

Loop 1

By 윤명근 / 박수현

수업목표

- What is a loop
- Counting loops, **for** loops
- Using a counting loop
- Loop variable names
- Counting by steps
- Counting without numbers
- **while** loops
- Bailing out of a loop – break and continue

What is a loop

- 컴퓨터가 가장 잘 하는 일
 - Loop: 반복 작업 (계산)
- Counting loops (**for**)
 - 특정 횟수만큼 반복
예) 100회 반복
- Conditional loops (**while**)
 - 특정한 조건이 만족될 때까지 반복
 - 정답을 맞추는 경우까지 반복

2 가지의 반복 구조

- for loops
 - 정해진 횟수만큼 반복하는 구조
- while loops
 - 어떤 조건이 만족되는 동안, 반복을 계속하는 구조

for loops

전체적인 구조



for 변수 in 시퀀스

반복 문장

반복 문장

각 반복마다 변수의 값이 컨테이너의
요소값으로 설정된다.

리스트처럼 요소들을 가지고
있는 객체이다.

블록으로 들여쓰기 하여야 한다.

Counting loops, for loops

- for loops

for 타깃 in 시퀀스
실행문

3.2example1.py

File Edit Format Run Options Window Help

#list에 대한 반복

```
print("\n> list에 대한 반복입니다")
```

```
for i in [10, 20, 30, 40]:
```

```
    print(">> list에서 값을 차례대로 가져옵니다. :", i)
```

```
print("\ndone")
```

> list에 대한 반복입니다

>> list에서 값을 차례대로 가져옵니다. : 10

>> list에서 값을 차례대로 가져옵니다. : 20

>> list에서 값을 차례대로 가져옵니다. : 30

>> list에서 값을 차례대로 가져옵니다. : 40

done

Counting loops, for loops

```
for1.py -
File Edit Format Run Options Window Help
# list에 대한 반복 2
i = 1
for name in ["철수", "영희", "길동", "유신", "경희"]:
    print("안녕! " + name, i)
    i = i + 1

print("\nDone")
```

```
1 철수
2 영희
3 길동
4 유신
5 경희
Done
```

```
*for2.py -
File Edit Format Run Options
# list에 대한 반복 3
i = 0
for in_char in "abcdefghi":
    print(in_char, i)
    i = i + 1

print("\nDone")
```

```
a 0
b 1
c 2
d 3
e 4
f 5
g 6
h 7
i 8
Done
>>>
```

Using a counting loop

- for loops

`range()` 함수

- `range(n)`

- `[0, 1, 2, ..., n-1]`

- 실제 리스트를 생성하지는 않음. 보충자료에서 자세히 설명.

- num1번 반복

- `range(n1, n2)` ($n1 < n2$)

- `[n1, n1 + 1, n1 + 2, ..., n2 - n1 - 1]`

- $(n2 - n1)$ 번 반복

- `range(n1, n2, n3)`

- $n1 < n2, 0 < n3$

- `[n1, n1 + n3, n1 + 2 * n3, ..., n1 + k * n3]`

- $(n1 + k * n3 < n2)$ 까지 반복

- $n1 > n2, 0 > n3$

- `[n1, n1 + n3, n1 + 2 * n3, ..., n1 + k * n3]`

- $(n1 + k * n3 > n2)$ 까지 반복

Using a counting loop

3.2example3.py

File Edit Format Run Options Window Help

```
print("\n--> range(10)")
for i in range(10):
    print(i, end=" ")

print("\n\n--> range(0,10)")
for i in range(0,10):
    print(i, end=" ")

print("\n\n--> range(3,9)")
for i in range(3,9):
    print(i, end=" ")

print("\n\n--> range(0,10, 2)")
for i in range(0,10, 2):
    print(i, end=" ")

print("\n\n--> range(10, 4, -2)")
for i in range(10, 4, -2):
    print(i, end=" ")
```

```
--> range(10)
0 1 2 3 4 5 6 7 8 9
```

```
--> range(0,10)
0 1 2 3 4 5 6 7 8 9
```

```
--> range(3,9)
3 4 5 6 7 8
```

```
--> range(0,10, 2)
0 2 4 6 8
```

```
--> range(10, 4, -2)
10 8 6
```

```
# range
print("\n--> range(0, 10)")
for i in range(0, 10):
    print (i, end=' ')

# list
print("\n--> list")
for i in [0,1,2,3,4,5,6,7,8,9]:
    print (i, end=' ')

print()
print ("\nlist(range(10)) = ", list(range(10)))
print ("\nlist(range(0,10)) = ", list(range(0,10)))

--> range(0, 10)
0 1 2 3 4 5 6 7 8 9

--> list
0 1 2 3 4 5 6 7 8 9

list(range(10)) = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

list(range(0,10)) = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```

Using a counting loop

- for loops
 - range() 함수
 - Python 3에서는 **range 객체 생성**
 - 데이터가 요구되는 경우에 하나씩 넘겨줌
 - » 게으른 계산 추가 (lazy evaluation)
 - 메모리를 효율적으로 사용할 수 있음
 - range()를 리스트로 변환하려면 **명시적 type casting 사용**
 - » **list(range(10))**
 - 객체에 대한 설명은 강의 후반부에 다룸

Using a counting loop

- for loops
 - `range()` 함수
 - `sys.getsizeof()` :
객체의 크기를
byte단위로 return

get_sizeof_range.py -

File Edit Format Run Options Window Help

