새내기 파이썬

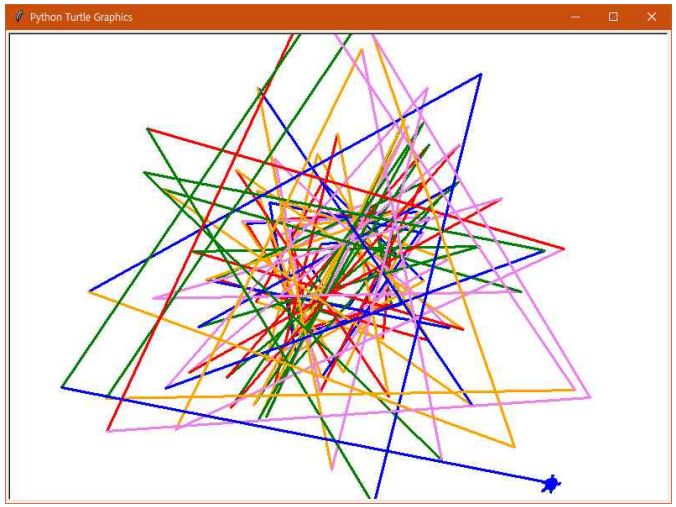
5장 반복문

학습 목표

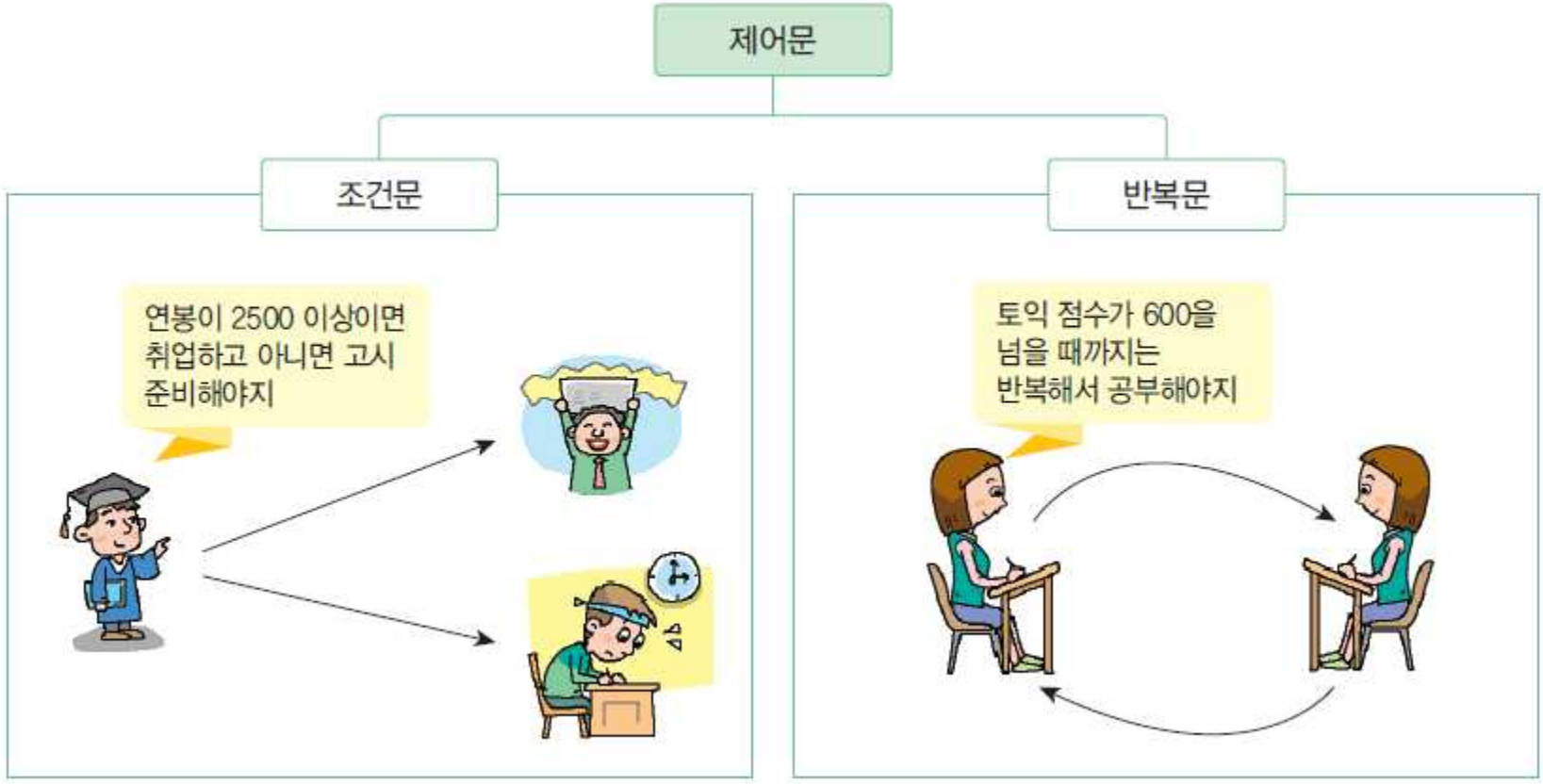
- 원하는 회수만큼 특정한 문장을 반복할 수 있나요?
- 리스트의 모든 요소에 대하여 반복할 수 있나요?
- 어떤 조건이 만족될 때까지 반복할 수 있나요?
- 반복문 안에 반복문을 넣을 수 있나요?
- 무한 반복문을 만들 수 있나요?

