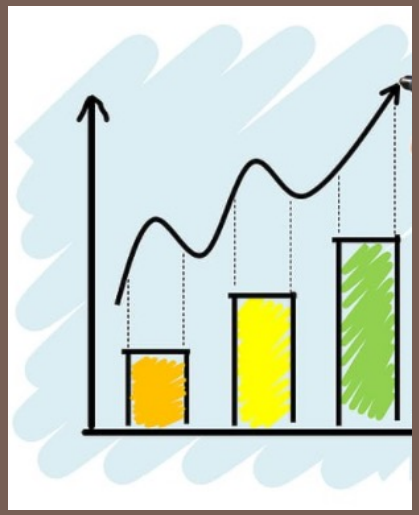
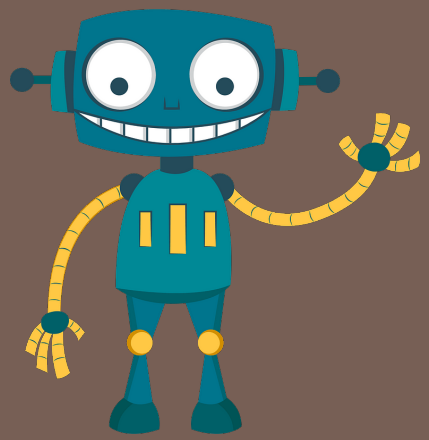
# 파이썬익스프레스





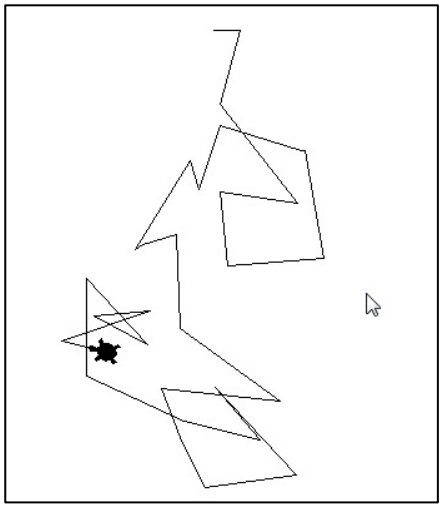
## 4장 반복문

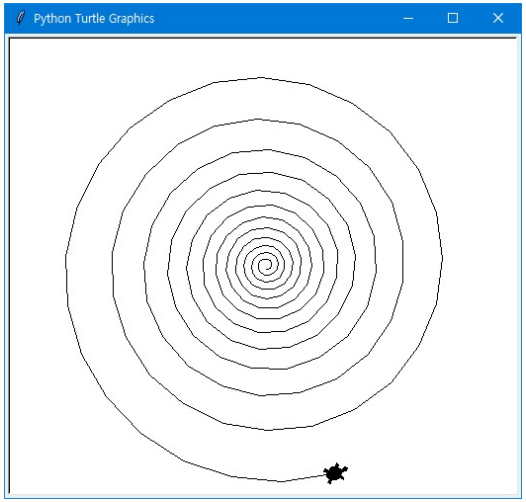
# 학습 목표

- 반복문의 필요성을 이해한다.
- **for** 문을 사용하여 정해진 횟수만큼 반복하는 방법을 학습한다.
- range() 함수를 이해하고 사용할 수 있다.
- **while** 문을 사용하여 조건으로 반복하는 방법을 학습한다.
- 중첩 반복문의 개념을 이해한다.
- 무한 루프가 사용되는 환경을 이해한다.

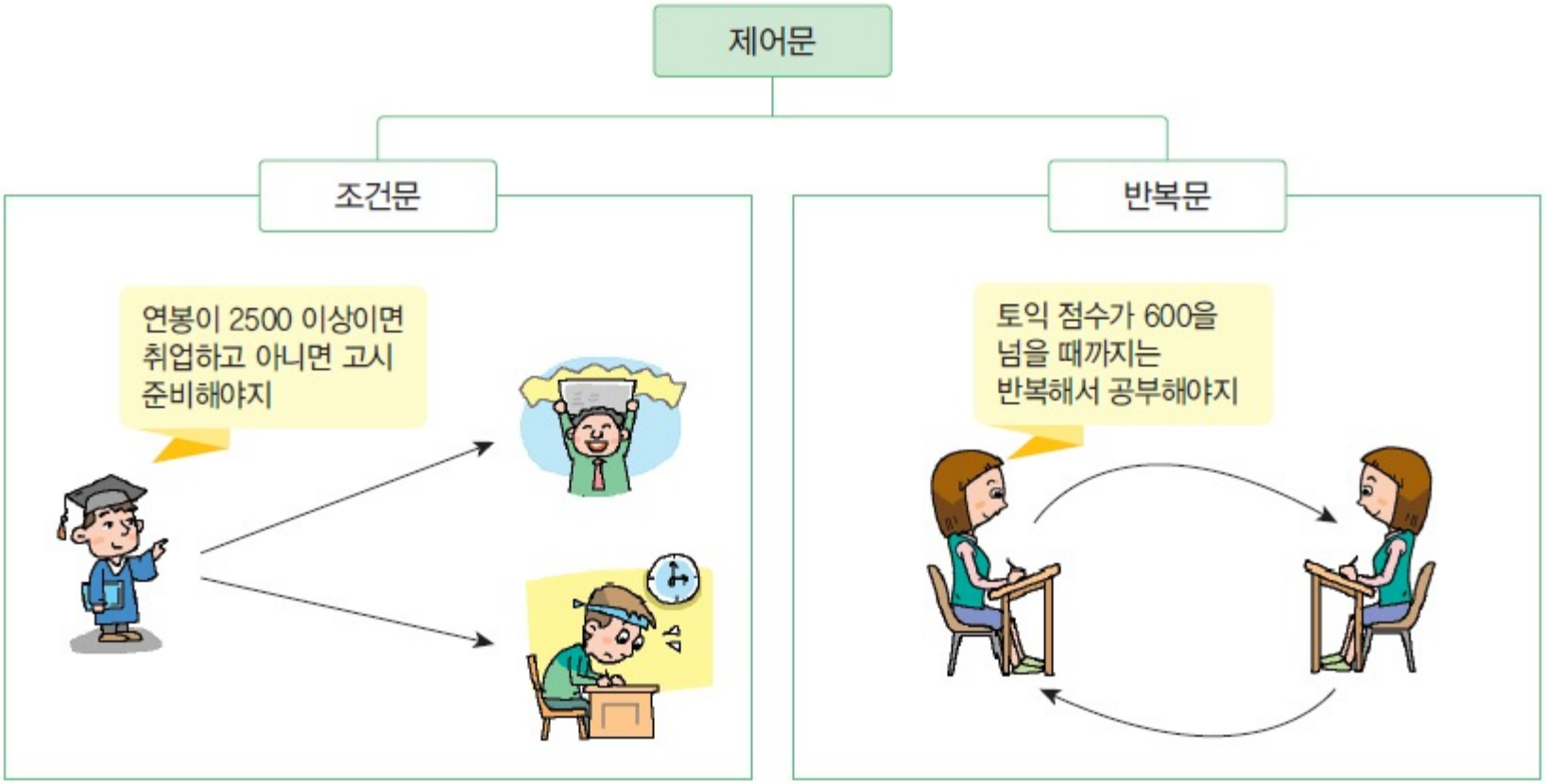
# 이번 장에서 만들 프로그램

```
1부터 100 사이의 숫자를 맞추시오
숫자를 입력하시오: 50
너무 낮음!
숫자를 입력하시오: 75
너무 낮음!
숫자를 입력하시오: 89
축하합니다. 시도횟수= 3
```