```
import sys
```

```
a = [1,2]
size = sys.getsizeof(a)
print("a size = ", size)
```

```
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
size = sys.getsizeof(b)
print("b size = ", size)
```

```
size = sys.getsizeof(range(2))
print("range(2) size = ", size)
```

```
size = sys.getsizeof(range(10))
print("range(10) size = ", size)
```

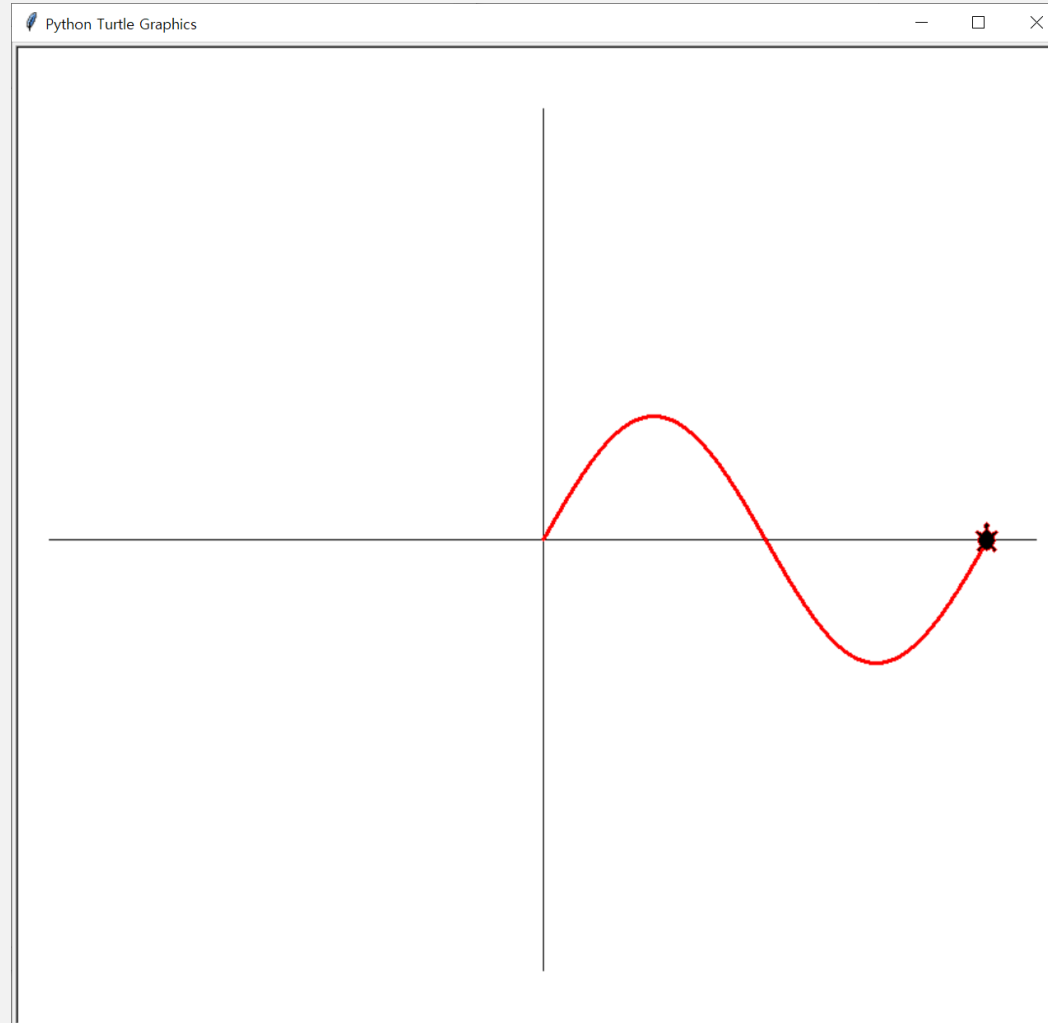
```
type_of_data = type(range(10))
print("type of range(10) : ", type_of_data)
```

```
type_of_data = type(a)
print("type of a : ", type_of_data)
```

```
a size = 36
b size = 68
range(2) size = 24
range(10) size = 24
type of range(10) : <class 'range'>
type of a : <class 'list'>
```

실습 함수 그래프 그리기

- Sine 그래프를 반복문을 이용하여 그리시오.
- Turtle graphics의 기능 사용



sine_wave.py

File Edit Format Run Options Window Help

