파이썬 Express

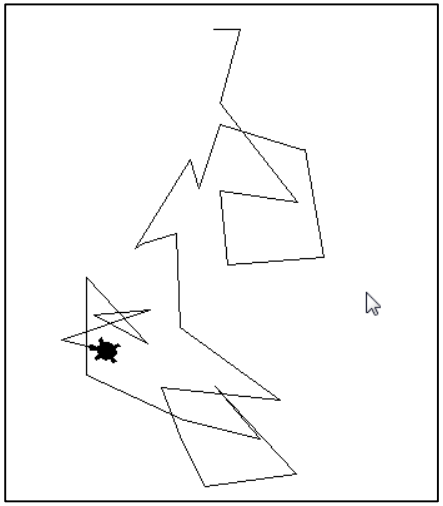
4장 반복문

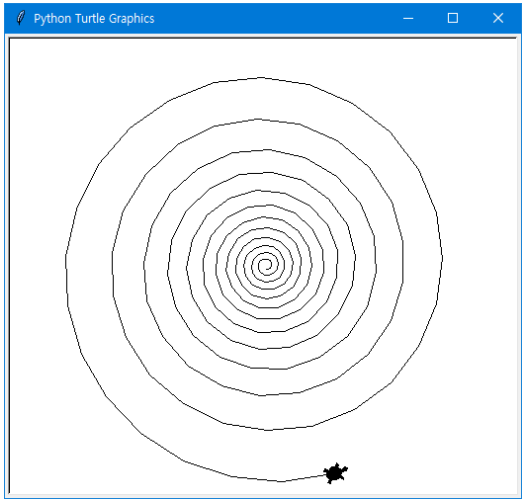
학습 목표

- 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.