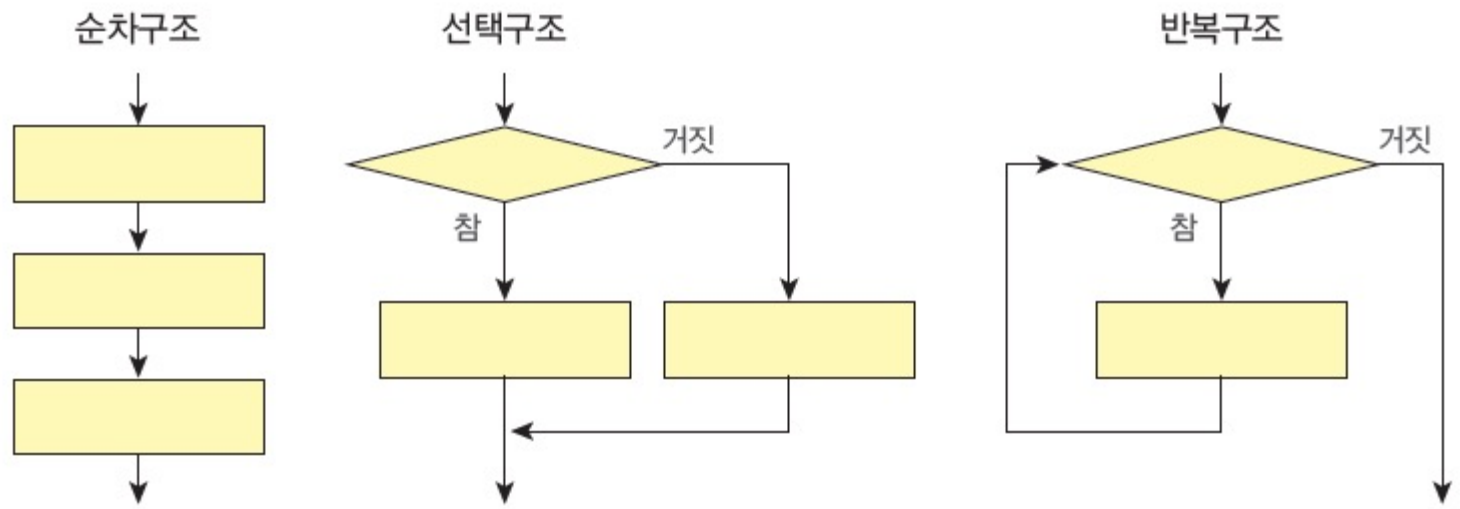




# 제어문


제어문
조건문
반복문
연봉이 2500 이상이면 취업하고 아니면 고시 준비해야지
토익 점수가 600을 넘을 때까지는 반복해서 공부해야지

# 3가지의 제어구조

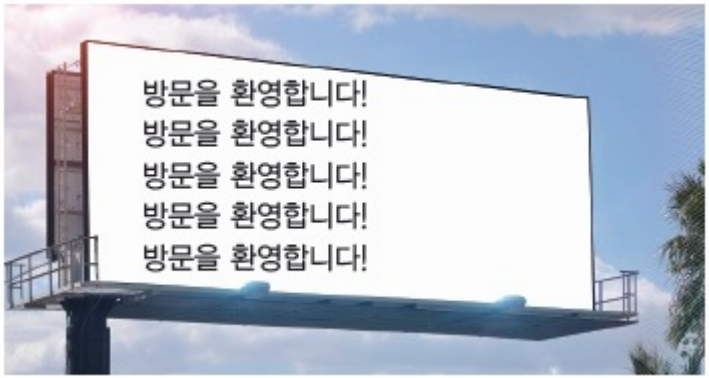


# 반복의 중요성

- 우리들의 생활에서는 반복적인 작업들이 필요하다.
- 반복적이고 단순한 작업은 컴퓨터를 이용하여 자동화하면 된다.


# 반복의 예

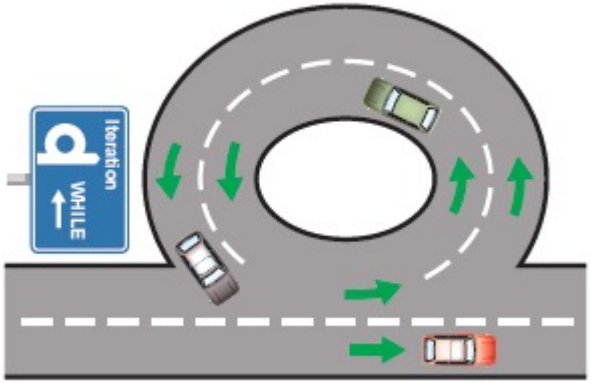
- 화면에 회사에 중요한 손님이 오셔서 대형 전광판에 “방문을 환영합니다!”를 5번 출력해야 한다고 가정하자.



# 반복을 사용 vs 사용하지 않을 때

```
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
print("방문을 환영합니다!")
```

```
for i in range(1000):
        print(“i =”, i, 방문을 환영합니다!")
```



# 반복의 종류

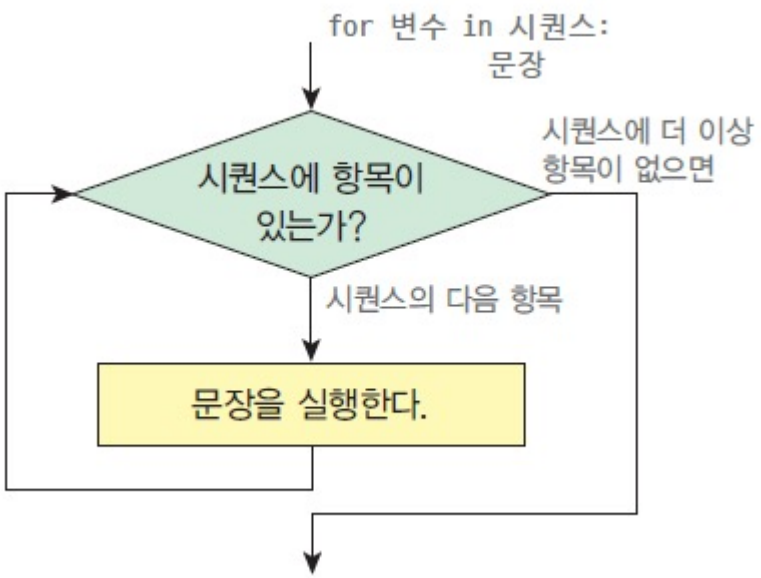
- 횟수 반복(for 문): 정해진 횟수만큼 반복한다.
- 조건 반복(while 문): 특정한 조건이 성립되는 동안 반복한다.

