

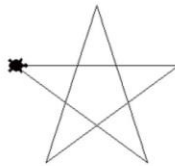
## 6. 반복

### 학습 내용

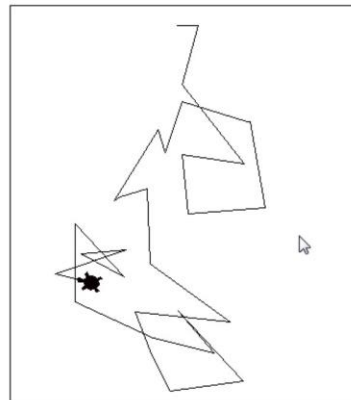
- 반복문의 필요성을 이해합니다.
- while문을 사용하여 조건으로 반복하는 방법을 학습합니다
- for문을 사용하여 정해진 횟수만큼 반복하는 방법을 학습합니다.

### 이번 장에서 만들 프로그램

- 터틀 그래픽에서 별을 반복을 이용해서 그려보자

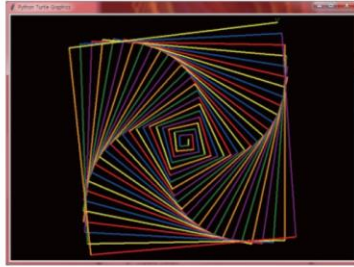


- 터틀 그래픽의 거북이를 랜덤하게 움직여보자



## 이번 장에서 만들 프로그램

- 터틀 그래픽을 이용하여 스파이럴을 그려보자.



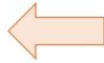
## 반복이란?

- 반복(iteration)
  - 동일한 문장을 여러 번 반복시키는 구조
- 컴퓨터는 인간과 다르게 반복적인 작업을 실수 없이 빠르게 할 수 있다. 이것이 컴퓨터의 가장 큰 장점이다.

## 왜 반복이 중요한가?

- 하나의 예로 화면에 회사에 중요한 손님이 오셔서 대형전광판에 '방문을 환영합니다!'를 5번 출력한다고 하자.

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



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

➤ welcome.py

## 만약 1000번 반복해야 한다면?

- 반복 구조를 사용하여야 한다.

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

1000번 반복시키는 구조

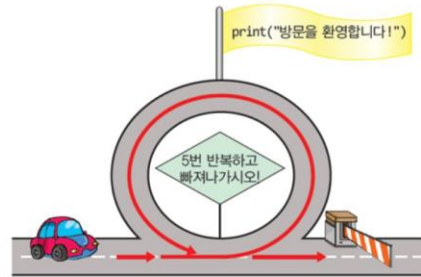
```
for 반복변수 in 순차값 :
    실행문1
else :
    실행문2
```

for 문을 수행하고 실행

## 횃수 제어 반복

- 파이썬에서 횃수 제어 반복은 for 루프라고도 한다.

```
1 ① for i in [1, 2, 3, 4, 5]:      # 끝에 :이 있어야 함
2      print("방문을 환영합니다.") # 들여쓰기하여야 함
```



> forloop.py

## 횃수 제어 반복

i가 1부터 5까지 변경되면서 반복된다.

```
for i in [1, 2, 3, 4, 5]:      # 끝에 :이 있어야 함
    print("방문을 환영합니다.") # 들여쓰기 하여야 함
```

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

> forloop.py

## 실습 - P01 : 횟수 제어 반복

●출력이 다음과 같게 나오도록 수정하세요.

The first screenshot shows the Python 3.7.2 Shell output:

```
1 번째 방문을 환영합니다.
2 번째 방문을 환영합니다.
3 번째 방문을 환영합니다.
4 번째 방문을 환영합니다.
5 번째 방문을 환영합니다.
>>>
```

The second screenshot shows the initial code in a text editor:

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

The third screenshot shows the modified code with comments:

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

Comments in the code: # 끝에 :이 있어야 함, # 들여쓰기하여야 함

> forloop.py

## i의 값을 출력해보자.

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

```
i= 1
i= 2
i= 3
i= 4
i= 5
```

> forloop1.py

## 실습 - P01-1 :

The screenshot shows an Eclipse IDE window titled 'P01-1.py'. The code in the editor is as follows:

```

for letter in 'Python':
    print('Current letter : ', letter)

print()

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:
    print('Current fruit : ', fruit)

print('Using Index')
for index in range(len(fruits)):
    print('Current fruit : ', fruits[index])
  
```

The output of the script is displayed in a separate window on the right:

```

Current letter : P
Current letter : y
Current letter : t
Current letter : h
Current letter : o
Current letter : n

Current fruit : banana
Current fruit : apple
Current fruit : mango
Using Index
Current fruit : banana
Current fruit : apple
Current fruit : mango
  
```

The status bar at the bottom of the IDE window indicates 'Ln: 12 Col: 36'.