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

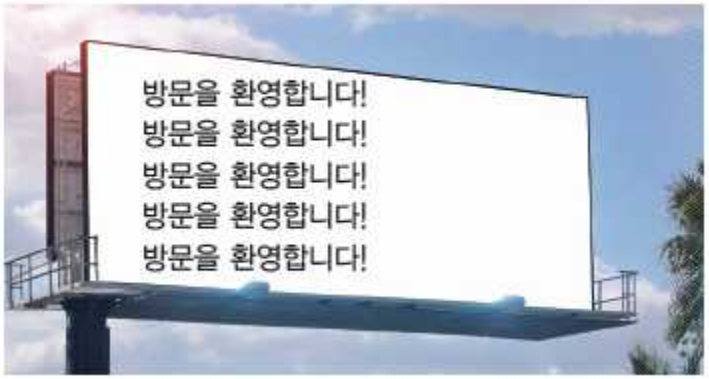

제어문
조건문
반복문
연봉이 2500 이상이면
취업하고 아니면 고시
준비해야지
토익 점수가 600을
넘을 때까지는
반복해서 공부해야지

반복의 중요성

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

반복의 예

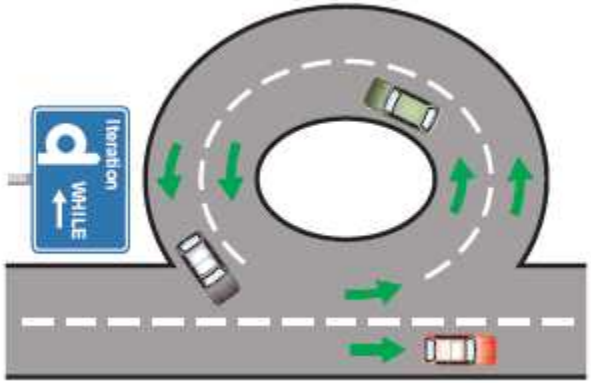
- 화면에 회사에 중요한 손님이 오셔서 대형 전광판에 “방문을 환영합니다!”를 5번 출력해야 한다고 가정하자.



