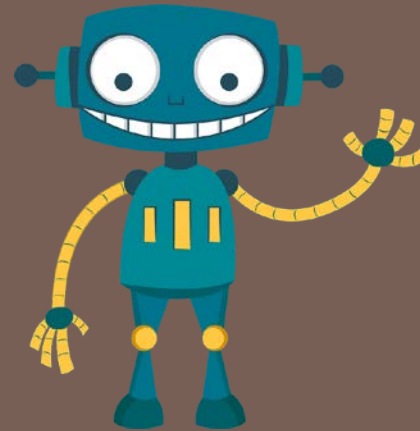
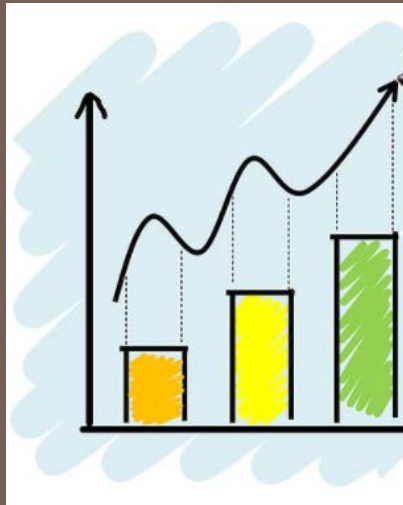


# 파이썬 익스프레스



## 4장 바보문

# 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

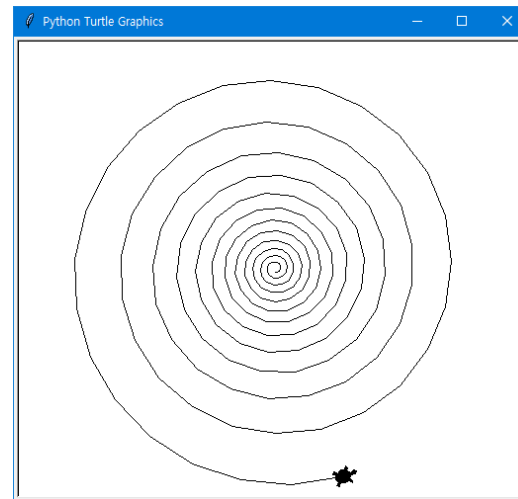
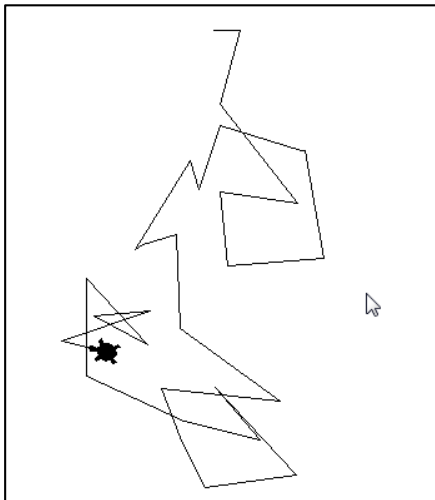
너무 낮음!

숫자를 입력하시오: 75

너무 낮음!