2019-04-18

© Chang Seung Kim - All rights reserved

11

## 실습 - P01-2 : for - else 문

The first screenshot shows the Eclipse IDE window titled 'P01-2.py' with the following code:

```

numbers = [11, 33, 55, 39, 21, 25, 9, 41, 13]

for num in numbers:
    if (num % 2) == 0:
        print('리스트는 짝수를 포함합니다.')
        break;
else:
    print('리스트에는 짝수가 없습니다.')
  
```

The output for this list is: '리스트에는 짝수가 없습니다.'

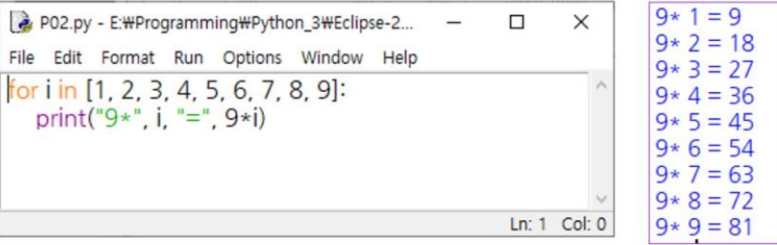
The second screenshot shows the same code but with a different list: `numbers = [11, 33, 55, 39, 21, 25, 9, 41, 13, 4]`. The output for this list is: '리스트는 짝수를 포함합니다.'

2019-04-18

© Chang Seung Kim - All rights reserved

12

## 실습 - P02 : 구구단을 출력해보자.

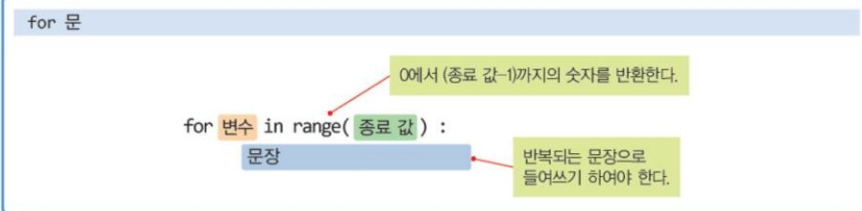


```
P02.py - E:\Programming\Python_3\Eclipse-2...
File Edit Format Run Options Window Help
for i in [1, 2, 3, 4, 5, 6, 7, 8, 9]:
    print('9*', i, '=', 9*i)
Ln: 1 Col: 0
```

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

> forloop2.py

## range() 함수



```
for 문
    for 변수 in range(종료 값) :
        문장
```

0에서 (종료 값-1)까지의 숫자를 반환한다.

반복되는 문장으로 들여쓰기 하여야 한다.

```
1 for i in range(5):           # (1)
2     print("방문을 환영합니다!") # (2)
```

방문을 환영합니다!  
방문을 환영합니다!  
방문을 환영합니다!  
방문을 환영합니다!  
방문을 환영합니다!

> range.py

## range() 함수

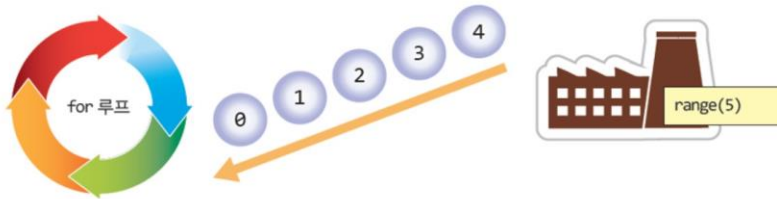
range() 함수

시작값이다.

종료 값이지만 stop은  
포함되지 않는다.

한 번에 증가되는 값이다.

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



## range() 함수의 사용

- 만약 1부터 시작하여서 5까지 반복하고 싶다면 어떻게 하면 될까?