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(5):                                  # 아직 이해하지 않아도 된다!!
        print("방문을 환영합니다!")
```



반복의 종류

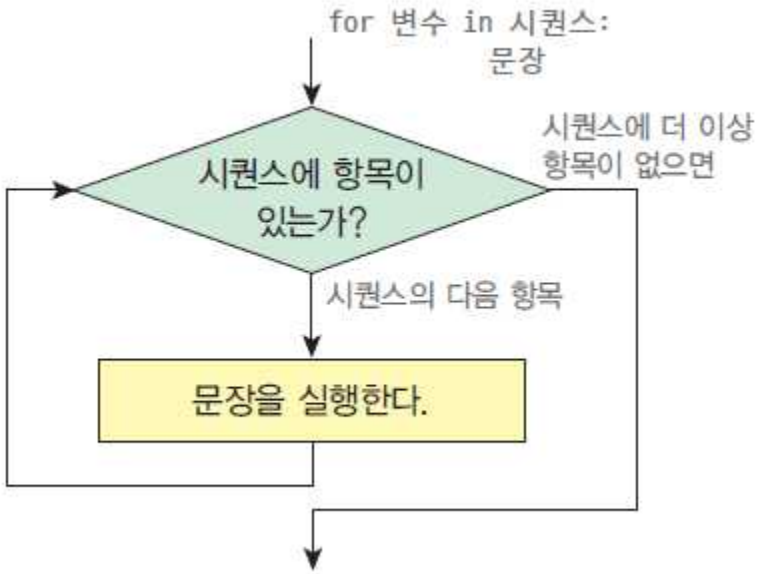
- 횟수 반복(for 문): 정해진 횟수만큼 반복한다.
- 조건 반복(while 문): 특정한 조건이 성립되는 동안 반복한다.

조건 반복

횟수 반복

횟수 반복

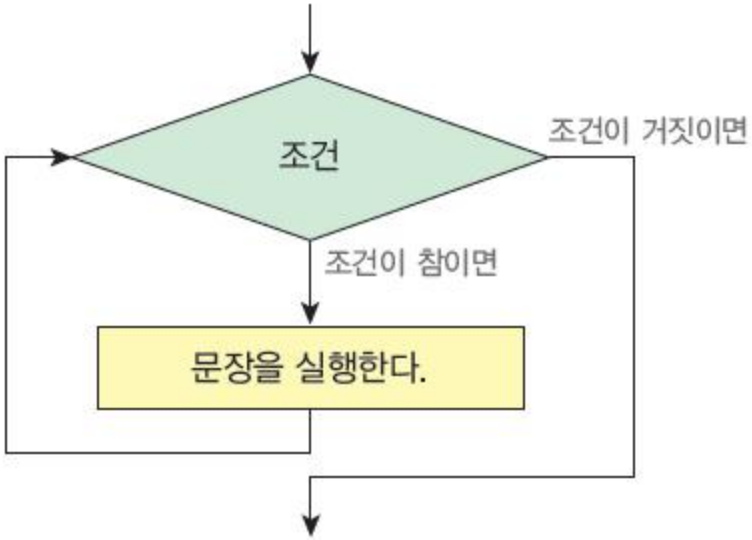
- 횟수 반복은 반복을 시작하기 전에 반복의 횟수를 미리 아는 경우에 사용한다.



횟수 반복

조건 반복

- 조건 반복은 특정한 조건이 만족되는 동안 계속 반복한다.



조건 반복

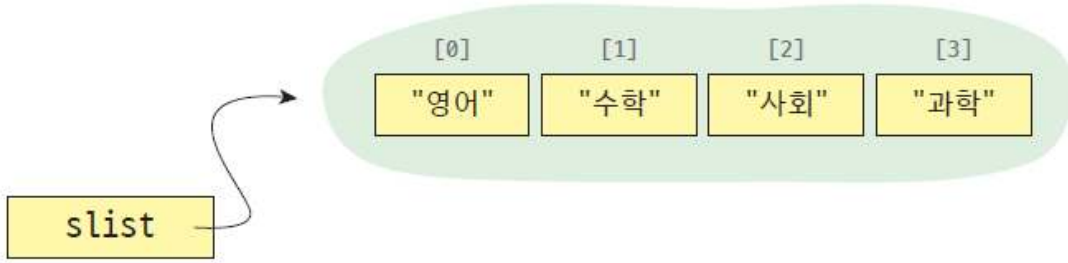
중간점검

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

리스트란?

- 항목들을 저장하는 자료 구조


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



리스트에 동적으로 항목 추가

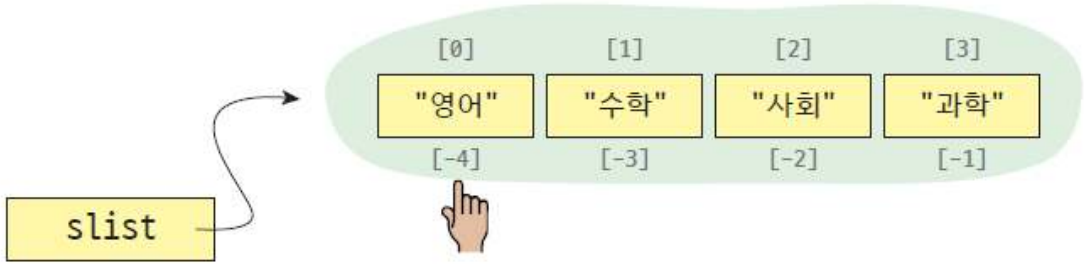
```
list = []                       # 공백 리스트를 생성한다.
list.append(1)                  # 리스트에 정수 1을 추가한다.
list.append(2)                  # 리스트에 정수 2을 추가한다.
list.append(6)                  # 리스트에 정수 6을 추가한다.
list.append(3)                  # 리스트에 정수 3을 추가한다.

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.