조건 반복

횟수 반복

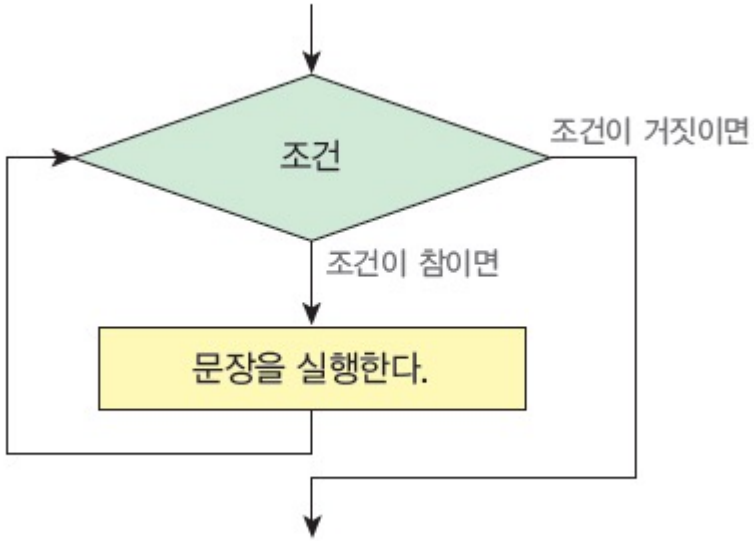
# 횟수 반복

- 횟수 반복은 반복을 시작하기 전에 반복의 횟수를 미리 아는 경우에 사용한다.



# 조건 반복

- 조건 반복은 특정한 조건이 만족되는 동안 계속 반복한다.



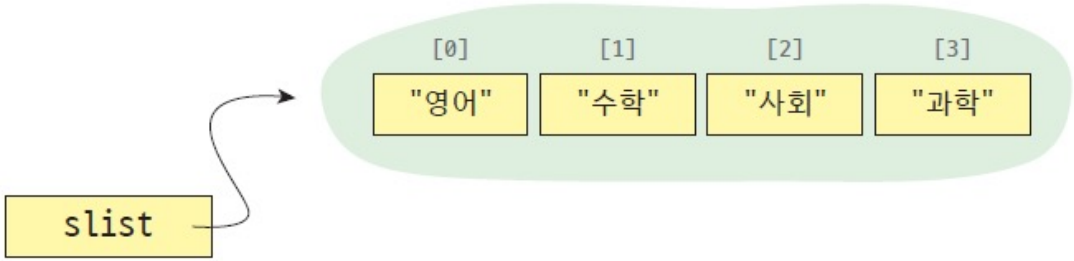
# 중간점검

- 프로그램에 반복 구조가 필요한 이유는 무엇인가?
- 반복 구조에는 _____반복과, _____반복이 있다.
- 조건 반복과 횟수 반복을 설명해보자.

# 리스트(list) 란?

- 연속된 항목들을 저장하는 자료 구조
- mutable : 추가, 삭제 등 변경이 가능한 자료구조(list..)
- immutable : 변경이 불가능한 자료구조(tuple…)


```
slist = [ "영어", "수학", "사회", "과학" ]
```



# 리스트에 동적으로 항목 추가

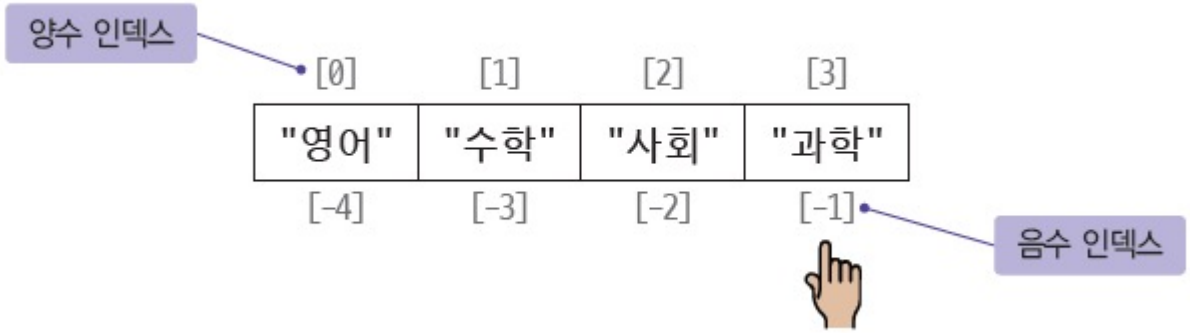
```
list = []                    # 공백 리스트를 생성한다.
list.append(1)               # 리스트에 정수 1을 추가한다.
list.append(2)               # 리스트에 정수 2을 추가한다.
list.append(6)               # 리스트에 정수 6을 추가한다.
list.append(3)               # 리스트에 정수 3을 추가한다.

print(list)                  # 리스트를 출력한다.
```

```
[1, 2, 6, 3]
```

# 리스트의 인덱스

- 인덱스는 0부터 시작한다.
- 첫 번째 항목의 인덱스는 0이고 두 번째 항목의 인덱스는 1, 세 번째 항목의 번호는 2인 것이다.

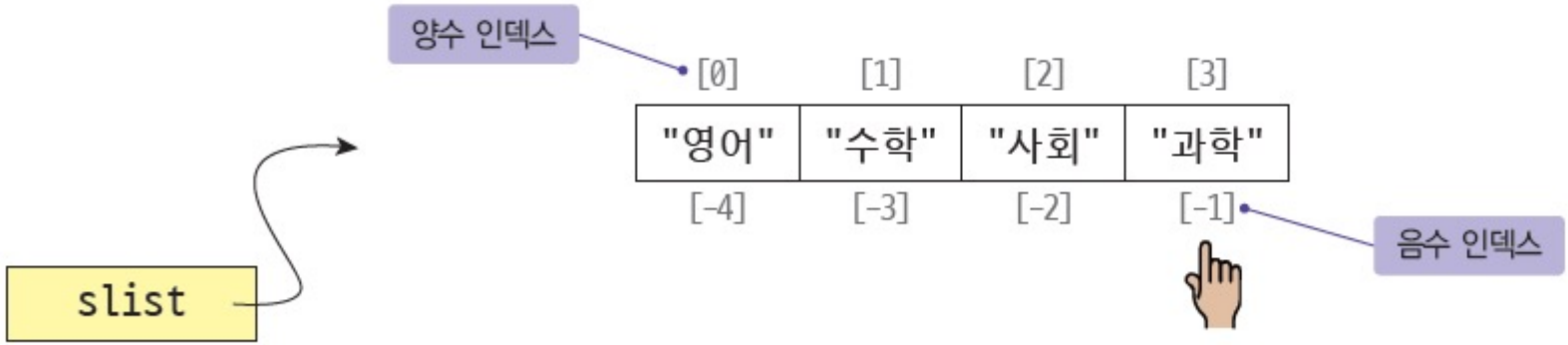
```
>>> slist = [ "영어", "수학", "사회", "과학" ]
>>> slist[0]
영어                                  # 리스트의 첫 번째 항목을 출력한다.
```



# 리스트 항목 변경

```
slist = [ "영어", "수학", "사회", "과학" ]
slist[-1] = "컴퓨터"
print(slist)
```

```
['국사', '수학', '사회', '컴퓨터']
```



# 중간점검

- [ 1, 2, 3, 4, 5 ]를 저장하는 리스트 myList를 생성해보자.
- myList의 첫 번째 항목을 0으로 변경해보자.
- myList의 마지막 항목을 9로 변경해보자.