숫자를 입력하시오: 89

축하합니다. 시도횟수= 3



# 제어문

## 제어문

### 조건문

연봉이 2500 이상이면  
취업하고 아니면 고시  
준비해야지



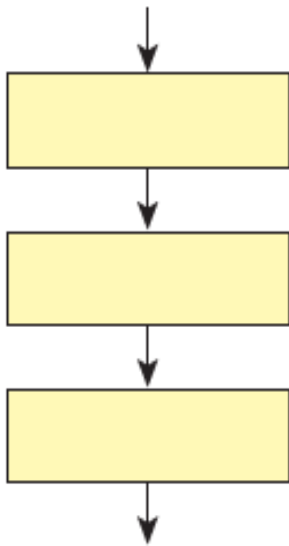
### 반복문

토익 점수가 600을  
넘을 때까지는  
반복해서 공부해야지

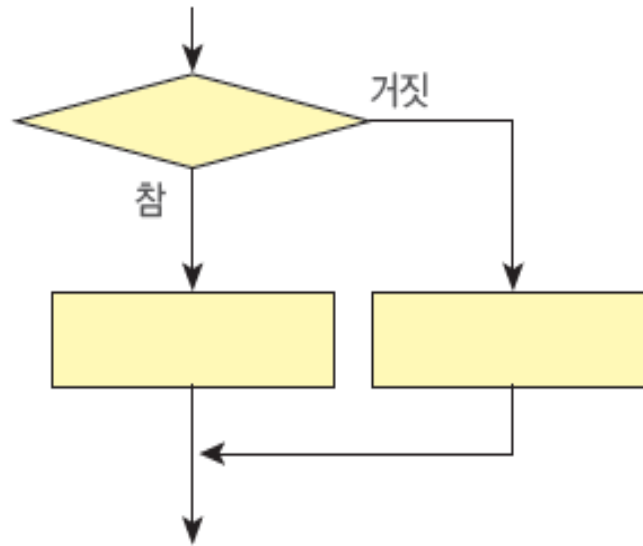


# 3가지의 제어구조

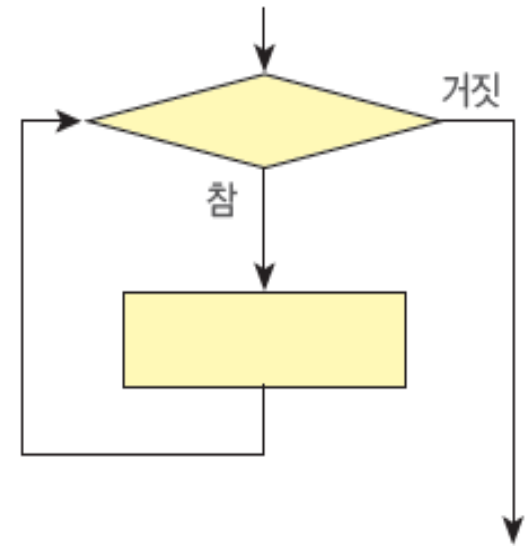
순차구조



선택구조

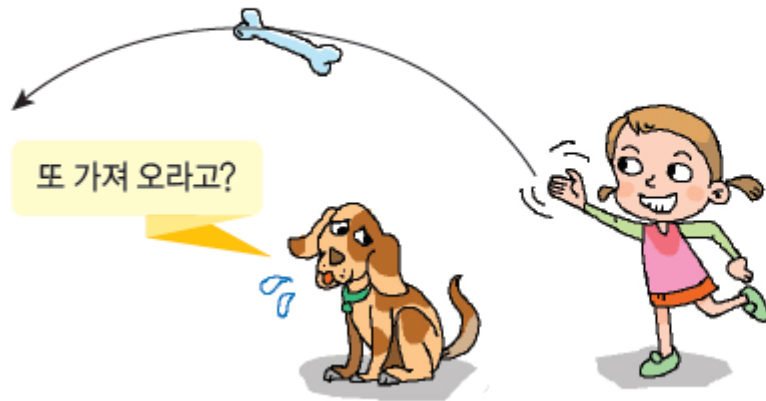


반복구조



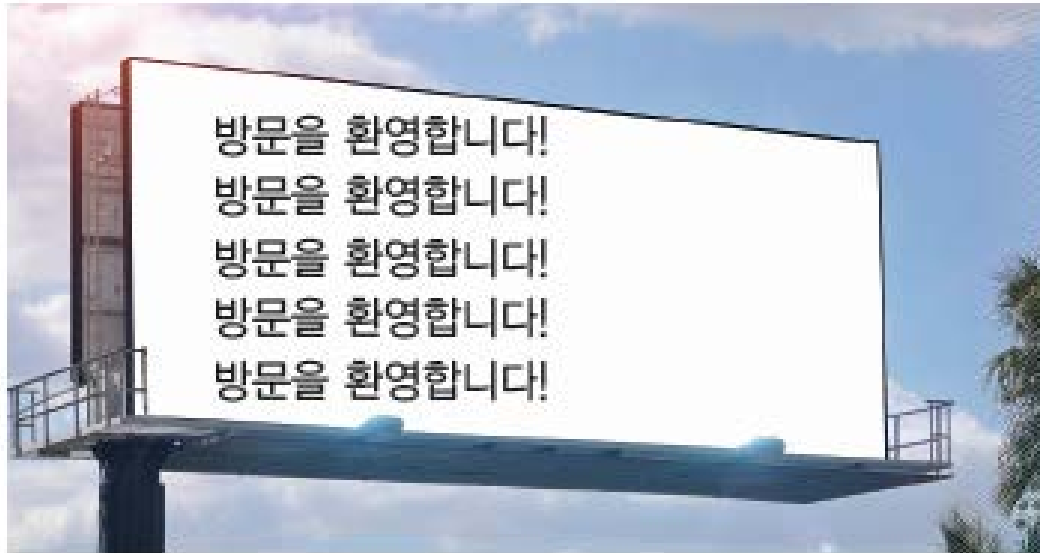
# 반복의 중요성

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



# 반복의 예

- 대형 전광판에 “방문을 환영합니다!”를 5번 출력해야 한다고 가정하자.

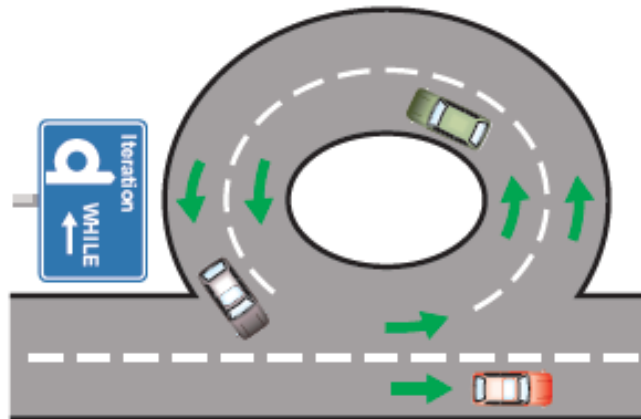


# 반복을 사용 vs 사용하지 않을 때

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

```
for i in range(1000):  
    print("방문을 환영합니다!")
```