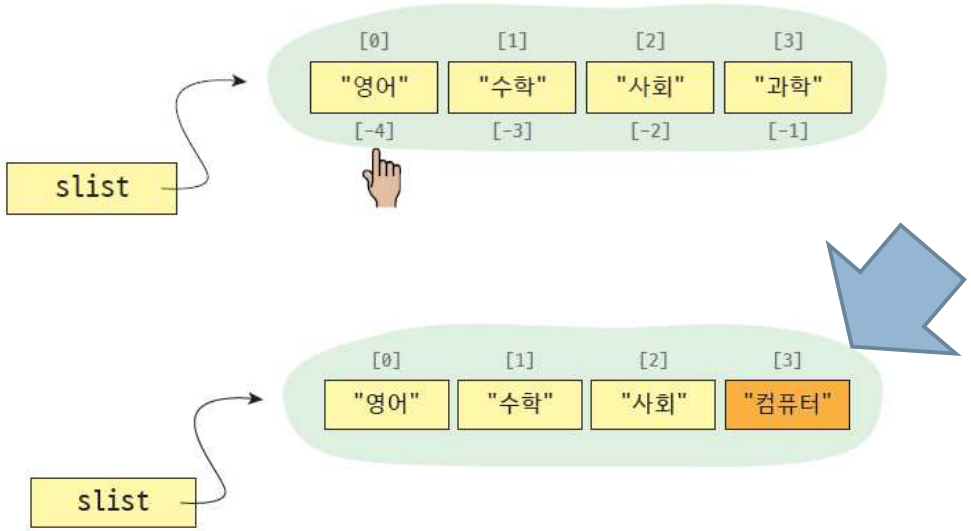
```
>>> slist = [ "영어", "수학", "사회", "과학" ]
>>> slist[0]
영어                                    # 리스트의 첫 번째 항목을 출력한다.
```



리스트 항목 변경

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

```
['영어', '수학', '사회', '컴퓨터']
```



중간점검

1. [1, 2, 3, 4, 5]를 저장하는 리스트 myList를 생성해보자.
2. myList의 첫 번째 항목을 0으로 변경해보자.
3. 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.")
```

콜론(:)은 복합문을 의미한다.

리스트의 값들이 하나씩 변수 i에 할당되면서 반복된다.

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

1. 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.

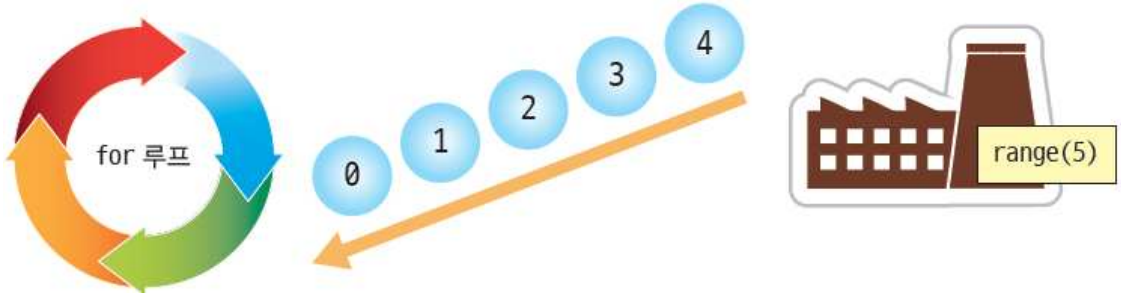
range() 함수의 이해

range(start=0, stop, step=1)

시작값이다.

종료값이지만 stop은 포함되지 않는다.

한 번에 증가되는 값이다.



예제

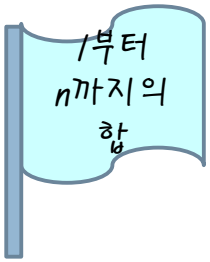

```
for i in range(1, 6, 1):
        print(i, end=" ")
```

```
1 2 3 4 5
```


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

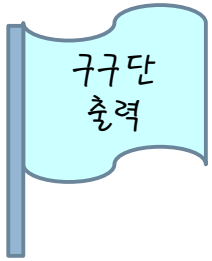
```
10 9 8 7 6 5 4 3 2 1
```

예제




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

```
합= 55
```




```
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
```

Example: 1부터 n까지의 합 계산 프로그램

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

print("합=", sum)
```

```
합= 55
```