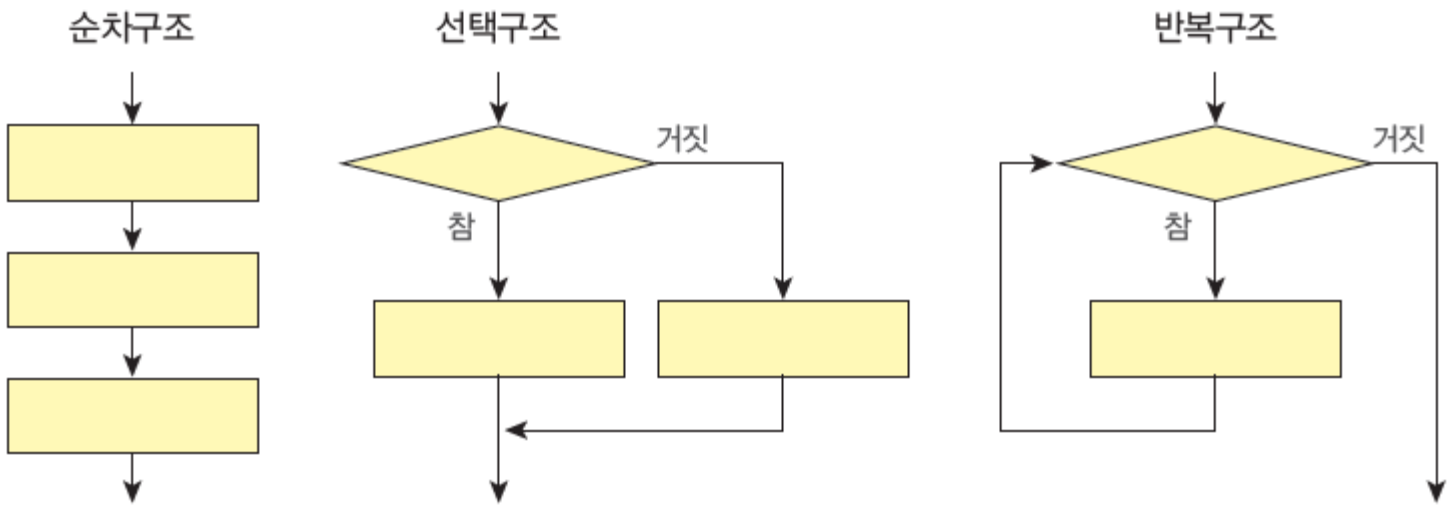
이번 장에서 만들 프로그램

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



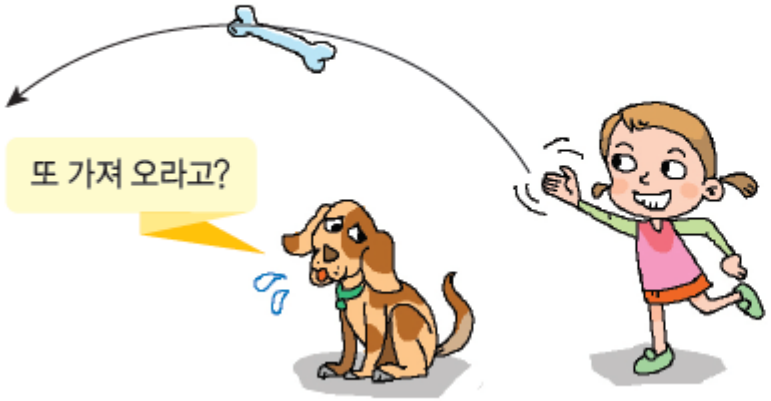


3가지의 제어구조



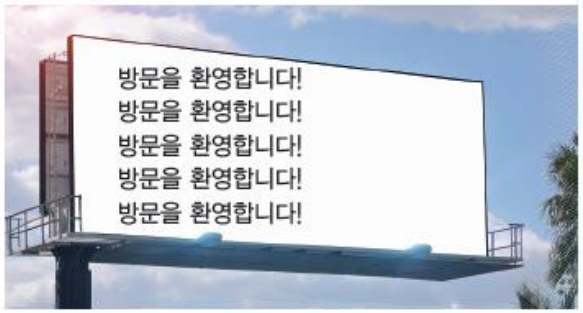
왜 반복이 중요한가

- 인간은 똑같은 작업을 반복하는 것을 지루해한다. 반복적이고 단순한 작업은 컴퓨터를 이용하여 자동화하면 된다. 동일한 작업을 오류 없이 반복하는 것은 컴퓨터가 잘할 수 있는 일이다.



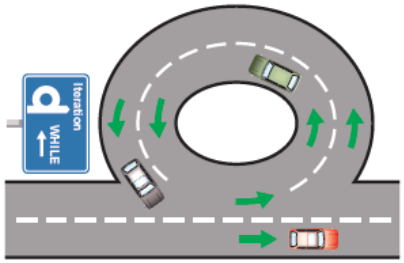
반복의 예

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




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

```
for i in range(1000):                # 아직 이해하지 않아도 된다!!
    print("방문을 환영합니다!")
```



반복의 종류

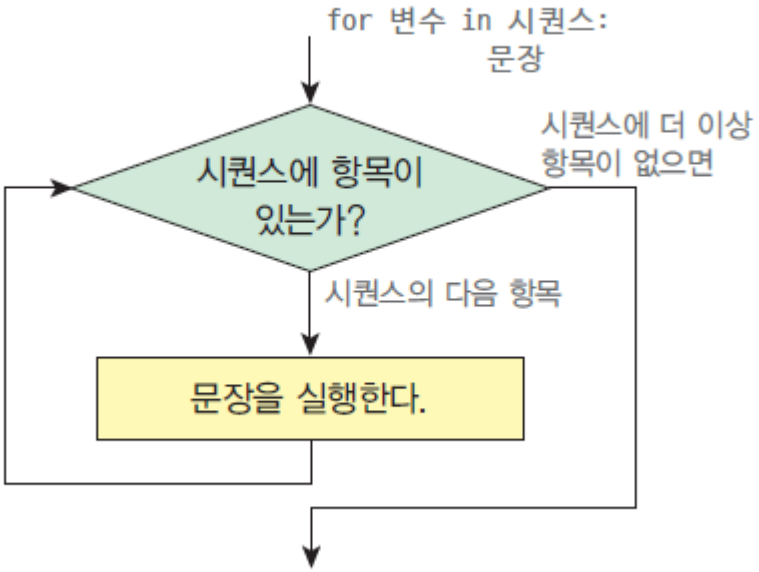
- 횟수 반복(for 문): 정해진 횟수만큼 반복한다.
- 조건 반복(while 문): 특정한 조건이 성립되는 동안 반복한다.