# 아직 이해하지 않아도 된다!!





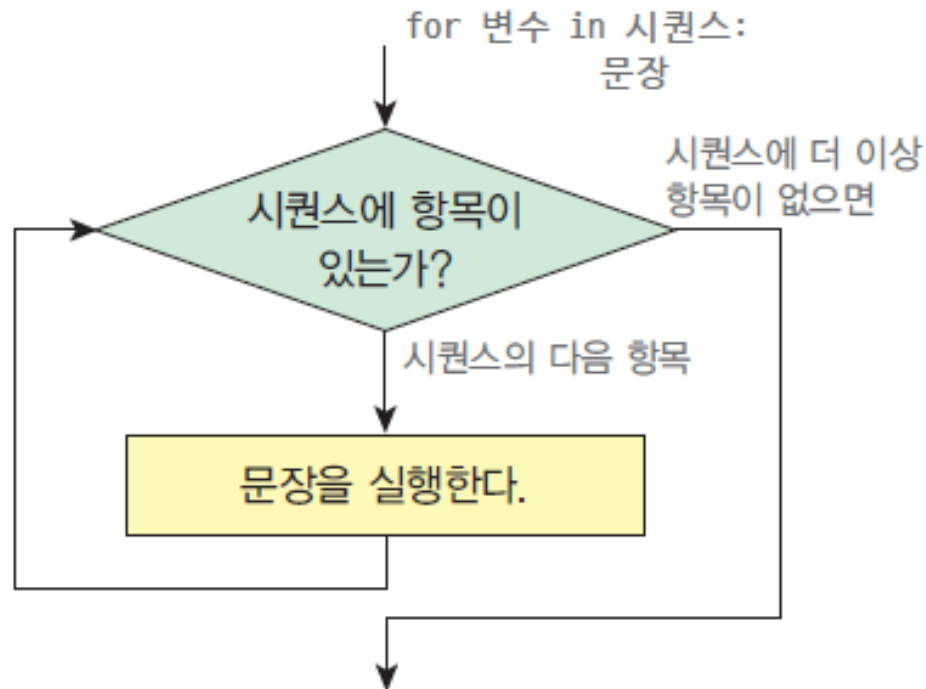
# 반복의 종류

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



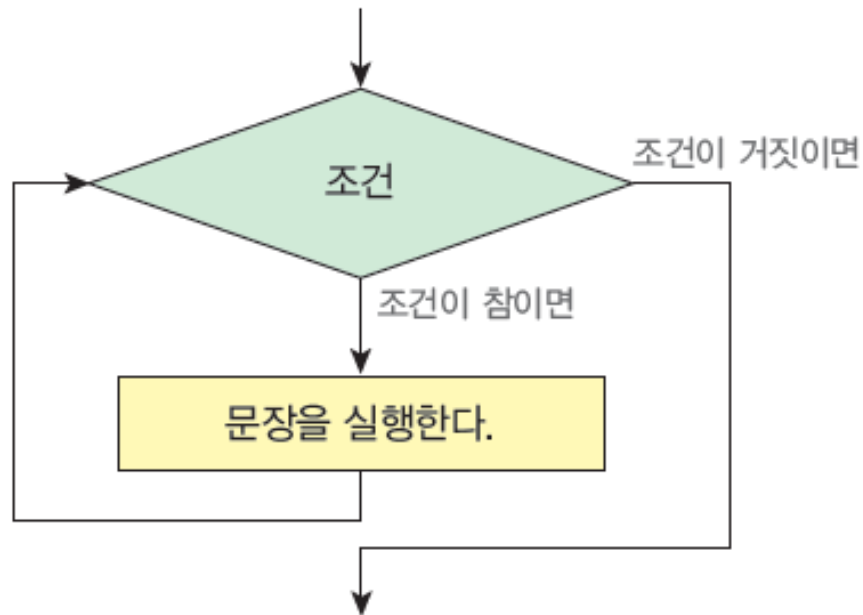
# 횟수 반복

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



# 조건 반복

- 조건 반복은 특정한 조건이 만족되는 동안 계속 반복한다.

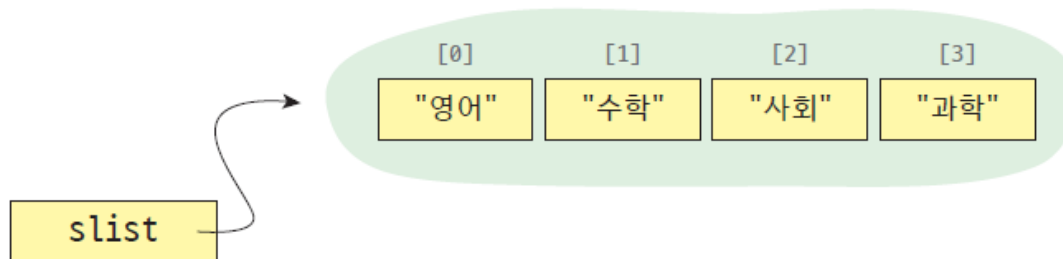


# 리스트(list)란?

- 항목들을 저장하는 자료 구조



```
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

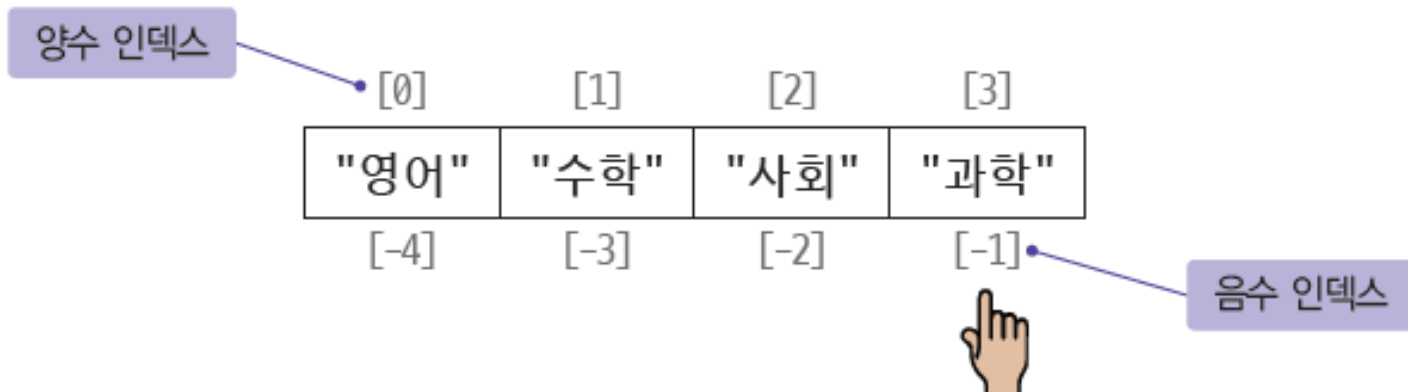
- 인덱스는 0부터 시작한다.
- 첫 번째 항목의 인덱스는 0이고 두 번째 항목의 인덱스는 1, 세 번째 항목의 번호는 2인 것이다.

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