Example: 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)
```

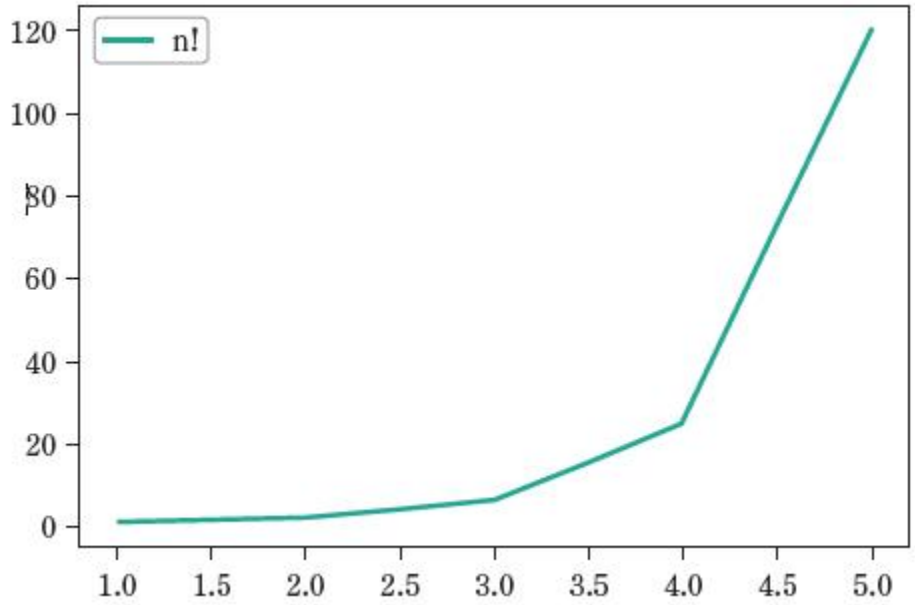
2. 다음 코드의 출력을 쓰시오.

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

Lab: 팩토리얼 계산하기

for문을 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

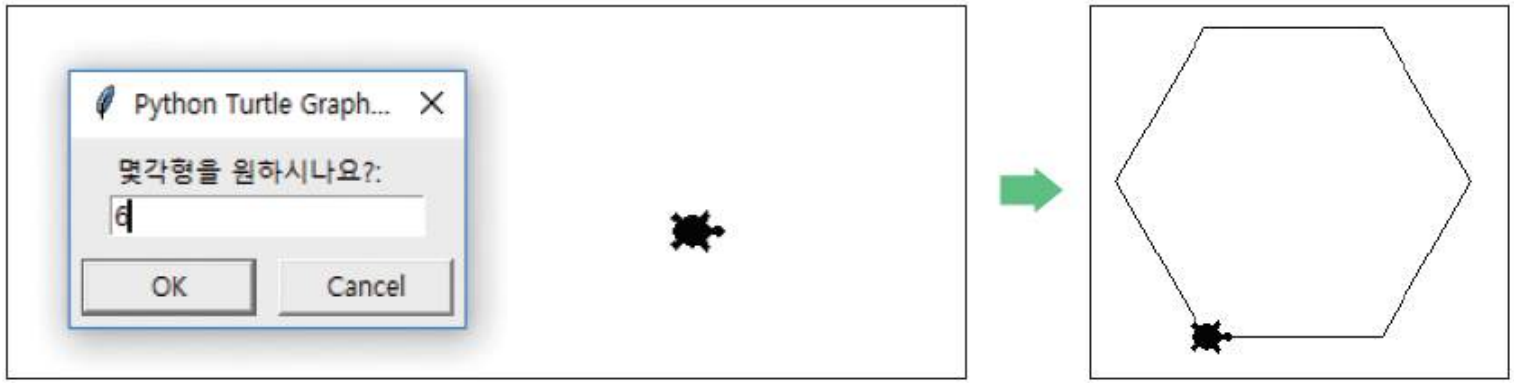
1. 팩토리얼을 거꾸로 계산해보자.

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

앞의 프로그램을 어떻게 수정하여야 하는가?

Lab: n-각형 그리기

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

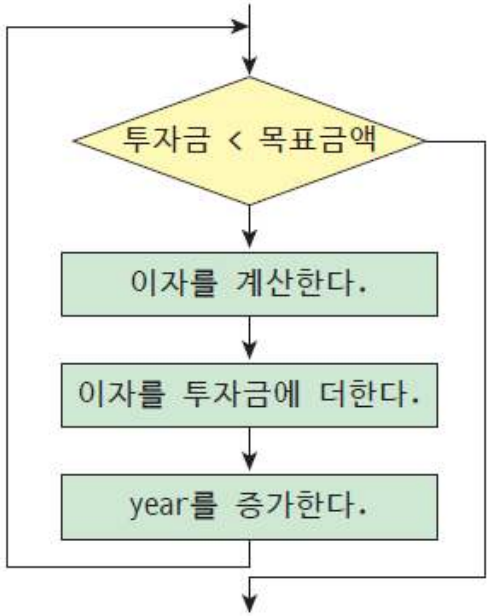


Lab: n-각형 그리기

```
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.done()
```

조건 제어 반복

- 조건 제어 반복은 어떤 조건이 만족 되는 동안 반복한다.
- (예) 투자금이 목표에 도달하는 기간을 계산해보자.



조건 제어 반복

Syntax: while 문

형식

```
while 조건식 :
        문장1
        문장2
```

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

예

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

콜론(:)은 복합문을 의미한다.

조건이 참이면 반복되는 문장들

반복되는 문장은 동일하게
들여쓰기가 되어야 한다.

Example: 투자금을 2배로 만드는 기간 프로그램

```
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
```

Example: 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
```

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: 사용자가 입력하는 숫자의 합 계산하기

- 사용자가 입력한 숫자들을 더하는 프로그램을 작성해보자. 사용자가 yes라고 답한 동안에만 숫자를 입력받는다.

```
숫자를 입력하시오:  10
계속?(yes/no): yes
숫자를 입력하시오:  20
계속?(yes/no): no
합계는 :  30
```