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

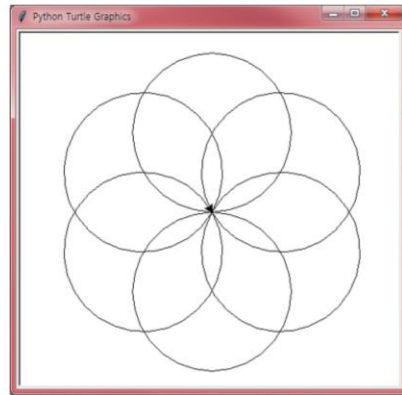
# end=" ": 줄이 바뀌지 않고 한줄에 전부 출력됨

1 2 3 4 5



## 예제

## ●6개의 원 그리기(ex1.py)



## 예제

## ●6개의 원 그리기(ex1.py)

```

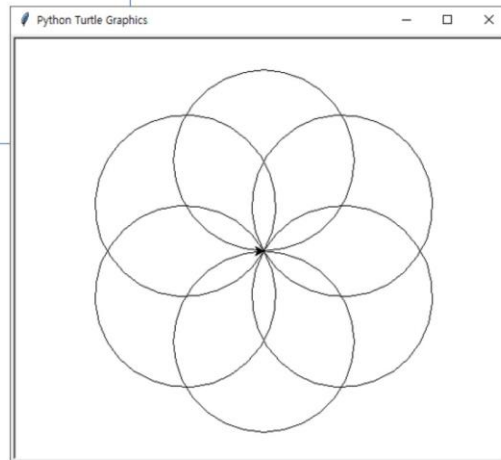
1  import turtle
2  t = turtle.Turtle()
3
4  t.circle(100)    # 반지름이 100인 원을 그린다.
5  t.left(60)       # 60도 만큼 터틀을 왼쪽으로 회전시킨다.
6  t.circle(100)
7  t.left(60)
8  t.circle(100)
9  t.left(60)
10 t.circle(100)
11 t.left(60)
12 t.circle(100)
13 t.left(60)
14 t.circle(100)

```

## 실습 - P03 :

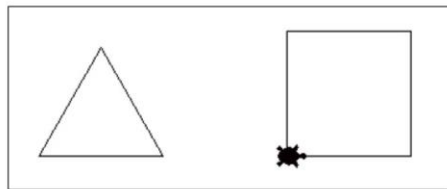
● ex2.py

```
1 import turtle
2 t = turtle.Turtle()
3
4 for count in range(6):
5     t.circle(100)
6     t.left(360/6)
```



## 실습 - P04 : 반복을 사용하여 도형 그리기

● 정삼각형과 정사각형을 반복을 이용하여 화면에 그려 보자.



## 실습 - P04 : 반복을 사용하여 도형 그리기

```

1  import turtle
2  t = turtle.Turtle()
3  t.shape("turtle")
4
5  # 정삼각형 그리기
6  for i in range(3):
7      t.forward(100)
8      t.left(360/3)
9
10 # 이동하기
11 t.penup()
12 t.goto(200, 0)
13 t.pendown()
14
15 # 정사각형 그리기
16 for i in range(4):
17     t.forward(100)
18     t.left(360/4)

```

## 실습 - P05 : n-각형 그리기

- 사용자로부터 정수  $n$ 을 받아서  $n$ -각형을 그리는 프로그램을 작성할 수 있는가?

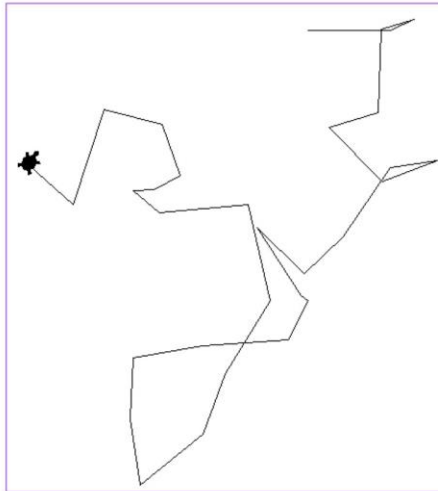


## 실습 - P05 : n-각형 그리기

```
1 import turtle
2 t = turtle.Turtle()
3 t.shape("turtle")
4
5 s = turtle.textinput("", "몇각형을 원하시나요?:")
6 n=int(s)
7
8 for i in range(n):
9     t.forward(100)
10    t.left(360/n)
```

## 실습 - P06 : 거북이를 랜덤하게 움직이게 하자

●터틀 그래픽에서 거북이가 술에 취한 것처럼 랜덤하게 움직이게 해보자.