```
>>> slist[0]
```

영어

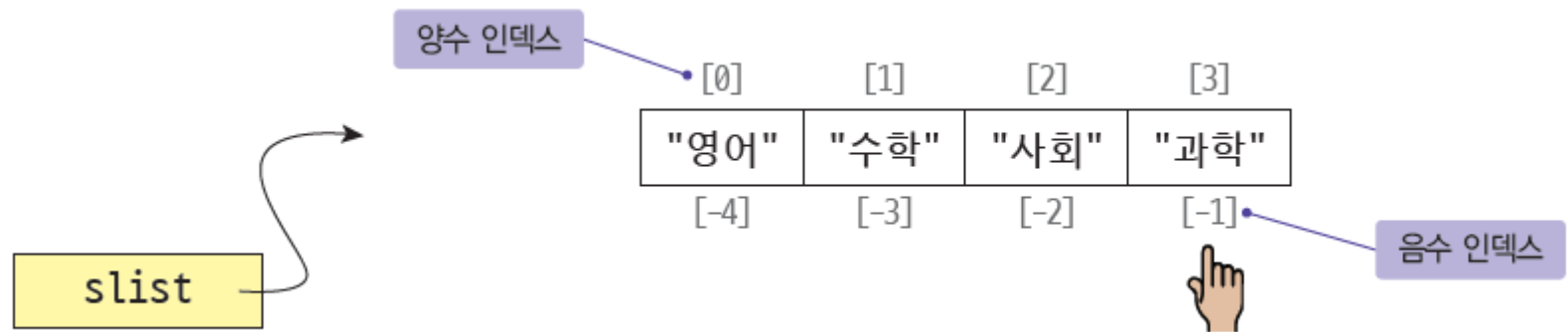
# 리스트의 첫 번째 항목을 출력한다.



# 리스트 항목 변경

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

```
['영어', '수학', '사회', '컴퓨터']
```



# 중간점검

- [ 1, 2, 3, 4, 5 ]를 저장하는 리스트 `myList`를 생성해보자.
- `myList`의 첫 번째 항목을 0으로 변경해보자.
- `myList`의 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 ] :

콜론(:)은 복합문을 의미한다.

리스트의 값들이  
하나씩 변수 i에  
할당되면서  
반복된다.

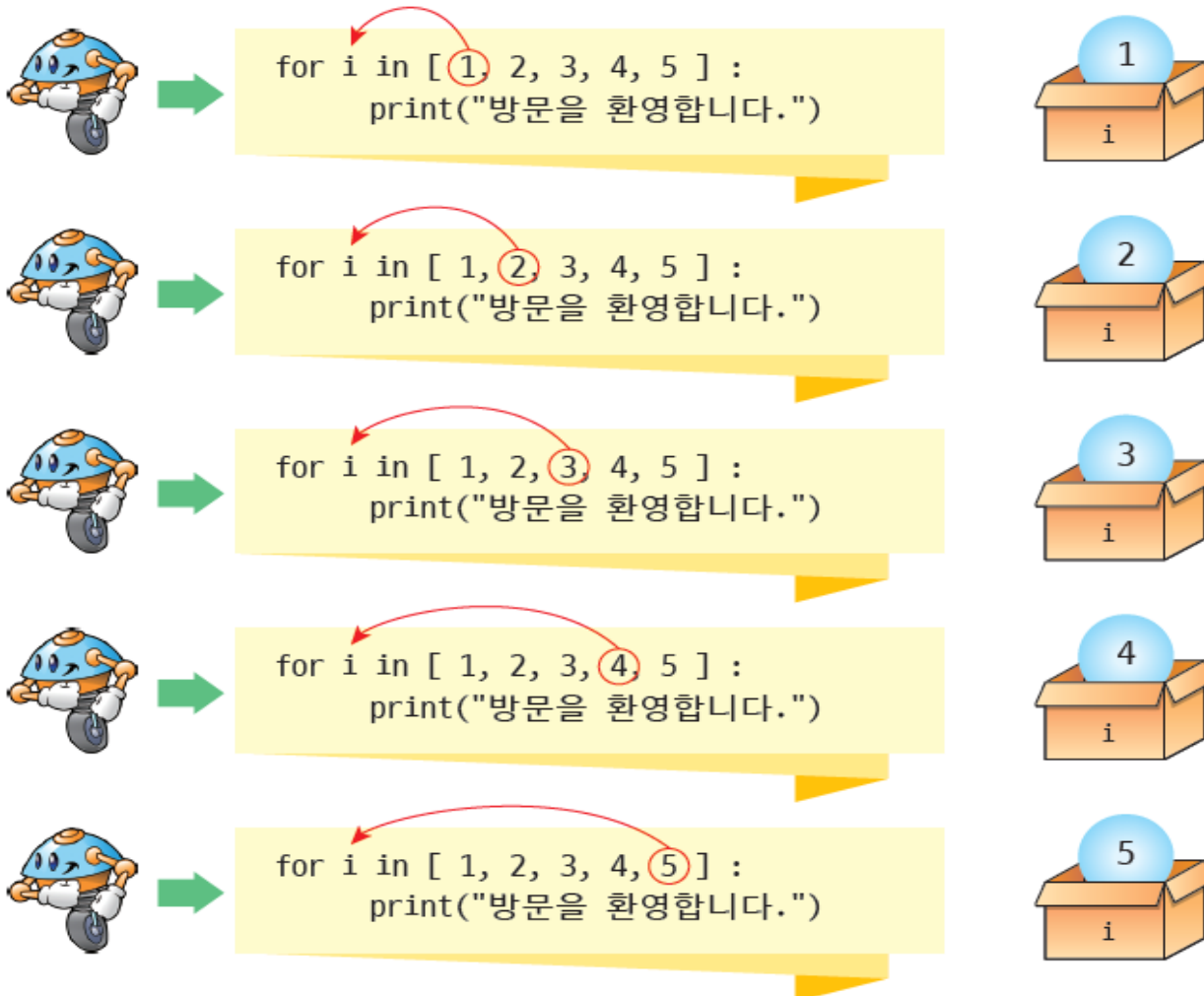
print("방문을 환영합니다.")

반복되는 문장은 동일하게  
들어쓰기가 되어야 한다.

반복되는 문장들

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

# 횟수 제어 반복의 이해



# 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

0, 1, 2, 3, 4의 값들이 생성  
되어서 변수 i에 할당된다.

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

반복되는 문장으로 들여쓰기 하여야 한다.

# range() 함수의 이해

Syntax: range() 함수

**형식** range(start=0, stop, step=1 )

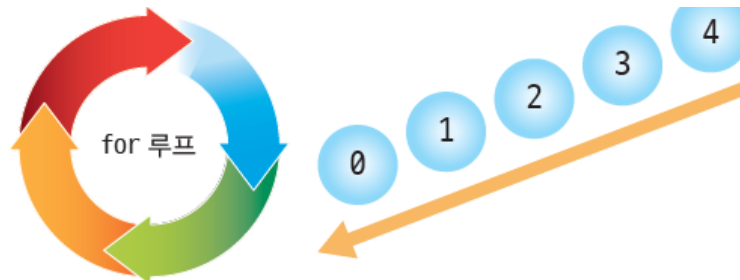
시작값이다.

종료값이지만 stop은  
포함되지 않는다.

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

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))
```