조건 반복

횟수 반복

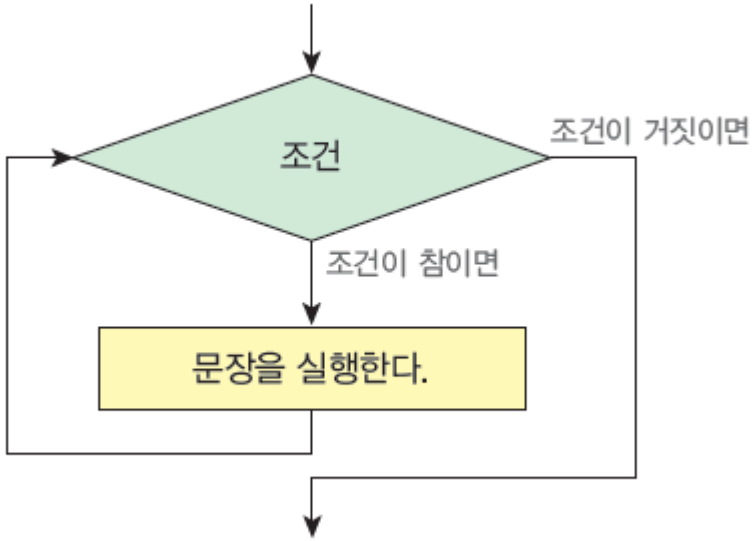
횟수 반복

- 횟수 반복은 반복을 시작하기 전에 반복의 횟수를 미리 아는 경우에 사용한다.
- 예. “환영합니다“ 문장을 10번 반복하여 출력



조건 반복

- 조건 반복은 특정한 조건이 만족되는 동안 계속 반복한다.



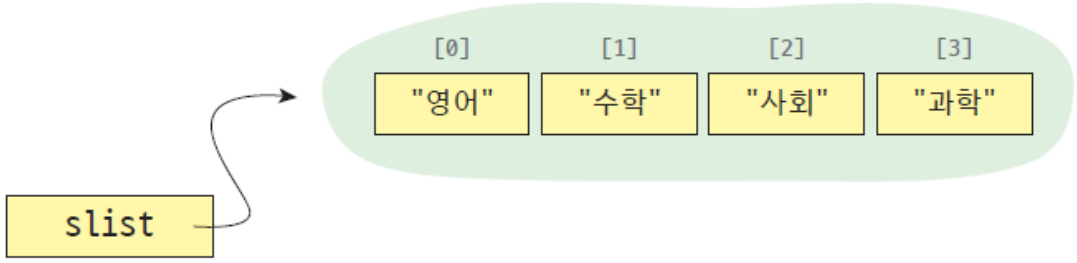
중간점검

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

리스트란

- 여러 개의 자료들을 하나의 묶음으로 저장할 수 있는 데이터 구조
- 반복구조에서 필수


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



리스트에 값을 추가하는 코드

p161.py

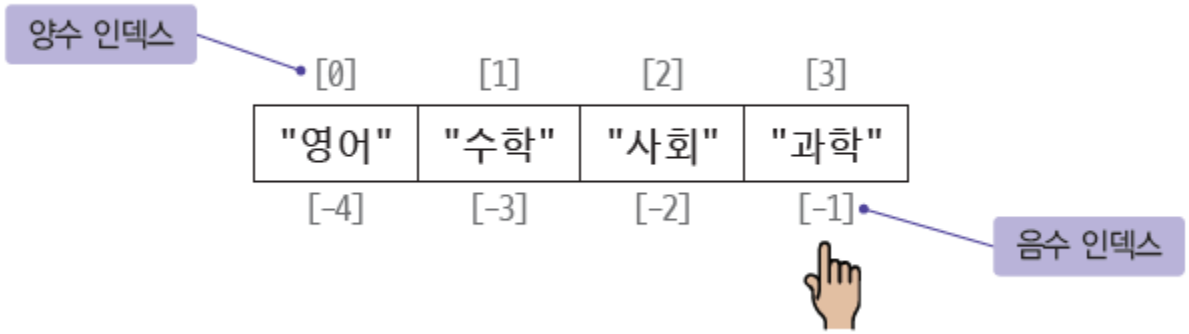
```
list = [ ]                  # 공백 리스트를 생성한다.
list.append(1)              # 리스트에 정수 1을 추가한다.
list.append(2)              # 리스트에 정수 2을 추가한다.
list.append(6)              # 리스트에 정수 6을 추가한다.
list.append(3)              # 리스트에 정수 3을 추가한다.

print(list)                 # 리스트를 출력한다.
```

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

리스트의 요소에 접근하기

- 인덱스는 0부터 시작한다.