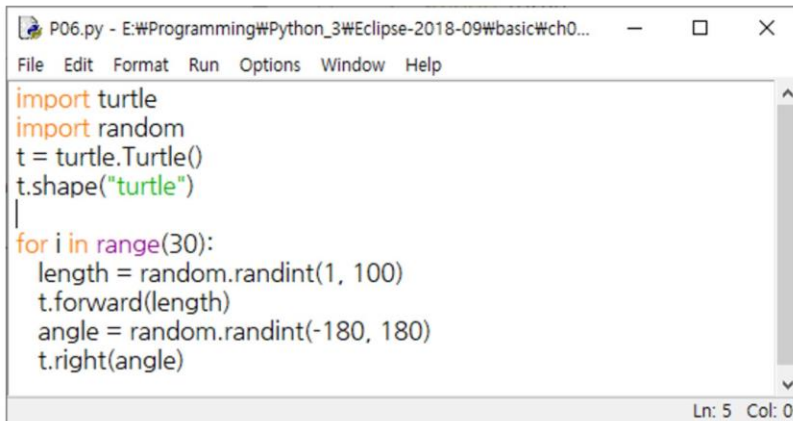
## 실습 - P06 : 거북이를 랜덤하게 움직이게 하자

### ●알고리즘

30번 반복

- \* [1, 100] 사이의 난수를 발생하여 변수 length에 저장한다.
- \* 거북이를 length만큼 움직인다.
- \* [-180, 180] 사이의 난수를 발생하여 변수 angle에 저장한다.
- \* 거북이를 angle만큼 회전시킨다.

## 실습 - P06 : 거북이를 랜덤하게 움직이게 하자



```
P06.py - E:\Programming\Python_3\Eclipse-2018-09\basic\ch0...
File Edit Format Run Options Window Help

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)

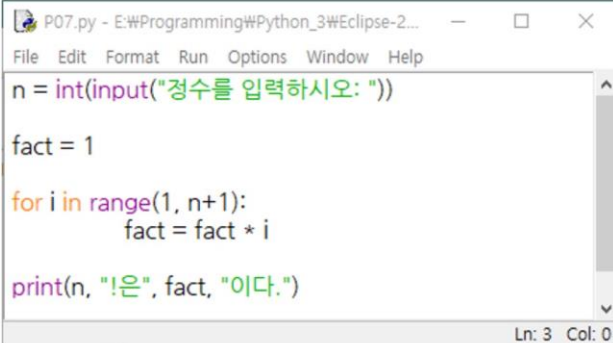
Ln: 5 Col: 0
```

## 실습 - P07 : 팩토리얼 계산하기

- for문을 이용하여서 팩토리얼을 계산해보자.
- 팩토리얼  $n!$ 은 1부터  $n$ 까지의 정수를 모두 곱한 것을 의미한다.
  - $n! = 1 \times 2 \times 3 \times \dots \times (n-1) \times n$ 이다.

정수를 입력하시오: 10  
10!은 3628800이다.

## 실습 - P07 : 팩토리얼 계산하기



```

P07.py - E:\Programming\Python_3\Eclipse-2...
File Edit Format Run Options Window Help
n = int(input("정수를 입력하시오: "))

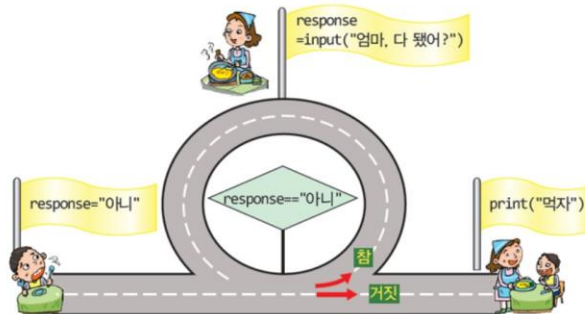
fact = 1

for i in range(1, n+1):
    fact = fact * i

print(n, "!은", fact, "이다.")
Ln: 3 Col: 0
  
```

## 조건 제어 반복

- 조건 제어 반복은 어떤 조건이 만족되는 동안 반복하는 구조



## while 문

while 루프

while 조건 :

반복 문장

반복을 하는 조건이다. 조건이 참이면 반복을 계속한다.

반복되는 문장이다.

```

1 response = "아니"
2 while response == "아니":
3     response = input("엄마, 다 됐어?");
4     print("먹자")
  
```

> while1.py

## 예제

- 사용자가 암호를 입력하고 프로그램에서 암호가 맞는지를 체크한다고 하자.

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

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

> login.py

## 실습 - P08

- 1부터 10까지의 합을 계산하는 예제를 while 루프로 작성해 보자.

```
P08.py - E:\Programming\Python_3\...
File Edit Format Run Options Window Help
count = 1
sum = 0
while count <= 10:
    sum = sum + count
    count = count + 1
print("합계는", sum)
Ln: 5 Col: 8
```

합계는 55

> while2.py



## 예제 (실습 - P09)

- 1부터  $n$ 까지의 합을 계산하는 예제를 while 루프로 작성해 보자.
- $n$ 의 값은 입력을 받아서 처리한다.

```

총합을 구하고 싶은 숫자는 : 15
1부터 1까지의 합계 : 1
1부터 2까지의 합계 : 3
1부터 3까지의 합계 : 6
1부터 4까지의 합계 : 10
1부터 5까지의 합계 : 15
1부터 6까지의 합계 : 21
1부터 7까지의 합계 : 28
1부터 8까지의 합계 : 36
1부터 9까지의 합계 : 45
1부터 10까지의 합계 : 55
1부터 11까지의 합계 : 66
1부터 12까지의 합계 : 78
1부터 13까지의 합계 : 91
1부터 14까지의 합계 : 105
1부터 15까지의 합계 : 120
  