# 횟수 제어 반복

Syntax: for 문

형식

```
for 변수 in 리스트 :
    문장1
    문장2
```

예

```
for i in [1, 2, 3, 4, 5] :
    print("방문을 환영합니다.")
```

- 리스트의 값들이 하나씩 변수 i에 할당되면서 반복된다.
- 콜론(:)은 복합문을 의미한다.
- 반복되는 문장은 동일하게 들여쓰기가 되어야 한다.
- 반복되는 문장들

```
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
방문을 환영합니다!
```

# 횟수 제어 반복의 이해


for i in [ 1, 2, 3, 4, 5 ] :
print("방문을 환영합니다.")
1
i
for i in [ 1, 2, 3, 4, 5 ] :
print("방문을 환영합니다.")
2
i
for i in [ 1, 2, 3, 4, 5 ] :
print("방문을 환영합니다.")
3
i
for i in [ 1, 2, 3, 4, 5 ] :
print("방문을 환영합니다.")
4
i
for i in [ 1, 2, 3, 4, 5 ] :
print("방문을 환영합니다.")
5
i

# 중간점검

- 다음 코드의 출력을 쓰시오.

```
for i in [9, 8, 7, 6, 5] :
    print("i=", i)
```

# range() 함수

- range() 함수로 반복 횟수를 전달하면 range() 함수가 자동으로 순차적인 정수들을 생성해준다.

Syntax: range() 함수를 사용하는 for 문

형식
```
for 변수 in range(종료값) :
    문장1
    문장2
```

예
```
for i in range(5) :          # 0, 1, 2, 3, 4가 생성된다.
    print("방문을 환영합니다.")
```

0, 1, 2, 3, 4의 값들이 생성되어서 변수 i에 할당된다.

반복되는 문장으로 들여쓰기 하여야 한다.

# range() 함수의 이해

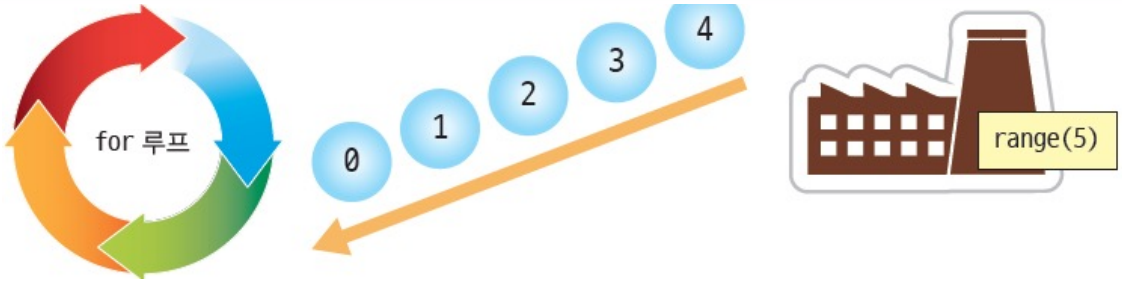
Syntax: range() 함수

형식 `range(start=0, stop, step=1 )`

- `start=0` : 시작값이다.
- `stop` : 종료값이지만 stop은 포함되지 않는다.
- `step=1` : 한 번에 증가되는 값이다.

예

```
for i in range(5) :          # 0, 1, 2, 3, 4
    ...
for i in range(1, 6) :       # 1, 2, 3, 4, 5
    ...
for i in range(1, 6, 2) :    # 1, 3, 5
    ...
```



# 예제 : step

간격 지정

```
for i in range(1, 6, 1):
        print(i, end=" ")
```

```
1 2 3 4 5
```

역순

```
for i in range(10, 0, -1):
        print(i, end=" ")
```

```
10 9 8 7 6 5 4 3 2 1
```

# 예제

1부터 n까지의 합

```
sum = 0
n = 10
for i in range(1, n+1) :
        sum = sum + i
print("합=", sum)
```

```
합= 55
```

구구단 출력

```
for i in range(1, 6) :
        print("9 *", i, "=", 9*i)
```

```
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
```

# 중간점검

1. 다음 코드의 출력을 쓰시오.

```
for i in range(1, 5, 1) :
    print(2*i)
```

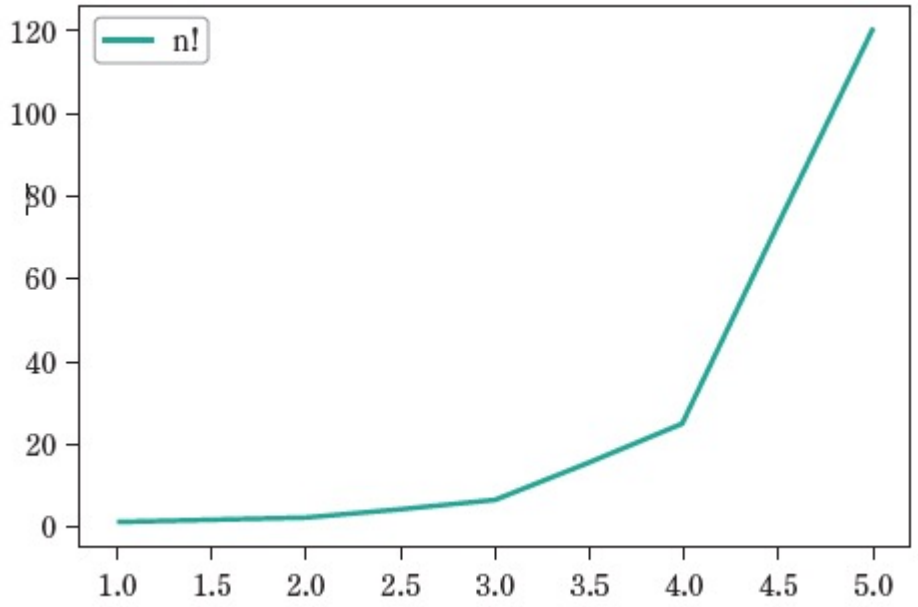
2. 다음 코드의 출력을 쓰시오.

```
for i in range(10, 0, -2) :
    print("Student" + str(i))
```

# Lab: 팩토리얼 계산하기

**for**문을 이용하여서 팩토리얼을 계산해보자.

```
정수를 입력하시오: 10
10!은 3628800이다.
```



# Lab: 팩토리얼 계산하기

```
n = int(input("정수를 입력하시오: "))
fact = 1
for i in range(1, n+1):
        fact = fact * i
print(n, "!은", fact, "이다.")
```

| | i의 값 | 반복여부 | fact의 값 |
|---|---|---|---|
| 1번째 반복 | 1 | 반복 | 1*1 |
| 2번째 반복 | 2 | 반복 | 1*1*2 |
| 3번째 반복 | 3 | 반복 | 1*1*2*3 |
| 4번째 반복 | 4 | 반복 | 1*1*2*3*4 |
| 5번째 반복 | 5 | 반복 | 1*1*2*3*4*5 |