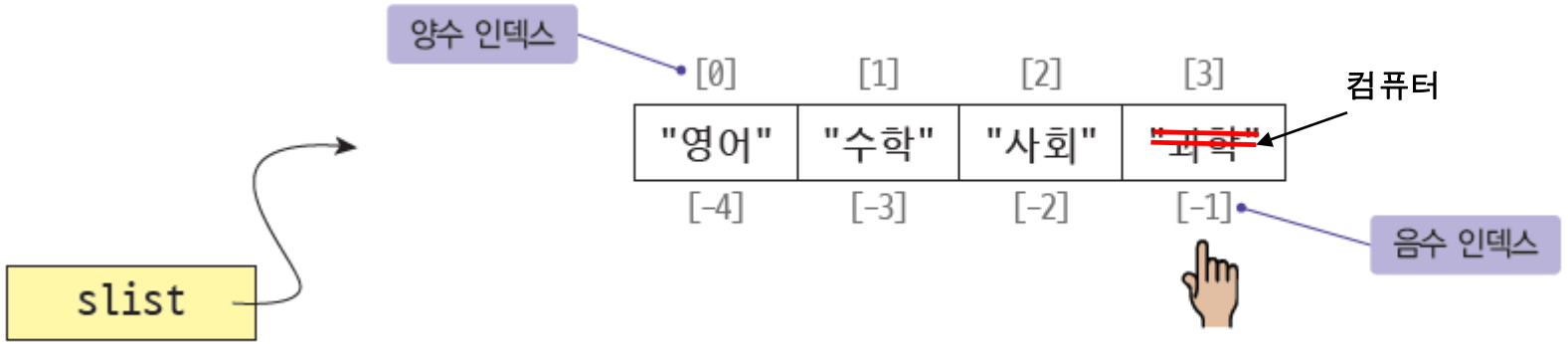
```
>>> slist = [ "영어", "수학", "사회", "과학" ]
>>> slist[0]
영어                                  # 리스트의 첫 번째 항목을 출력한다.
```



리스트의 요소에 접근하기

```
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.")
```

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 반복


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

- 앞에서 소개한 리스트에 정수를 저장해두고 하나씩 꺼내서 반복하는 방법은 반복 횟수가 1000번이라면 불가능 -> range() 함수로 반복 횟수를 전달하면 range() 함수가 자동으로 순차적인 정수들을 생성.

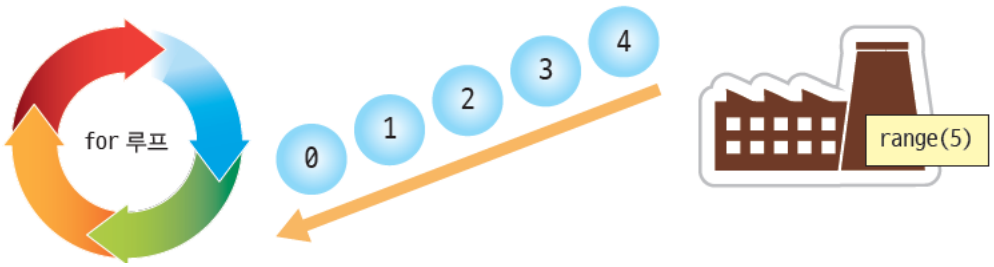
Syntax: range() 함수를 사용하는 for 문

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

0, 1, 2, 3, 4의 값들이 생성되어서 변수 i에 할당된다.