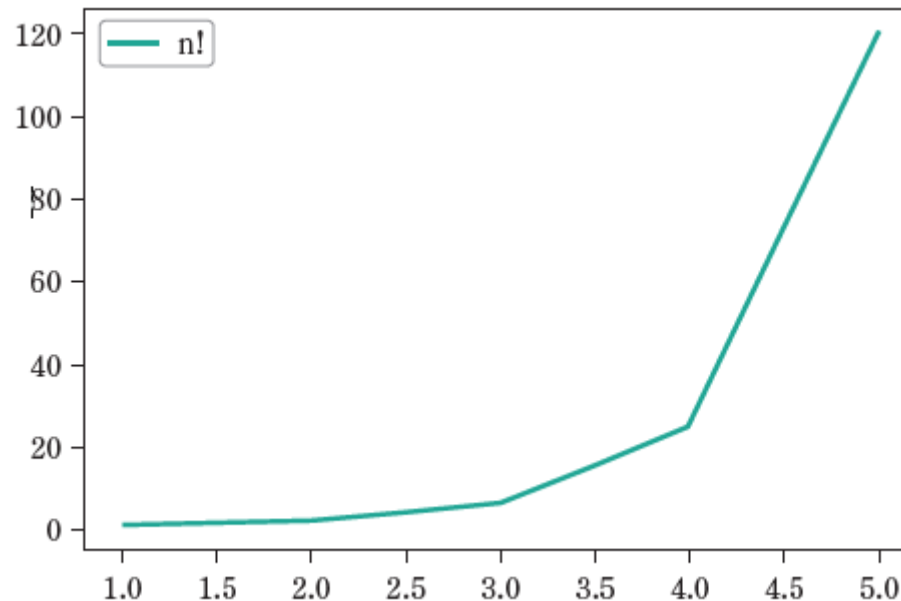


# Lab: 팩토리얼 계산하기

**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.

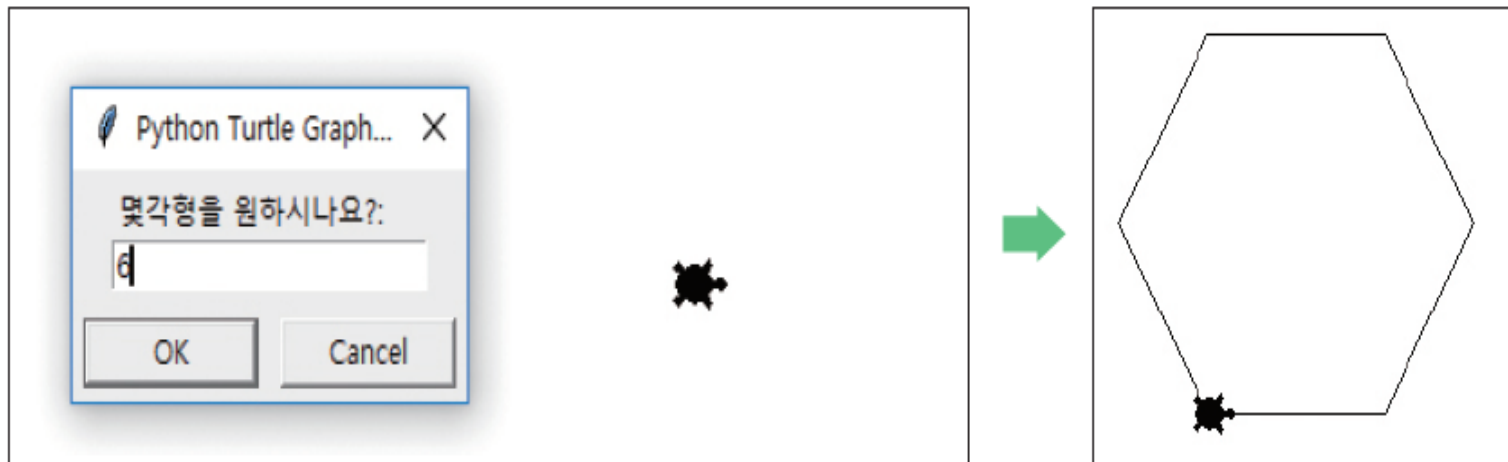
$$n! = n \times (n-1) \times \dots \times 2 \times 1$$

앞의 프로그램을 어떻게 수정하여야 하는가?