# 도전문제

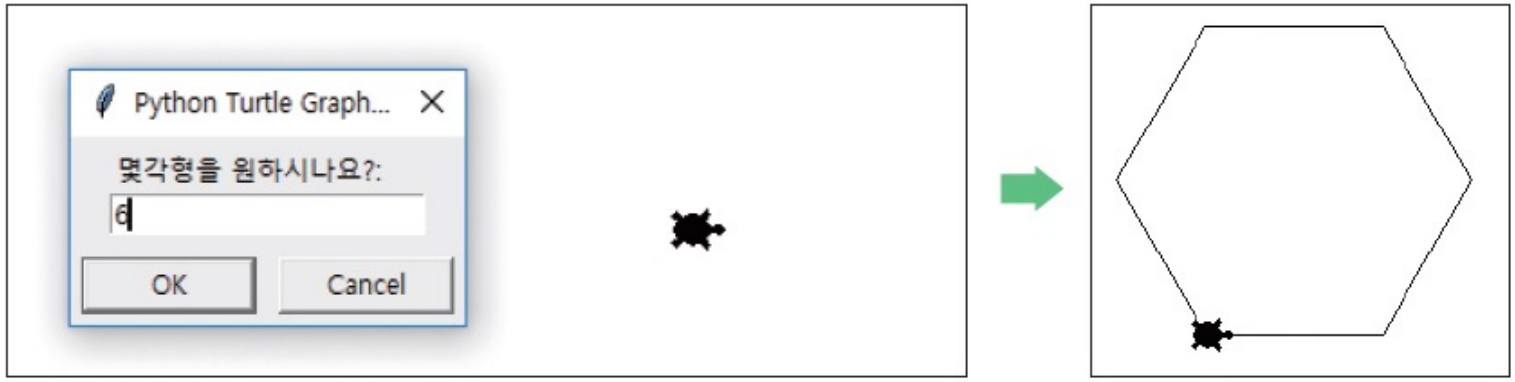
1. 팩토리얼을 거꾸로 계산해보자.

$$n! = n \times (n-1) \times \cdots\cdots \times 2 \times 1$$

앞의 프로그램을 어떻게 수정하여야 하는가?

# Lab: n-각형 그리기

- 터틀 그래픽에서 사용자로부터 정수 n을 받아서 n-각형을 그리는 프로그램을 작성해보자.

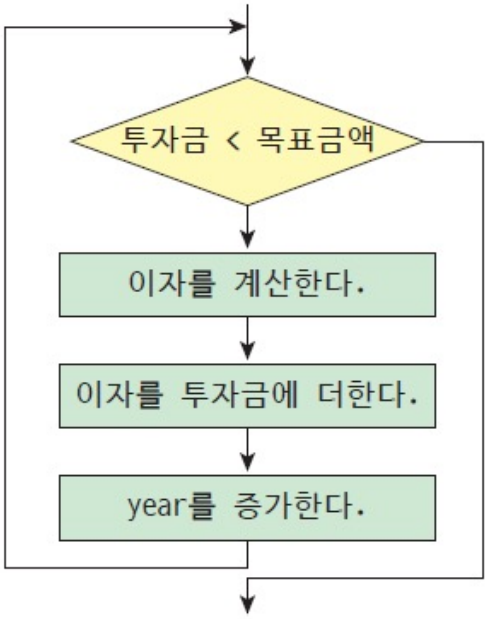


# Lab: n-각형 그리기

```
import turtle
t = turtle.Turtle()
t.shape("turtle")

s = turtle.textinput("", "몇각형을 원하시나요?:")
n=int(s)

for i in range(n):
        t.forward(100)
        t.left(360/n)

turtle.mainloop()
turtle.bye()
```

# 조건 제어 반복

- 조건 제어 반복은 어떤 조건이 만족 되는 동안 반복한다.




# 조건 제어 반복

Syntax: while 문

형식

```
while 조건식 :
        문장1
        문장2
```

참이나 거짓으로 계산되는 조건식,
관계 연산자 == != < > >= <=을 사용한다.

콜론(:)은 복합문을 의미한다.

예

```
while money < TARGET :
    money = money + money * rate
    year = year + 1
```

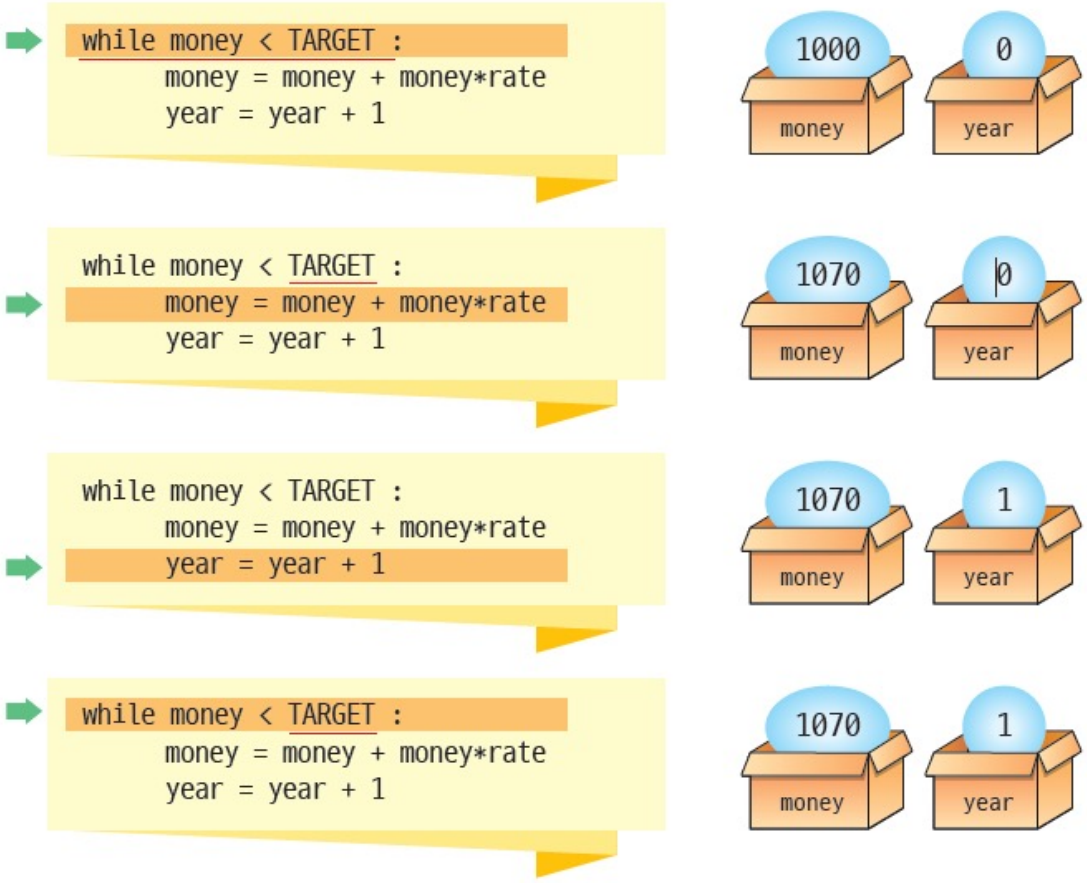
조건이 참이면 반복되는 문장들

반복되는 문장은 동일하게
들여쓰기가 되어야 한다.

```
TARGET = 2000                    # 목표 금액
money = 1000                     # 초기 자금
year = 0                  # 연도
rate = 0.07                      # 이자율

# 현재 금액이 목표 금액보다 작으면 반복한다.
while money < TARGET :
        money = money + money * rate
        year = year + 1

print(year, "년")
```

```
11 년
```