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

반복되는 문장으로 들여쓰기 하여야 한다.



range() 함수

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

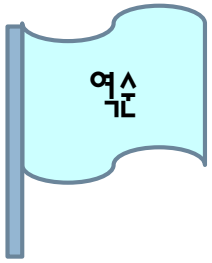
```
for i in range(5) :          # 0, 1, 2, 3, 4
    ...
for i in range(1, 6) :       # 1, 2, 3, 4, 5
    ...
for i in range(1, 6, 2) :    # 1, 3, 5
    ...
```

예제

p167.py


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

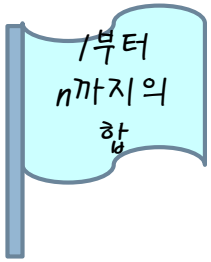
```
1 2 3 4 5
```




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

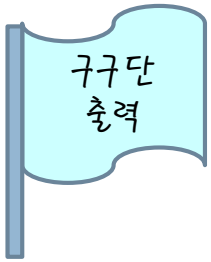
```
10 9 8 7 6 5 4 3 2 1
```

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
```

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

```
정수를 입력하시오: 10
10!은 3628800이다.
```

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

p169_factorial.py

	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

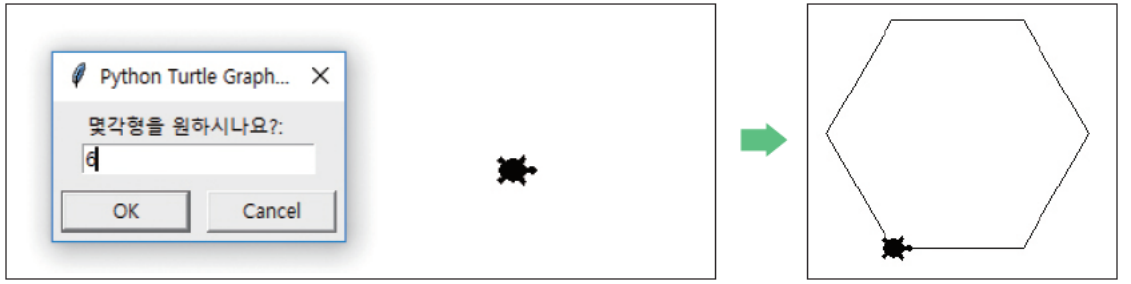
1. 팩토리얼을 거꾸로 계산해보자.

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

앞의 프로그램을 어떻게 수정하여야 하는가?

Lab: n-각형 그리기

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




```
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()
```

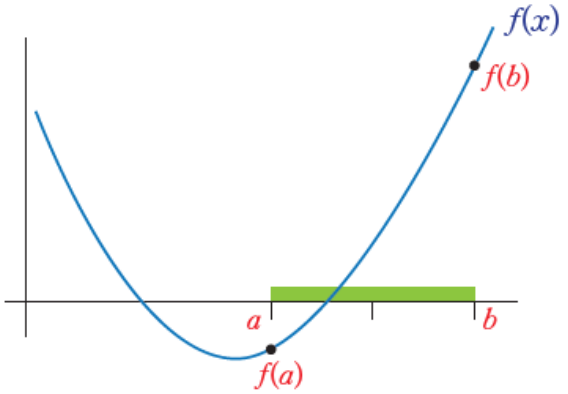
p170_shape1.py

Lab: 방정식의 해 구하기

- 수치 해석적인 방법 - 프로그램으로 해를 계산. 프로그램은 매우 빠르게 반복할 수 있다. 모든 수를 넣어서 방정식을 계산하는 것이 가능.

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

방정식의 해는 1.62

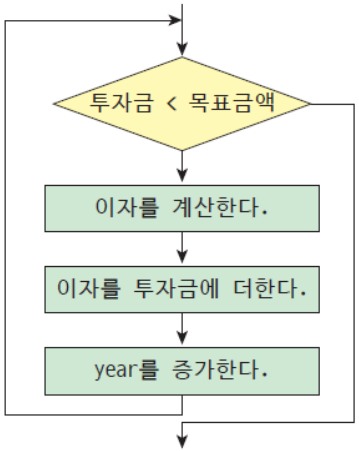