# Lab: $n$ -각형 그리기

- 터틀 그래픽에서 사용자로 부터 정수  $n$ 을 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)

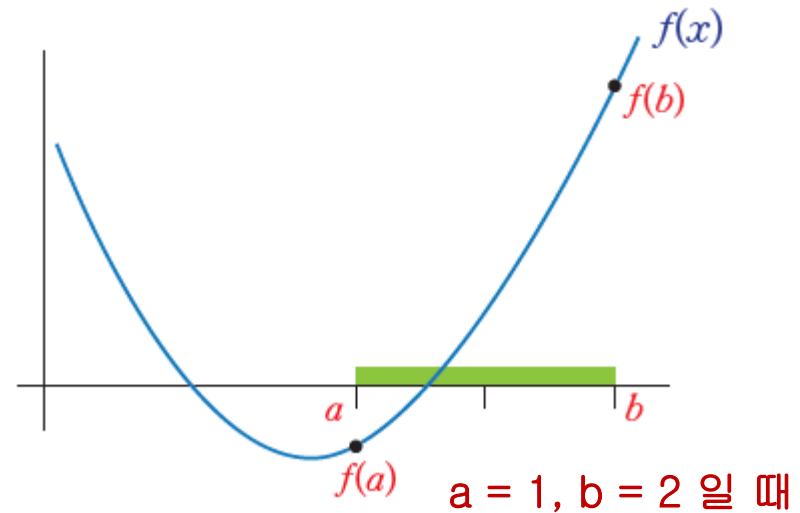
for i in range(n):
    t.forward(100)
    t.left(360/n)

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

# Lab: 방정식의 해 구하기

- 3차 방정식이나 4차, 5차 방정식인 경우에 어떻게 해를 계산할 것인가? 이런 경우에 사용할 수 있는 방법이 수치 해석적인 방법이다. 즉 프로그램으로 해를 계산하는 것이다. 프로그램은 매우 빠르게 반복할 수 있다.

$$f(x) = x^2 - x - 1$$



방정식의 해는 1.62

# Lab: 방정식의 해 구하기

```
COUNT = 100
START = 1.0      # a값
END = 2.0        # b값

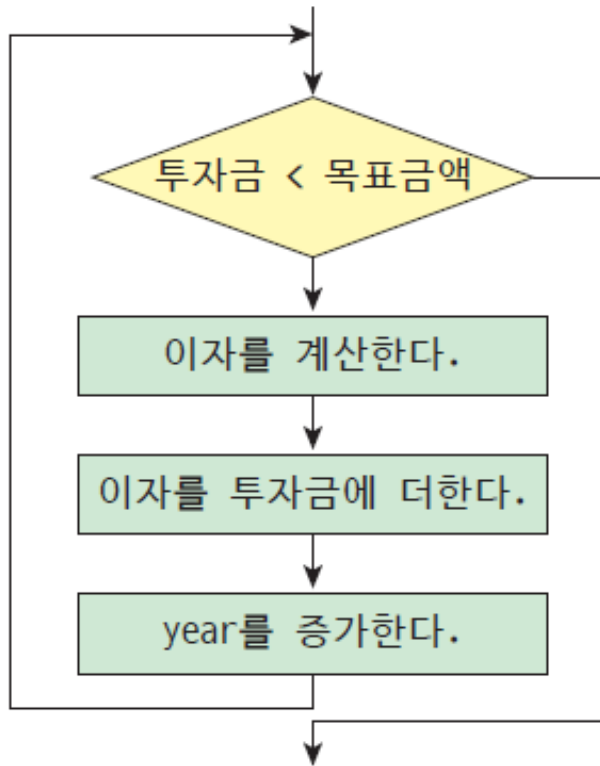
for i in range(COUNT):
    x = START + i * ((END-START)/COUNT)    # x값을 0.01씩 증가시킴
    f = x**2 - x - 1
    if abs(f-0.0) < 0.01 :                  # f(x)의 값이 거의 0이 되는 x값 구하기
        print("방정식의 해는 ", x)
```

- (1) if  $\text{abs}(f-0.0) < 0.0001$  :으로 변경하여 보자. 또 COUNT 상수의 값을 100000으로 설정하여 보자. 어떤 일이 발생하는가?
- (2) 이분법(bisection) 방법을 인터넷에서 조사해보고 이분법을 구현해보자.



# 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.

예

while money < TARGET :

콜론(:)은 복합문을 의미한다.

money = money + money \* rate

year = year + 1

반복되는 문장은 동일하게  
들여쓰기가 되어야 한다.

조건이 참이면 반복되는 문장들

# 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

# Lab: 투자금이 2배가 되는데 걸리는 시간

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

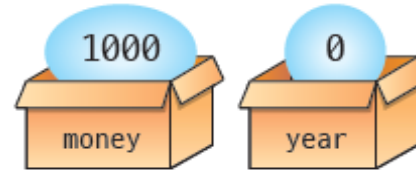
# 현재 금액이 목표 금액보다 작으면 반복한다.
while money < TARGET :
    money = money + money * rate
    year = year + 1

print(year, "년")
```

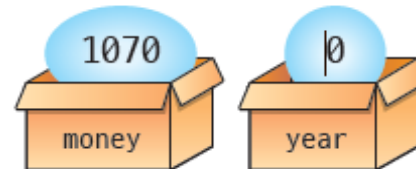
11 년

# 조건 제어 반복

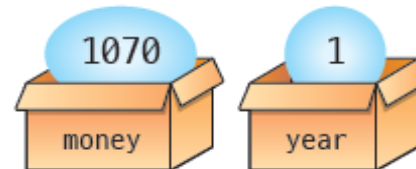
→ while money < TARGET :  
    money = money + money\*rate  
    year = year + 1



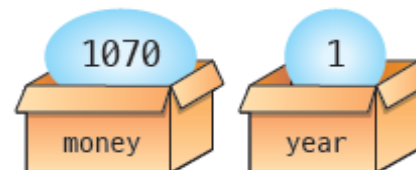
→ while money < TARGET :  
    money = money + money\*rate  
    year = year + 1



→ while money < TARGET :  
    money = money + money\*rate  
    year = year + 1



→ while money < TARGET :  
    money = money + money\*rate  
    year = year + 1



# else가 있는 while 루프