# 조건 제어 반복


while money < TARGET :
money = money + money*rate
year = year + 1
1000
money
0
year
while money < TARGET :
money = money + money*rate
year = year + 1
1070
money
0
year
while money < TARGET :
money = money + money*rate
year = year + 1
1070
money
1
year
while money < TARGET :
money = money + money*rate
year = year + 1
1070
money
1
year

# 예제: 1과 10까지의 합 계산하기

```
# 제어 변수를 선언한다.
i = 1
sum = 0

# i 값이 10보다 작으면 반복
while i <= 10 :
        sum = sum + i
        i = i + 1

print("합계는", sum)
```

```
합계는 55
```

# else가 있는 while 루프

```
i = 0

while i < 3:
        print("루프 안쪽")
        i = i + 1
else:
        print("else 부분")
```

```
루프 안쪽
루프 안쪽
루프 안쪽
else 부분
```

# 무한 반복 오류

```
# 무한 루프의 예
i = 0

# 변수 i를 증가시키는 부분이 없어서 무한루프가 된다.
while i < 10 :
        print("Hello!")
```

```
Hello!
Hello!
Hello!
...
```

# 중간점검

1. **if** 문과 **while** 문을 비교하여 보라. 조건식이 같다면 어떻게 동작하는가?

2. **for** 문과 **while** 문을 비교해보자. 어떤 경우에 **for** 문을 사용하는 것이 좋은가? 또 어떤 경우에 **while** 문을 사용하는 것이 좋은가?

# Lab: 구구단 출력

- 구구단 중에서 9단을 반복문을 이용하여 출력하여 보자. 9*1, 9*2, 9*3, .., 9*9까지 9번 반복시키면 출력하면 될 것이다.

```
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
```

# Lab: 구구단 출력

```
dan = int(input("원하는 단은: "))
i = 1

while i <= 9:
        print("%s*%s=%s" % (dan, i, dan*i))
        i = i + 1
```

NOTE

**print() 함수**

print() 함수를 이용하여 특정 형식의 값을 출력할 때는 위와 같이 "%d" % dan과 같은 형식을 사용한다. %d는 정수 형식으로 출력하라는 의미이다. % 이후의 변수의 값이 %d 자리에 출력된다. 예를 들어서 변수 dan에 9가 저장되어 있다면 "%d" % dan은 "9"가 된다.

# Lab: 숫자 맞추기 게임

- 이 예제는 프로그램이 가지고 있는 정수를 사용자가 알아맞히는 게임이다. 사용자가 답을 제시하면 프로그램은 자신이 저장한 정수와 비교하여 제시된 정수가 더 높은지 낮은지 만을 알려준다.

```
1부터 100 사이의 숫자를 맞추시오
숫자를 입력하시오: 50
너무 낮음!
숫자를 입력하시오: 75
너무 낮음!
숫자를 입력하시오: 89
축하합니다. 시도횟수= 3
```

# Solution:

```
import random

tries = 0                                   # 시도 횟수
guess = 0;                                  # 사용자의 추측값
answer = random.randint(1, 100)     # 1과 100사이의 난수

print("1부터 100 사이의 숫자를 맞추시오")

while guess != answer:
    guess = int(input("숫자를 입력하시오: "))
    tries = tries + 1
    if guess < answer:
        print("너무 낮음!")
    elif guess > answer:
        print("너무 높음!")

if guess == answer:
    print("축하합니다. 시도횟수=", tries)
else:
    print("정답은 ", answer)
```

# Lab: 초등생을 위한 산수 문제 발생기

- 초등학생들을 위하여 산수 문제를 발생시키는 프로그램을 작성해보자. 한 번이라도 틀리면 반복을 중단한다.

```
25 + 78 = 103
잘했어요!!

3 + 32 = 37
틀렸어요. 하지만 다음번에는 잘할 수 있죠?
```

# Solution:

```
##
#          이 프로그램은 산수 문제를 출제한다.
#

import random

flag = True

while flag:
    x = random.randint(1, 100)
    y = random.randint(1, 100)
    answer = int(input(f"{x} + {y} = "))
    if answer == x + y:
        print("잘했어요!!")
    else:
        print("틀렸어요. 하지만 다음번에는 잘할 수 있죠?")
        flag = False
```

# 문자열 만들기

- str 합치기(concatenation) : +
- str.format( ) 함수
- f-strings
- %-서식출력(formatting) : %

```
name = "Kildong"
dept = "IT"
```

- **str concatenation(+)**

```
print("I am " + name + ". I belong to " + dept + " department.")
```

- **%-formatting (%)**

```
print("I am %s. I belong to %s department." % (name, dept))
```

- **str.format( )**

```
print("I am {}. I belong to {} department.".format(name, dept))
```

- **f-strings (최근 가장 인기)**

  **- 앞에 f 를 붙이고 {변수명} 형태로 작성. Python 3.6부터 지원**

```
print(f"I am {name}. I belong to {dept} department.")
```

# print()

- print(‘Hello World’)　　# 문자열 출력
- print(a)　　# 변수 값 출력
- print(‘계산 결과는’, result, ‘입니다’) # 문자열과 변수 값 출력
- print(100 + 100)　　# 연산 결과값 출력
- print(‘100’ + ‘100’).　　# 문자열 합치기
- print(‘값 : %d’ % sum)  # 서식 출력

# print() : 인수

- end : 마지막 출력 문자 지정 (기본값은 \n)
  - print(1,2, end=' ')
  - print(3,4)
- sep : 항목 간의 출력 문자 지정 (기본값은 공백)
  - print(1,2,3,4,5)
  - print(1,2,3,4,5, sep=',')
- file : 파일 객체로 출력 (기본값은 표준출력)
  - f = open('out.txt', 'w')
  - print(1,2,3,4,5, file=f)
  - f.close()
  - print(open('out.txt').read())

# print() – 서식 출력(1)

- 사용법
  - print(서식문자열 %(서식대응값))

| 서식 | 값의 예 | 설명 |
|---|---|---|
| %d, %x, %o | 10, 100, 1234 | 정수(10진수, 16진수, 8진수) |
| %f | 0.5 , 1.0 , 3.14 | 실수(소수점이 붙은 수) |
| %c | "b", "한" | 문자 한 글자 |
| %s | "안녕", "abcdefg", "a" | 한 글자 이상의 문자열 |

- 사용예
  - print('이름 : %s, 나이 : %d' %(name, age))

# print() – 서식 출력(2)

- 실습
  - data = 123
  - print('입력하신 값은 %5d 입니다' % data)
  - print('%05d' % data)
  - print('%d, %x, %o' % (data, data, data))

  - data = 123.45
  - print('%f' % data)
  - print('%10.1f' % data)
  - print('%-10.3f' % data)



```
>>> data = 123.45
>>> print('%f : 값 출력' % data)
>>> print('%10.1f : 값출력' % data)
>>> print('%-10.3f : 값 출력' % data)
```

  - data = 'Python'
  - print('%s' % data)
  - print('%10s' % data)

- name = '윤소정'
- age = 20
- print('%-10s:%10d' % (name, age))

# 다양한 특수 문자(2)

- 사용 예
  - print('\t 탭키 \t 연습')
  - print("I'm a student")
  - print('작은 따옴표 안에서 \' 사용하기\'! ')
  - print('\\\\\\ 역슬래시 3개 출력')
  - print(r'\n \t \' \\ 를 그대로 출력')

  - 실행 후 결과 확인!!!