```
COUNT = 100
START = 1.0
END = 2.0

for i in range(COUNT):
    x = START + i*((END-START)/COUNT)
    f = (x**2 -x -1)
    if abs(f-0.0) < 0.01 :
        print("방정식의 해는 ", x)
```

p171_root_.py

조건 제어 반복

- 어떤 조건이 만족 되는 동안 반복한다.
- 예. 원금이 2배로 될 때까지.



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.

반복되는 문장은 동일하게 들여쓰기가 되어야 한다.

조건이 참이면 반복되는 문장들

조건 제어 반복

p173_money.py

```
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
```

예제: 1과 10까지의 합 계산하기

- 반복횟수를 알고 있는 경우 보통 for 반복문을 사용하는데 while 루프를 사용할 수도 있다.

p174_sum.py

```
# 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

p175_else.py

```
i = 0

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

```
루프 안쪽
루프 안쪽
루프 안쪽
else 부분
```

무한 루프 오류

- 반복 루프에서 조건식을 잘못 지정하면 무한히 반복(infinite loop).

p175_inf_loop1.py

```
# 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 출력.

```
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
```

p176_gugu.py

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

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

Lab: 숫자 맞추기 게임

- 프로그램이 가지고 있는 정수를 사용자가 알아맞히는 게임.
- 사용자가 답을 제시하면 프로그램은 자신이 저장한 정수와 비교하여 제시된 정수가 더 높은지 낮은지 만을 알려준다. 1부터 100가지로 한정하면 최대 7번이면 누구나 알아맞힐 수 있다.
- 이진탐색(binary search) 알고리즘

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

Solution:

p178_guess.py

```
import random

tries = 0                                  # 시도 회수
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. 시도회수=", tries)
else:
    print("정답은 ", answer)
```

Lab: 초등생을 위한 산수 문제 발생기

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

```
25 + 78 = 103
잘했어요!!

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

p180_math.py

```
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 체크.

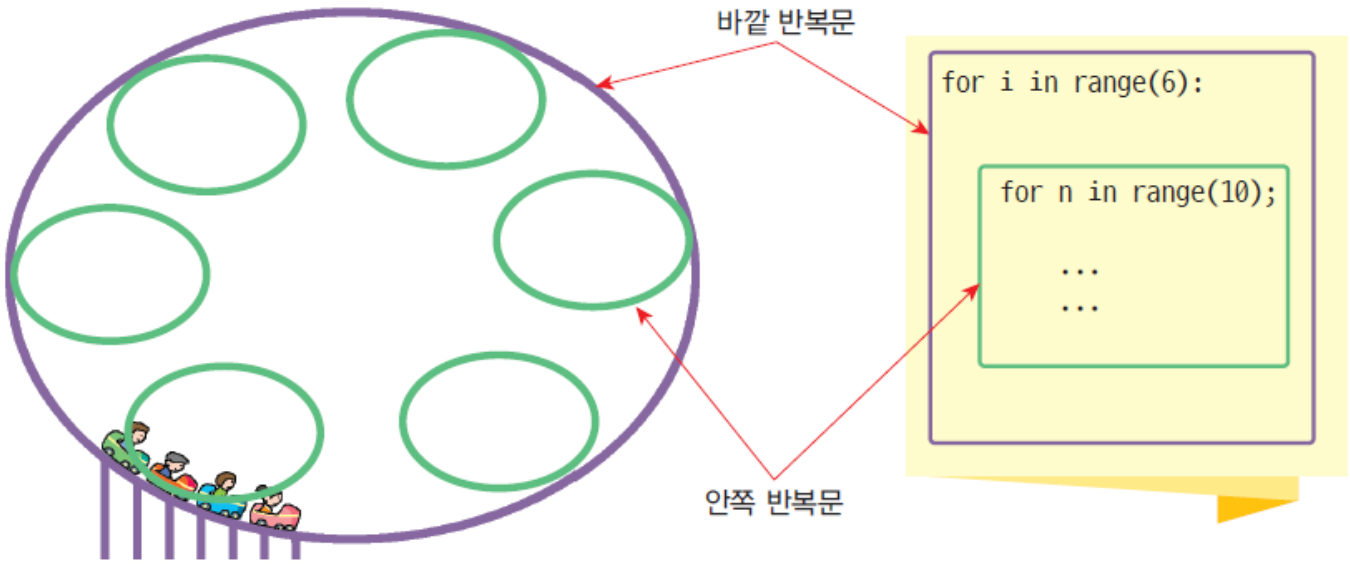
```
암호를 입력하시오: 12345678
암호를 입력하시오: password
암호를 입력하시오: pythonisfun
로그인 성공
```

p181_password.py

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

중첩 반복문

- 반복 루프 안에 다시 반복 루프가 있을 수 있다.
- 내부 반복문은 외부 반복문이 한번 반복할 때 마다 새롭게 실행된다.