```

➤ while2.py

## 실습 - P10 : 구구단 출력

- 구구단 중에서 원하는 단을 while문을 이용하여 출력해보자.
- $9*1, 9*2, 9*3, \dots, 9*9$ 까지 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
  
```

## 실습 - P10 : 구구단 출력

```

P10.py - E:\Programming\Python_3\Python-2018-09\lab5\
File Edit Format Run Options Window Help
dan = int(input("원하는 단은: "))
i = 1

while i <= 9:
    print("%s * %s = %s" % (dan, i, dan*i))
    i = i + 1
Ln: 5 Col: 25

```

원하는 단은: 7

```

7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63

```

> lab5.py

## 중첩 반복문

```

for 반복변수1 in 순차값2 :
    실행문1
    for 반복변수2 in 순차값2 :
        실행문2
    else :
        실행문3

```

```

while 조건문1 :
    실행문1
    while 조건문2 :
        실행문2
    실행문3

```

## 실습 - P11-1 : 도전문제: 구구단 출력

- 구구단의 1단부터 9단까지를 출력하도록 프로그램을 수정하세요.

```
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
```

```
3*1=3
3*2=6
3*3=9
3*4=12
```

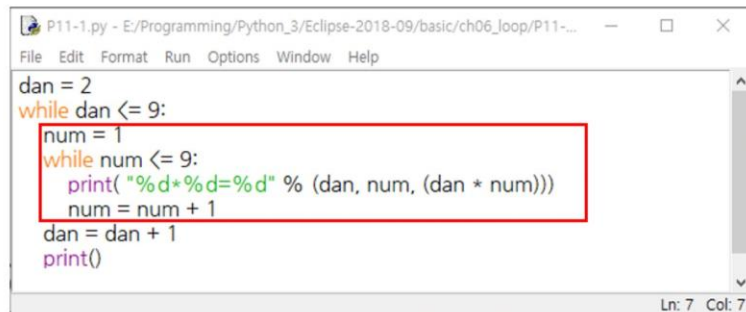
```
⋮
```

```
8*8=64
8*9=72
```

```
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
```

➤ lab5.py

## 실습 - P11-1 : 도전문제: 구구단 출력



```
P11-1.py - E:/Programming/Python_3/Eclipse-2018-09/basic/ch06_loop/P11-...
File Edit Format Run Options Window Help

dan = 2
while dan <= 9:
    num = 1
    while num <= 9:
        print( "%d*%d=%d" % (dan, num, (dan * num)))
        num = num + 1
    dan = dan + 1
print()
```

Ln: 7 Col: 7

## 실습 - P11-2 : 도전문제: 구구단 출력

- 구구단의 1단부터 9단까지를 출력하도록 프로그램을 수정하세요.

2*1=2	3*1=3	4*1=4	5*1=5	6*1=6	7*1=7	8*1=8	9*1=9
2*2=4	3*2=6	4*2=8	5*2=10	6*2=12	7*2=14	8*2=16	9*2=18
2*3=6	3*3=9	4*3=12	5*3=15	6*3=18	7*3=21	8*3=24	9*3=27
2*4=8	3*4=12	4*4=16	5*4=20	6*4=24	7*4=28	8*4=32	9*4=36
2*5=10	3*5=15	4*5=20	5*5=25	6*5=30	7*5=35	8*5=40	9*5=45
2*6=12	3*6=18	4*6=24	5*6=30	6*6=36	7*6=42	8*6=48	9*6=54
2*7=14	3*7=21	4*7=28	5*7=35	6*7=42	7*7=49	8*7=56	9*7=63
2*8=16	3*8=24	4*8=32	5*8=40	6*8=48	7*8=56	8*8=64	9*8=72
2*9=18	3*9=27	4*9=36	5*9=45	6*9=54	7*9=63	8*9=72	9*9=81

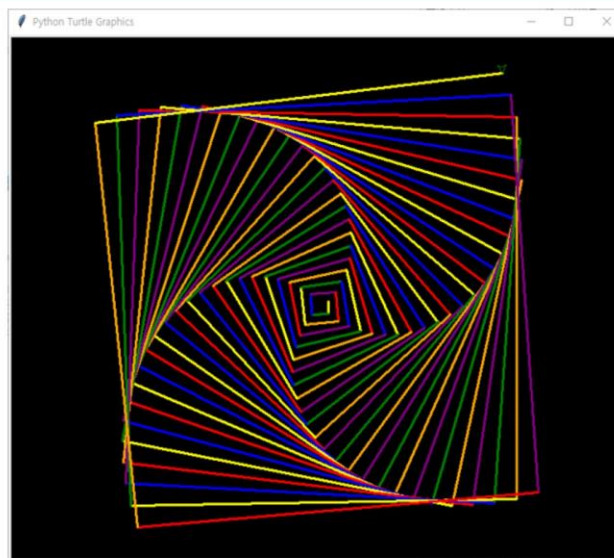
```