  - 다시 복습...

```
파이썬의 세계에 오신 것을 환영합니다.
파이썬은 참 쉽고 강력합니다.
나는 파이썬이 좋습니다.
>>>
```

# Lab: 로그인 프로그램

- 사용자가 암호를 입력하고 프로그램에서 암호가 맞는 지를 체크한다고 하자.

```
암호를 입력하시오: 12345678
암호를 입력하시오: password
암호를 입력하시오: pythonisfun
로그인 성공
```

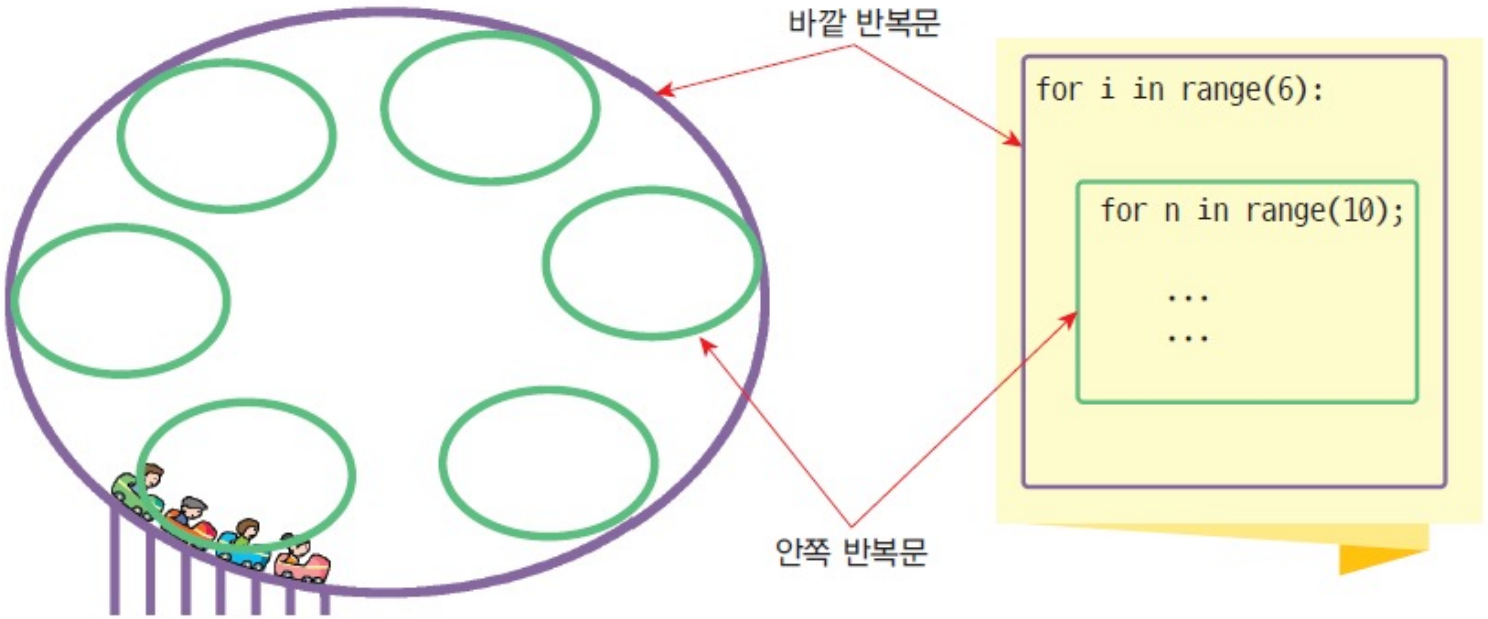
# Solution:

1. 암호=""
2. 암호가 "pythonisfun"이 아니면 다음을 반복한다.
    * 사용자로부터 암호를 입력받는다.
3. “로그인 성공”을 출력한다.

```
password = ""
while password != "pythonisfun":
    password = input("암호를 입력하시오: ")
print("로그인 성공")
```

# 중첩 반복문

- 프로그램에서도 반복 루프 안에 다시 반복 루프가 있을 수 있다.



# 예제 #1

```
for y in range(5) :
        for x in range(10) :
                print("*", end="" )
        print("")
```

```
**********
**********
**********
**********
**********
```

# 예제 #2

```
for y in range(1, 6) :
        for x in range(y) :
                print("*", end="" )
        print("")
```

```
*
**
***
****
*****
```

# 예제 #3: 모든 조합 구하기

```
small apple
small banana
small grape
medium apple
medium banana
medium grape
large apple
large banana
large grape
```

```
adj = ["small", "medium", "large"]
nouns = ["apple", "banana", "grape"]
for x in adj:
        for y in nouns:
                print(x, y)
```

# 중간점검

1. 다음 코드의 출력을 쓰시오.

```
for i in range(3) :
    for j in range(3) :
        print(i, "곱하기", j, "은", i*j)
```

# Lab: 주사위 합이 6이 되는 경우

- 주사위 2개를 던졌을 때, 합이 6이 되는 경우를 전부 출력하여 보자.

```
첫 번째 주사위=1 두 번째 주사위=5
첫 번째 주사위=2 두 번째 주사위=4
첫 번째 주사위=3 두 번째 주사위=3
첫 번째 주사위=4 두 번째 주사위=2
첫 번째 주사위=5 두 번째 주사위=1
```

# Solution:

```
##
#          이 프로그램은 주사위 2개의 합이 6인 경우를 전부 출력한다.
#
for a in range(1, 7) :
    for b in range(1, 7) :
        if a + b == 6 :
            print(f"첫 번째 주사위={a} 두 번째 주사위={b}" )
```

f-문자열이다. {a}라고 쓰면 변수 a의 값이 출력된다.