예제

p182_nested1.py

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

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

p183_nested2.py

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

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

예제 : 모든 조합 구하기

p183_nested3.py

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

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

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
```

p184_dice.py

```
##
#       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}" )
```

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
...
```

p185_res.py

```
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.
```

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

p186_break.py

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

무한 루프와 break, continue

- continue : 다음 반복을 강제적으로 실행
- 1부터 10까지 3의 배수만 빼고 출력하는 프로그램

p187_continue.py

```
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: 소수 찾기

- 2부터 시작하여 50개의 소수를 찾는 프로그램을 작성해보자.

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

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

Lab: 소수 찾기

p188_prime.py

```
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: 파이 계산하기

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

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

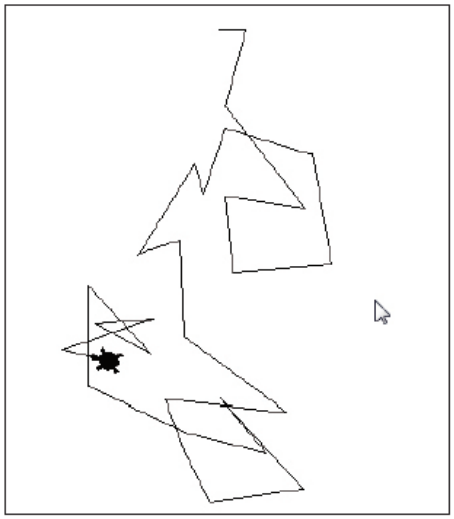
```
반복횟수:10000
Pi = 3.141493
```

p190_cal_pi_.py

```
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: 거북이를 랜덤하게 움직이게 하자

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



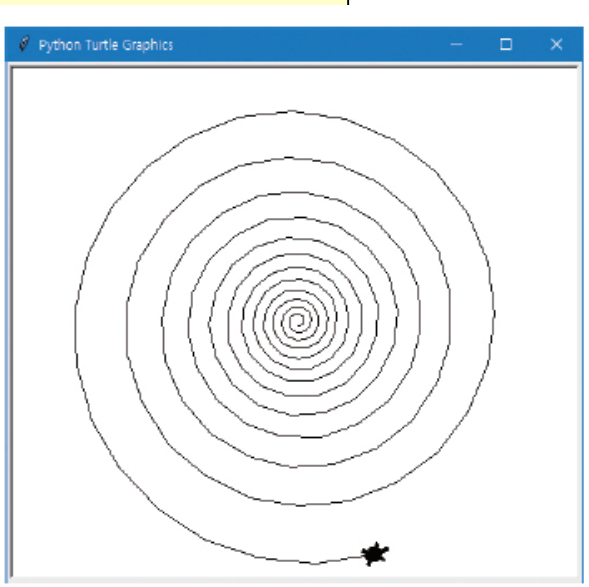
p191_random_walk.py

```
##
# 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

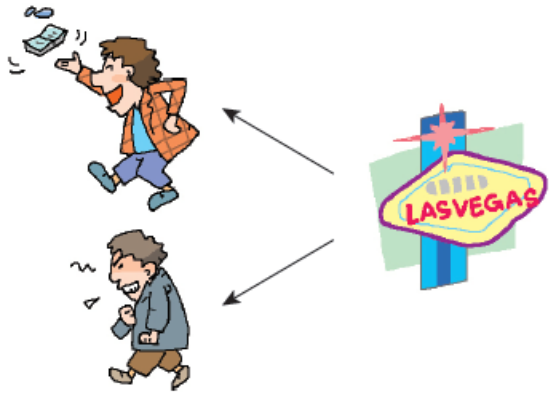
p193_spiral.py

```
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: 도박사의 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

Lab: 도박사의 확률

p194_gambler.py

```
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