P11-2.py - E:\Programming\Python_3\Eclipse-2018-09\basic\ch06_loop\WP1...
File Edit Format Run Options Window Help
num1 = 2
while num1 <= 9:
    num2 = 1
    while num2 <= 9:
        num2 = num2 + 1
    num1 = num1 + 1
    print()
Ln: 10 Col: 0

```

➤ lab5.py

## Lab7: 스파이럴 그리기



## Lab7: 스파이럴 그리기

### ●lab7.py

```

1 import turtle
2
3 # 색상은 리스트에 저장했다가 하나씩 꺼내서 변경하도록 하자.
4 colors = ["red", "purple", "blue", "green", "yellow", "orange"]
5 t = turtle.Turtle()
6
7 # 배경색은 다음과 같은 문장으로 변경이 가능하다.
8 turtle.bgcolor("black")
9
10 # 거북이의 속도는 0으로 설정하면 최대가 된다.
11 t.speed(0)
12
13 # 거북이가 그리는 선의 두께는 width()를 호출하면 된다.
14 t.width(3)
15
16 length = 10 # 초기 선의 길이는 10으로 한다.
17
18 # while 반복문이다. 선의 길이가 500보다 작으면 반복한다.
19 while length < 500:
20     t.forward(length)           # length만큼 전진한다.
21     t.pencolor(colors[length%6]) # 선의 색상을 변경한다.
22     t.right(89)                # 89도 오른쪽으로 회전한다.
23     length += 5                # 선의 길이를 5만큼 증가한다.

```

2019-04-18

© Chang Seung Kim - All rights reserved

41

## 실습 - P12 : 사용자가 입력하는 숫자의 합 계산하기

### ●사용자가 입력한 숫자를 더하는 프로그램

- 사용자가 yes 라고 답한 경우에만 입력받는다.
- 출력 화면

```

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

```

2019-04-18

© Chang Seung Kim - All rights reserved

42

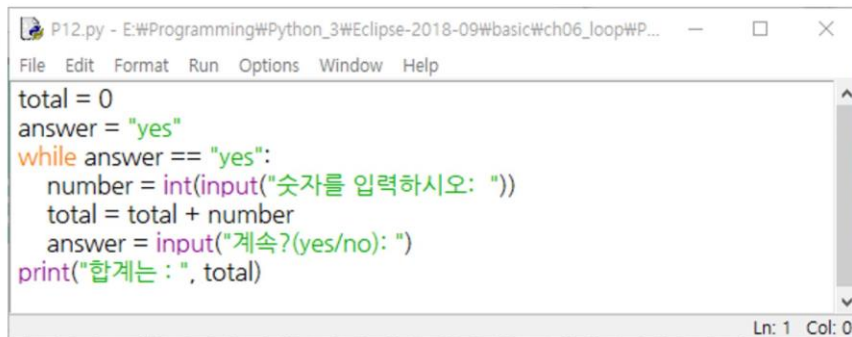
## 실습 - P12 : 사용자가 입력하는 숫자의 합 계산하기

### ●사용자가 입력한 숫자를 더하는 프로그램

#### ●알고리즘

1. total을 0으로 설정한다.
2. answer를 'yes' 로 설정한다.
3. answer가 'yes' 인 동안에 다음을 반복한다.
  - \* 숫자를 입력받는다.
  - \* 숫자를 total에 더한다.
  - \* '계속? yes/no' 을 묻는다.
4. total의 값을 출력한다.

## 실습 - P12 : 사용자가 입력하는 숫자의 합 계산하기



```

total = 0
answer = "yes"
while answer == "yes":
    number = int(input("숫자를 입력하시오: "))
    total = total + number
    answer = input("계속?(yes/no): ")
print("합계는 : ", total)
  
```

Ln: 1 Col: 0



## 실습 - P13 : 숫자 맞추기 게임

- 1부터 100사이의 숫자를 맞추는 게임을 작성하시오. 맞출 때까지 계속 반복한다.
  - 숫자를 입력하면 입력된 숫자와 프로그램이 만든 숫자를 비교하여 높고 낮음을 알려준다.
  - 맞출때까지 시도한 횟수를 계산하여 맞추면 시도횟수를 출력한다.
  - 출력 화면