# Lab: 모든 조합 출력하기

```
persons = [ "Kim" , "Park" , "Lee" , "Choi" ]
restaurants = [ "Korean" , "American" , "French", "Chinese" ]
```

```
Kim이 Korean 식당을 방문
Kim이 American 식당을 방문
Kim이 French 식당을 방문
Park이 Korean 식당을 방문
Park이 American 식당을 방문
...
```

# Solution:

```
##
#          이 프로그램은 중첩 반복문을 사용하여서 모든 사람/식당 조합을 출력
한다.
#
persons = [ "Kim" , "Park" , "Lee"]
restaurants = [ "Korean" , "American" , "French"]

for person in persons:
        for restaurant in restaurants:
                print(person + "이 " + restaurant+" 식당을 방문")
```

# 무한 루프와 break, continue

Syntax: 무한루프 문

형식

```
while True :
    if 조건 :
        break           # 반복을 중단한다.
    if 조건 :
        continue        # 다음 반복을 시작한다.
```

# 예제

```
신호등 색상을 입력하시오 red
신호등 색상을 입력하시오 yellow
신호등 색상을 입력하시오 green
전진!!
```

```
while True:
    light = input('신호등 색상을 입력하시오 ‘)
    if light == 'green’:
        break
print( '전진!!')
```

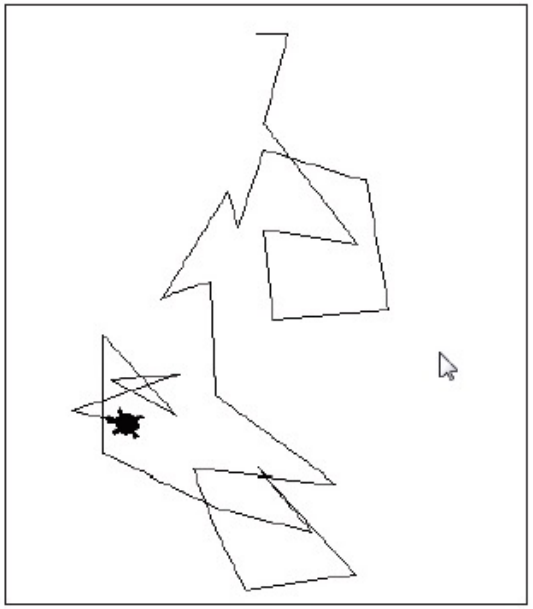
# 예제

```
for i in range(1, 11):
    if i%3 == 0 :              # 3의 배수이면
        continue               # 다음 반복을 시작한다.
    print(i, end=" ")          # 줄바꿈을 하지 않고 스페이스만 출력한다.
```

```
1 2 4 5 7 8 10
```

# Lab: 거북이 랜덤 워크

- 윈도우 상에서 거북이가 술에 취한 것처럼 랜덤하게 움직이게 하여 보자.

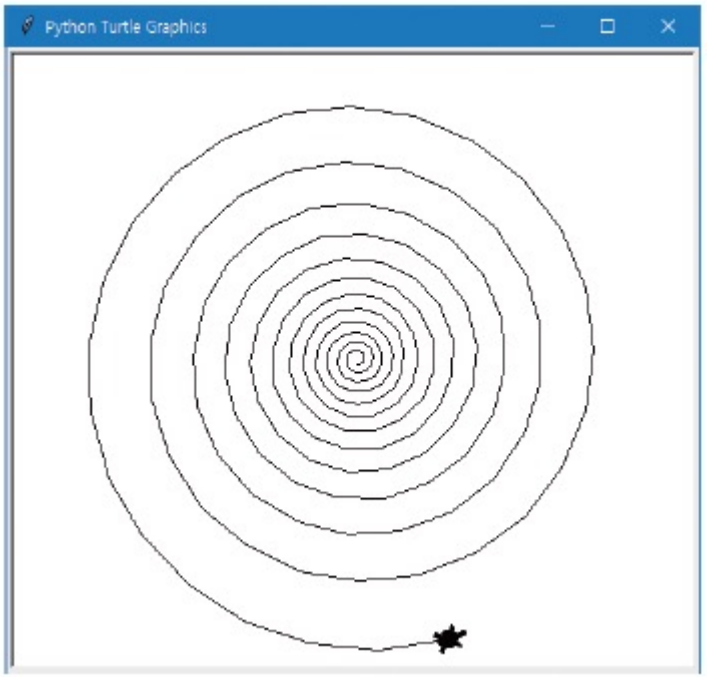


# Solution:

```
##
#          이 프로그램은 터틀 그래픽으로 랜덤 워크를 구현한다.
#
import turtle
import random
t = turtle.Turtle()
t.shape("turtle")

for i in range(30):
   length = random.randint(1, 100)
   t.forward(length)
   angle = random.randint(-180, 180)
   t.right(angle)

turtle.mainloop()
turtle.bye()
```

# Lab: 스파이럴 그리기

- 반복문을 터틀 그래픽과 결합하면 상당히 복잡한 형상을 쉽게 그릴 수 있다.



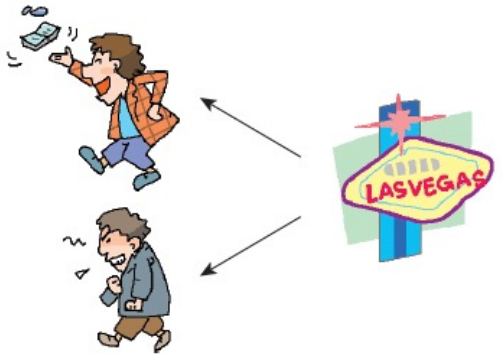
# Solution:

```
import turtle

t = turtle.Turtle()
t.shape("turtle")

for i in range(200):
        t.forward(2+i/4)          # 반복에 따라 조금씩 증가시킨다.
        t.left(30-i/12)           # 반복에 따라 조금씩 감소시킨다.

turtle.mainloop()
turtle.bye()
```

# Lab: 도박상의 확률

- 어떤 사람이 50달러를 가지고 라스베가스에서 게임을 한다고 하자. 한 번의 게임에 1달러를 건다고 가정하자. 돈을 딸 확률은 0.5이라고 가정하자. 한번 라스베가스에 가면, 가진 돈을 다 잃거나 목표 금액인 250달러에 도달할 때까지 게임을 계속한다. 어떤 사람이 라스베가스에 100번을 갔다면 몇 번이나 250달러를 따서 돌아올 수 있을까?

초기 금액 $50
목표 금액 $250
100번 중에서 25번 성공



# Solution:

```
import random

initial_money = 50
goal = 250
wins = 0

for i in range(100) :                    # 라스베가스에 100번 간다.
    cash = initial_money
    while cash > 0 and cash < goal :  # 돈이 0이거나 250불을 따면 반복 중단
        number = random.randint(1, 2)
        if number == 1 :
            cash = cash + 1              # $1을 딴다.
        else :
            cash = cash - 1              # $1을 잃는다.
    if cash == goal : wins = wins + 1

print("초기 금액 $%d" % initial_money)
print("목표 금액 $%d" % goal)
print("100번 중에서 %d번 성공" % wins)
```

# 이번 장에서 배운 것

- 문장들을 반복 실행하려면 for 문이나 while 문을 사용한다.
- 반복 실행되는 문장들을 들여쓰기 하여야 한다.
- for 문은 반복 회수를 정해져있을 때 유용하다.
- while 문은 반복 조건이 정해져 있을 때 유용하다.
- 반복문의 초입에서 조건식은 검사된다.

# Q & A