Solution:

```
total = 0
answer = "yes"                                  # 반복이 수행되려면 answer의 초기값은 "yes"
while answer == "yes":                          # answer가  "yes"인지를 검사한다.
        number = int(input("숫자를 입력하시오:  "))
        total = total + number
        answer = input("계속?(yes/no): ")
print("합계는 : ", total)
```

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
```

Lab: 로그인 프로그램

- 사용자가 암호를 입력하고 프로그램에서 암호가 맞는 지를 체크한다고 하자.

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

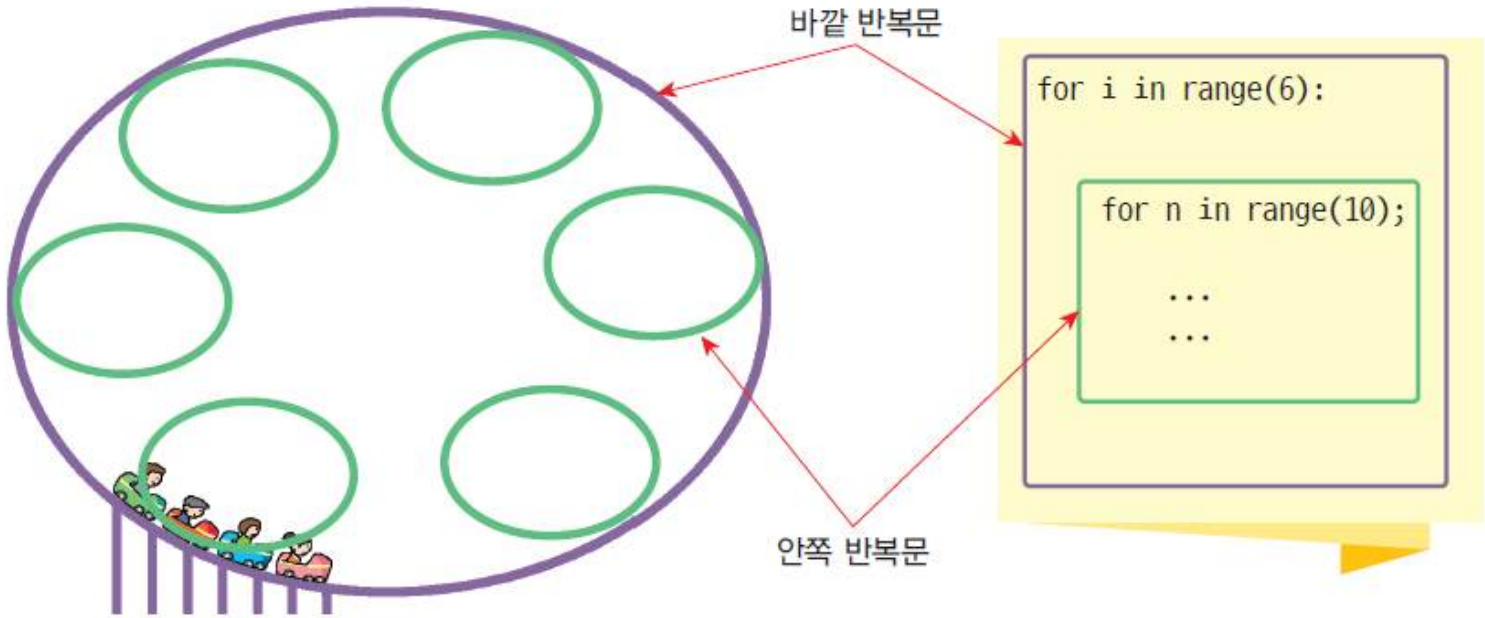
Solution:

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.



Example: 사각형 패턴 출력하기

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

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

Example: 삼각형 패턴 출력하기

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

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

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

```
coffee = [ "Americano" , "Latte" , "Mocha" ]
cake = [ "CheeseCake" , "StrawberryCake", "ChocolateCake" ]
```

```
Americano CheeseCake
Americano StrawberryCake
Americano ChocolateCake
Latte CheeseCake
Latte StrawberryCake
Latte ChocolateCake
Mocha CheeseCake
Mocha StrawberryCake
Mocha ChocolateCake
```

Solution:

```
##
#          이 프로그램은 중첩 반복문을 사용하여서 모든 조합을 출력한다.
#
coffee = [ "Americano" , "Latte" , "Mocha" ]
cake = [ "CheeseCake" , "StrawberryCake", "ChocolateCake" ]

for co in coffee:
    for ca in cake:
        print(co + " " + ca)
```

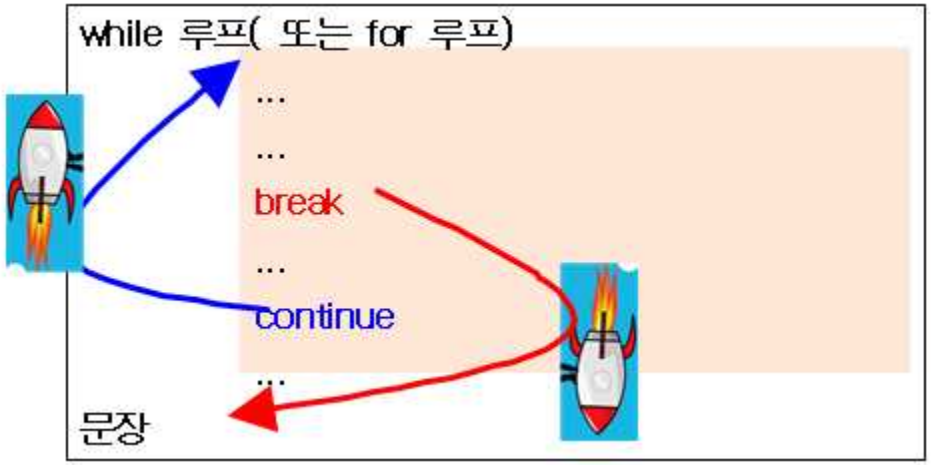
무한 루프와 break, continue

Syntax: 무한루프 문

형식

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

무한 루프와 break, continue



Example: 신호등 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!!')
```

무한루프는 언제 사용하는가?

- 무한 루프는 실제 코드에서 많이 사용된다. 특히 반복을 빠져나가는 조건이 까다로운 경우에 많이 사용된다. 예를 들어서 사용자가 입력한 수가 3의 배수이거나 음수인 경우에 while 루프를 빠져나가야 한다고 하자. 이때는 아래 그림의 왼쪽과 같이 while 루프의 조건문을 만드는 것보다, 오른쪽처럼 무한 루프를 만들고 그 안에서 루프를 벗어나는 조건을 검사하는 편이 이해하기 쉽다.

```
while (x%3 == 0) or (x < 0) :
    ...
    ...
    ...
```

```
while True:
    if x%3 == 0: break
    if x<0 : break
    ...
```

Example: 3의 배수만 빼고 출력하기

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

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

Lab: 파이 계산하기

- 파이를 계산하는 것은 무척 시간이 많이 걸리는 작업으로 주로 수퍼 컴퓨터의 성능을 시험하는 용도로 사용된다. 지금은 컴퓨터의 도움으로 10조개의 자리수까지 계산할수 있다. 파이를 계산하는 가장 고전적인 방법은 Gregory-Leibniz 무한 수열을 이용하는 것이다.

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

```
반복횟수:10000
Pi = 3.141493
```

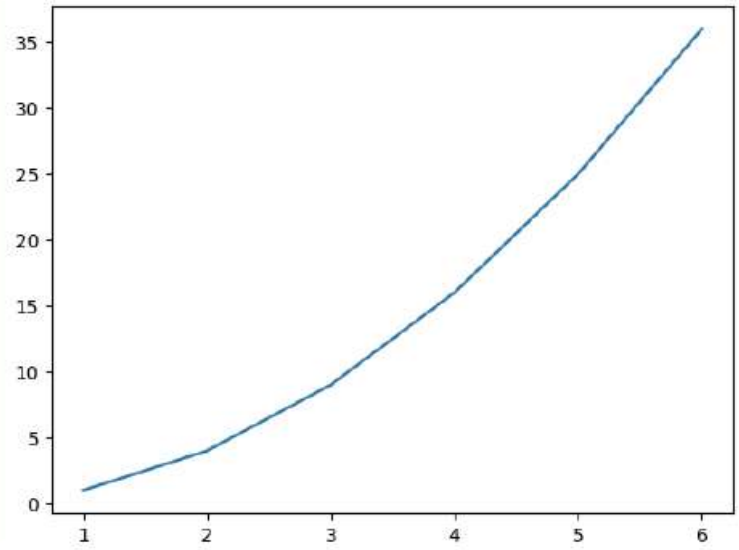
Solution:

```
divisor = 1.0
divident = 4.0
sum = 0.0
loop_count = int(input("반복횟수:"))

while(loop_count > 0) {
        sum = sum + divident / divisor
        divident = -1.0 * divident
        divisor = divisor + 2
        loop_count = loop_count - 1;
print("Pi = %f" % sum)
```