```
1부터 100 사이의 숫자를 맞추시오
숫자를 입력하시오: 50
낮음!
숫자를 입력하시오: 86
낮음!
숫자를 입력하시오: 87
축하합니다. 시도횟수= 3
```

2019-04-18

© Chang Seung Kim - All rights reserved

45

## 실습 - P13 : 숫자 맞추기 게임

- 1부터 100사이의 숫자를 맞추는 게임을 작성하시오. 맞출 때까지 계속 반복한다.
  - 알고리즘

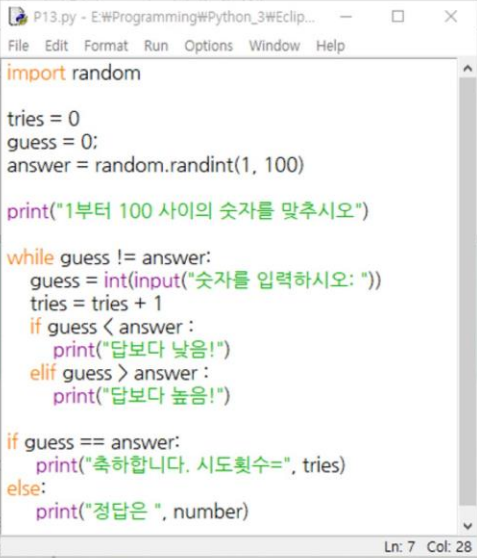
```
while guess != answer
    사용자로부터 숫자를 guess로 입력받는다.
    시도횟수를 증가한다.
    if( guess < answer )
        숫자가 낮다고 출력한다.
    if( guess > answer )
        숫자가 높다고 출력한다.
    "축하합니다"와 시도횟수를 출력한다.
```

2019-04-18

© Chang Seung Kim - All rights reserved

46

## 실습 - 13 : 숫자 맞추기 게임



```

import random

tries = 0
guess = 0;
answer = random.randint(1, 100)

print("1부터 100 사이의 숫자를 맞추시오")

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

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

```

1부터 100 사이의 숫자를 맞추시오

숫자를 입력하시오: 50

답보다 높음!

숫자를 입력하시오: 25

답보다 낮음!

숫자를 입력하시오: 37

답보다 낮음!

숫자를 입력하시오: 43

답보다 높음!

숫자를 입력하시오: 40

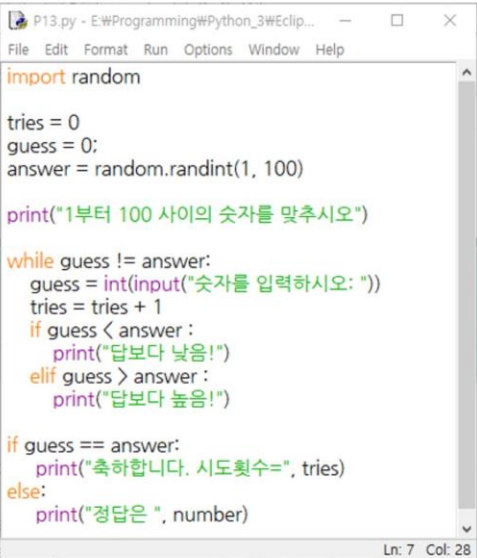
답보다 높음!

숫자를 입력하시오: 38

축하합니다. 시도횟수= 6

2019-04-18
© Chang Seung Kim - All rights reserved
47

## 실습 - 13 : 숫자 맞추기 게임



```

import random

tries = 0
guess = 0;
answer = random.randint(1, 100)

print("1부터 100 사이의 숫자를 맞추시오")

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

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

```

이 프로그램의  
잘못된 부분은?

2019-04-18
© Chang Seung Kim - All rights reserved
48



## 실습 - 13 : 숫자 맞추기 게임

```

P13.py - E:\Programming\Python_3\Weclip...
File Edit Format Run Options Window Help

import random

tries = 0
guess = 0;
answer = random.randint(1, 100)

print("1부터 100 사이의 숫자를 맞추시오")

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

if guess == answer:
    print("축하합니다. 시도횟수=", tries)
else:
    print("정답 실행 횟수 없다. ", number)
    
```

이 프로그램의  
잘못된 부분은?

2019-04-18

© Chang Seung Kim - All rights reserved

49

## 무한 루프와 break

### ●무한 루프 (Infinite Loop)

- 조건 제어 루프에서 프로그램이 무한히 반복하는 것
- 프로그램이 끝나지 않는다.
- 무한 루프를 사용하는 경우가 많다.
- ◆예 : 신호등 제어 프로그램은 무한히 반복하여야 한다.

### ●무한 반복 루프 형식