```
i = 0
```

```
while i < 3:
```

```
    print("루프 안쪽")
```

```
    i = i + 1
```

```
else:    # 조건식이 False가 되면 else절 실행됨, break는 실행 안됨
```

```
    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.  $9*1$ ,  $9*2$ ,  $9*3$ , ...,  $9*9$ 까지 9번 반복하여 출력하면 된다.

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("%d * %d = %d" % (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 입력된 정수가 더 높은지 낮은지를 알려준다.

1부터 100 사이의 숫자를 맞추시오

숫자를 입력하시오: **50**

너무 낮음!

숫자를 입력하시오: **75**

너무 낮음!

숫자를 입력하시오: **89**

축하합니다. 시도횟수= 3

# Lab: 숫자 맞추기 게임

```
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 다음 번에는 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 다음 번에는 잘할 수 있죠?")  
        flag = False
```

# Lab: 로그인 프로그램

- 사용자가 암호를 입력하고 프로그램에서 암호가 맞는 지를 체크한다고 하자.

암호를 입력하시오: **12345678**

암호를 입력하시오: **password**

암호를 입력하시오: **pythonisfun**

로그인 성공

# Solution:

# 알고리즘

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

```
password = ""
```

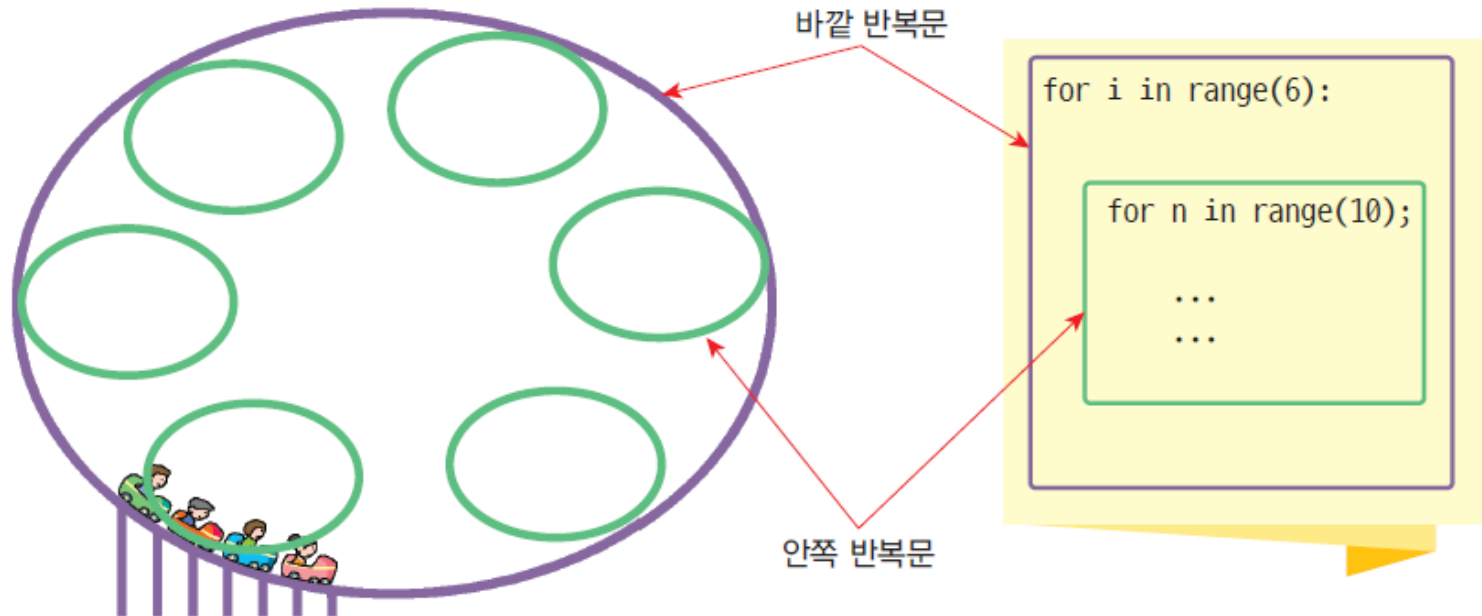
```
while password != "pythonisfun":
```

```
    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

- 식당에서 식사하는 사람의 모든 조합을 출력해보자.