Lab: 그래프 그리기

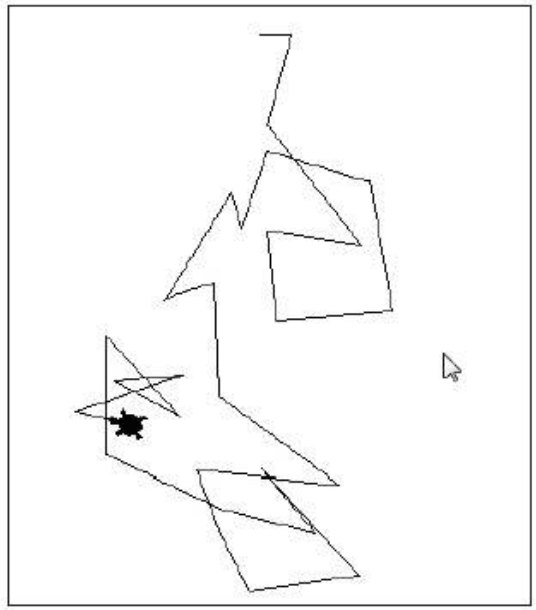
- 리스트에 저장된 데이터를 이용하여서 간단한 그래프를 그려보자. 그래프를 그리는 라이브러리는 Matplotlib이다.

```
import matplotlib.pyplot as plt

X = [1, 2, 3, 4, 5, 6]
Y = [1, 4, 9, 16, 25, 36]

plt.plot(X, Y)
plt.show()
```



Lab: 거북이 랜덤 워크

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

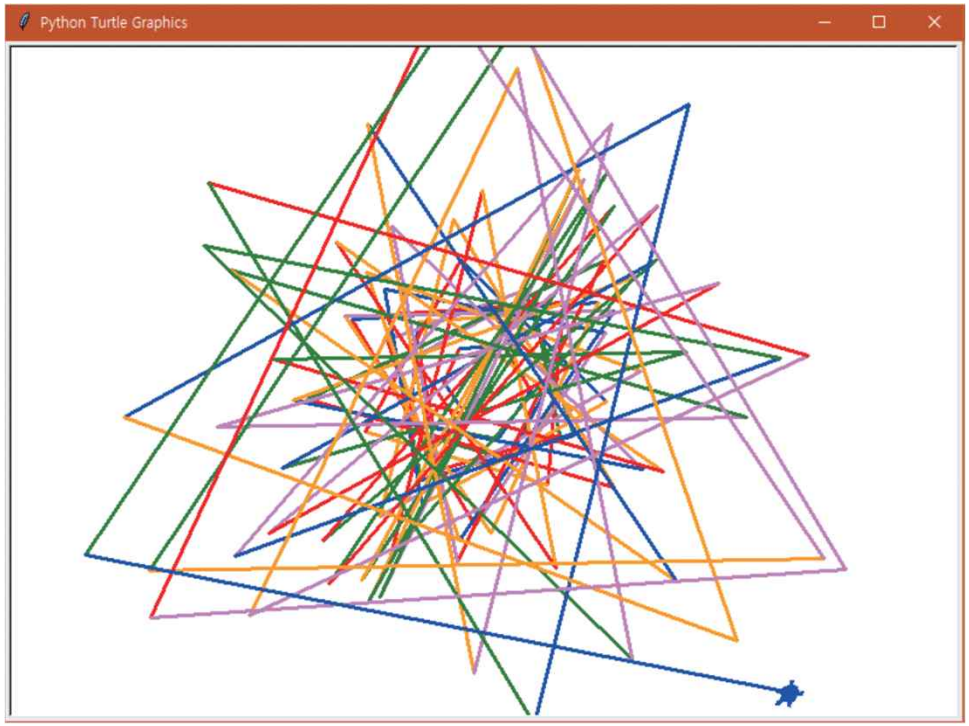


Solution:

```
##
#          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.done()
```

Lab: 스파이럴 그리기

- 색상을 변경하면서 거북이를 랜덤한 방향으로 전진시키고 회전시켜 보자.



Solution:

```
import turtle
import random

t = turtle.Turtle()
t.shape('turtle')
t.width(3)
t.speed(7)
colors = ['orange', 'red', 'violet', 'green', 'blue']

for i in range(100):
    t.color(random.choice(colors))          # 리스트 중에서 랜덤하게 하나를 선택한다.
    t.right(20 + i)
    t.forward(1 + (i * 6))
    t.right(30 + i)

turtle.done()
```

Mini Project: 사용자의 숫자 맞추기

- 우리는 앞에서 컴퓨터가 숨기고 있는 숫자를 사람이 맞추는 프로그램을 살펴보았다. 이번에는 반대로 하여 보자. 사용자가 숨기고 있는 숫자를 컴퓨터가 알아맞히는 프로그램을 작성해보자. 게임 도중에 사용자는 숫자를 변경하면 안 된다.

```
컴퓨터가 당신이 생각하는 숫자를 알아맞히는 게임입니다.
하나의 숫자를 생각하세요.
컴퓨터가 제시한 숫자보다 정답이 높으면 high, 낮으면 low라고 하세요.
컴퓨터가 숫자를 맞추면 yes라고 하세요.

숫자가 50 인가요? low
숫자가 25 인가요? high
숫자가 37 인가요? high
숫자가 42 인가요? yes
```

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