```

while True :
    반복 문장
    반복 문장
    if 조건 :
        break;
    
```

2019-04-18

© Chang Seung Kim - All rights reserved

50

## 실습 - P14 : 소수(Prime Number) 구하기

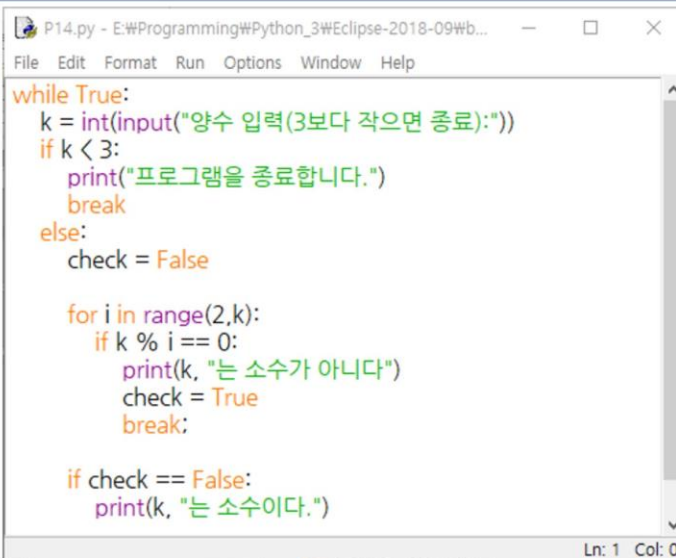
### ●1부터 입력받은 숫자까지 소수를 찾는 프로그램

- 소수 : 1과 자기 자신으로만 나누어 지는 수
- 실행 예
  - ◆ 숫자를 입력받는다.
  - ◆ 입력받은 수가 소수인지 확인하여 "소수가 아니다", "소수이다"를 프린트한다.
  - ◆ 3보다 작은 숫자를 넣으면 종료한다.

```

양수 입력(3보다 작으면 종료):13
13 는 소수이다.
양수 입력(3보다 작으면 종료):15
15 는 소수가 아니다
양수 입력(3보다 작으면 종료):7
7 는 소수이다.
양수 입력(3보다 작으면 종료):2
프로그램을 종료합니다.
  
```

## 실습 - P14 : 소수(Prime Number) 구하기



```

P14.py - E:\Programming\Python_3\Eclipse-2018-09\wb...
File Edit Format Run Options Window Help
while True:
    k = int(input("양수 입력(3보다 작으면 종료):"))
    if k < 3:
        print("프로그램을 종료합니다.")
        break
    else:
        check = False
        for i in range(2,k):
            if k % i == 0:
                print(k, "는 소수가 아니다")
                check = True
                break;
        if check == False:
            print(k, "는 소수이다.")
Ln: 1 Col: 0
  
```

## 과제

●수업에서 다룬 모든 프로그램을 작성하여 다음을 제출하시오.

- for 문을 사용한 프로그램을 while 문으로 고칠것
- while 문을 사용한 프로그램을 for 문으로 고칠것
- 같은 프로그램이 2 세트 있어야 함 - for 문과 while 문

2019-04-18

© Chang Seung Kim - All rights reserved

53

## 과제

●제출파일명 : 과제20190412-학번-이름.zip

●제출물

- 프로그램 소스 (\*.py)
- 설명 파일 (Word, Powerpoint, HWP 중 택1)을 작성
  - ◆실행 화면 캡처
  - ◆간단한 프로그램 설명

●제출기한 : 다음 수업시간 전일 자정까지

●제출 방법 : lms.sch.ac.kr

2019-04-18

© Chang Seung Kim - All rights reserved

54

## 이번 장에서 배운 것

- 반복문(for, while)은 프로그래밍을 하는데 중요한 제어구조이다.
- 횟수 제어반복이 필요할때는 for 문이 적당하다.
- 어떤 조건이 만족되는 동안에만 반복을해야 할때는 while문이 적당하다.
- 합계 계산을 할때와같이 숫자들의 합/곱 등을 누적해 갈때는 그값을 저장할 변수를 초기화해놓고 써야한다. (예, sum = 0, prod = 1)
- 구구단 전체 출력할때와 같이 반복문안에 또 하나의 반복문이있는 경우 이를 nested loop 라고한다.
- 반복문, 조건문등은 필요한만큼 조합하여 사용할수있다. 즉 반복문안에 조건문이 있을 수 있고, 조건문 안에 반복문이 있을 수도 있다.