```
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 사용하여 모든 사람/식당 조합을 출력한다.  
#  
  
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 반복 시작  
    print(i, end=" ")      # 줄바꿈을 하지 않고 스페이스만 출력
```

1 2 4 5 7 8 10

# Lab: 소수 찾기

- 2부터 시작하여 50개의 소수를 찾는 프로그램을 작성해보자.

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101  
103 107 109 113  
127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223  
227 229

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

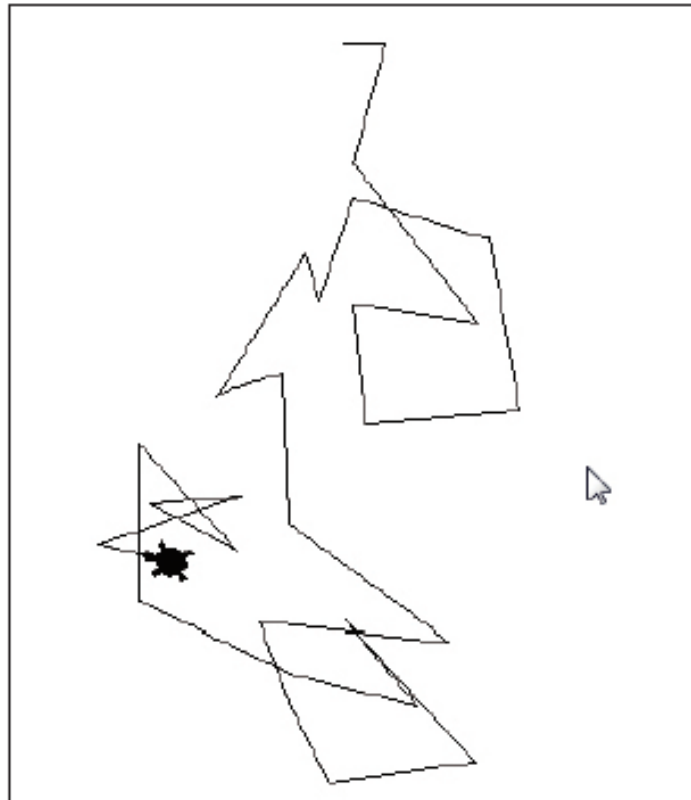
# Solution:

```
N_PRIMES = 50          # 찾아야 하는 소수의 개수
number = 2             # 2부터 시작한다.
count = 0

while count < N_PRIMES :
    divisor = 2         # 나누는 수는 2부터 시작하여 하나씩 증가한다.
    prime = True
    while divisor < number :
        if number % divisor == 0: # 나누어지면 소수가 아니다.
            prime = False
            break
        divisor += 1
    if prime:           # 소수이면 소수 개수를 증가하고 출력한다.
        count += 1
        print(number, end=" ")
    number += 1         # 다음 수로 간다.
```

# Lab: 거북이 랜덤 워크

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

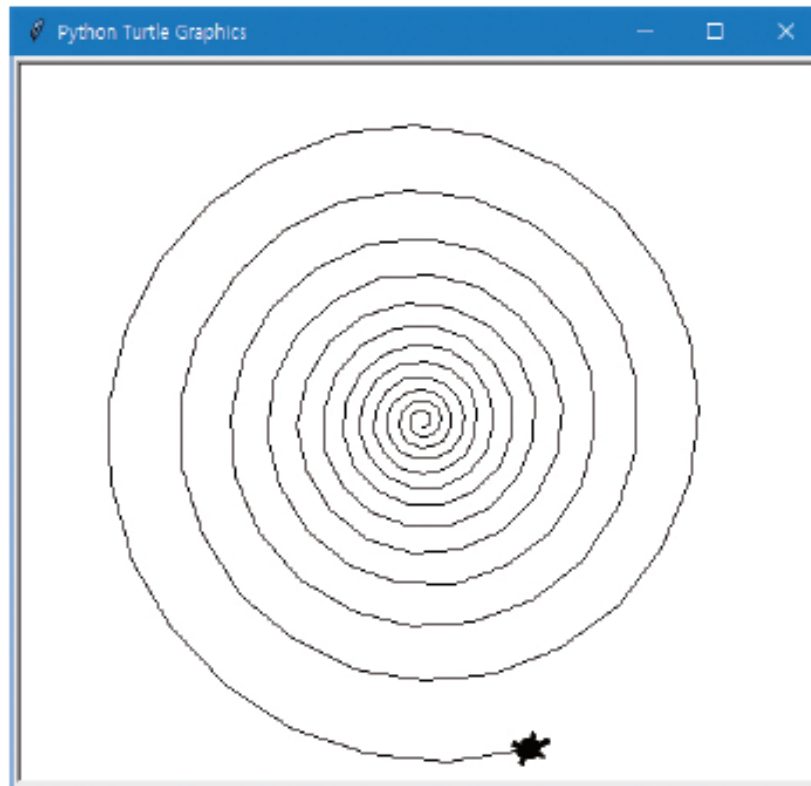


# Solution:

```
##  
#       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.

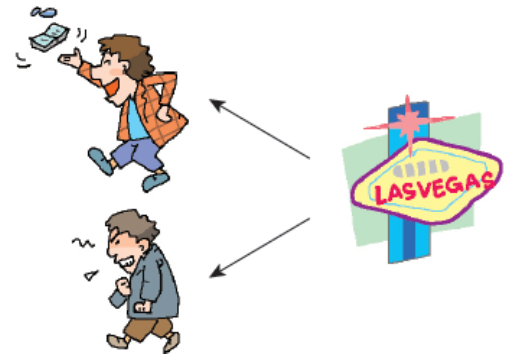
```
turtle.mainloop()
```

```
turtle.bye()
```

# Lab: 도박상의 확률

- 어떤 사람이 50달러를 가지고 라스베가스에서 게임을 한다고 하자. 한 번의 게임에 1달러를 건다고 가정하자. 돈을 딸 확률은 0.5라고 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 따다.
        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 {:d}번 성공" % wins)
```

# 이번 장에서 배운 것

- 문장들을 반복 실행하려면 **for 문**이나 **while 문**을 사용한다.
- 반복 실행되는 문장들을 **들여쓰기** 하여야 한다.
- **for 문**은 반복 회수를 정해져 있을 때 유용하다.
- **while 문**은 반복 조건이 정해져 있을 때 유용하다.
- 반복문의 초입에서 조건식은 검사된다.