```
import math
import turtle

t = turtle.Turtle()
```

```
# x축 그리기
t.forward(400)
t.penup()
t.goto(0,0)
```

```
t.left(180)
t.pendown()
t.forward(400)
t.goto(0,0)
```

```
# y축 그리기
t.left(90)
t.pendown()
t.forward(350)
t.penup()
t.goto(0,0)
```

```
t.left(180)
t.pendown()
t.forward(350)
t.penup()
t.goto(0,0)
```

```
t.pensize(3)
t.shape('turtle')
```

```
# Sine wave
t.pencolor("red")
t.pendown()
```

```
for angle in range(360):
```

```
# 360번 반복
```

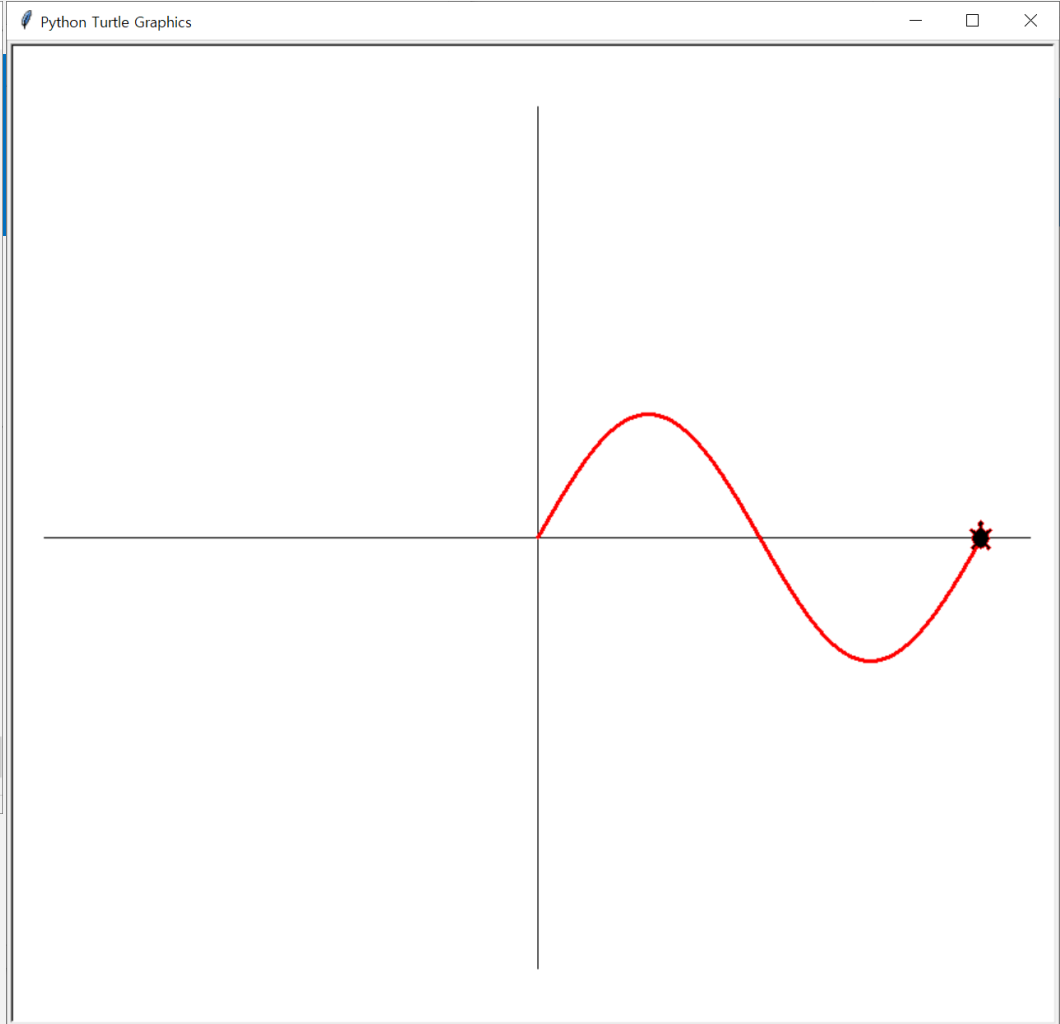
```
    y = math.sin(math.radians(angle)) # angle 값에 대응되는 sine값을 계산
    scaledX = angle                  # x축의 좌표값을 각도로 정함
    scaledY = y * 100                 # y축의 좌표값을 싸인값으로 정함
    t.goto(scaledX, scaledY)          # 터틀 객체를 (scaledX, scaledY)로 이동
    print("x =", scaledX, "y =", scaledY)
```

Python 3.8.5 Shell

File Edit Shell Debug Options Window Help

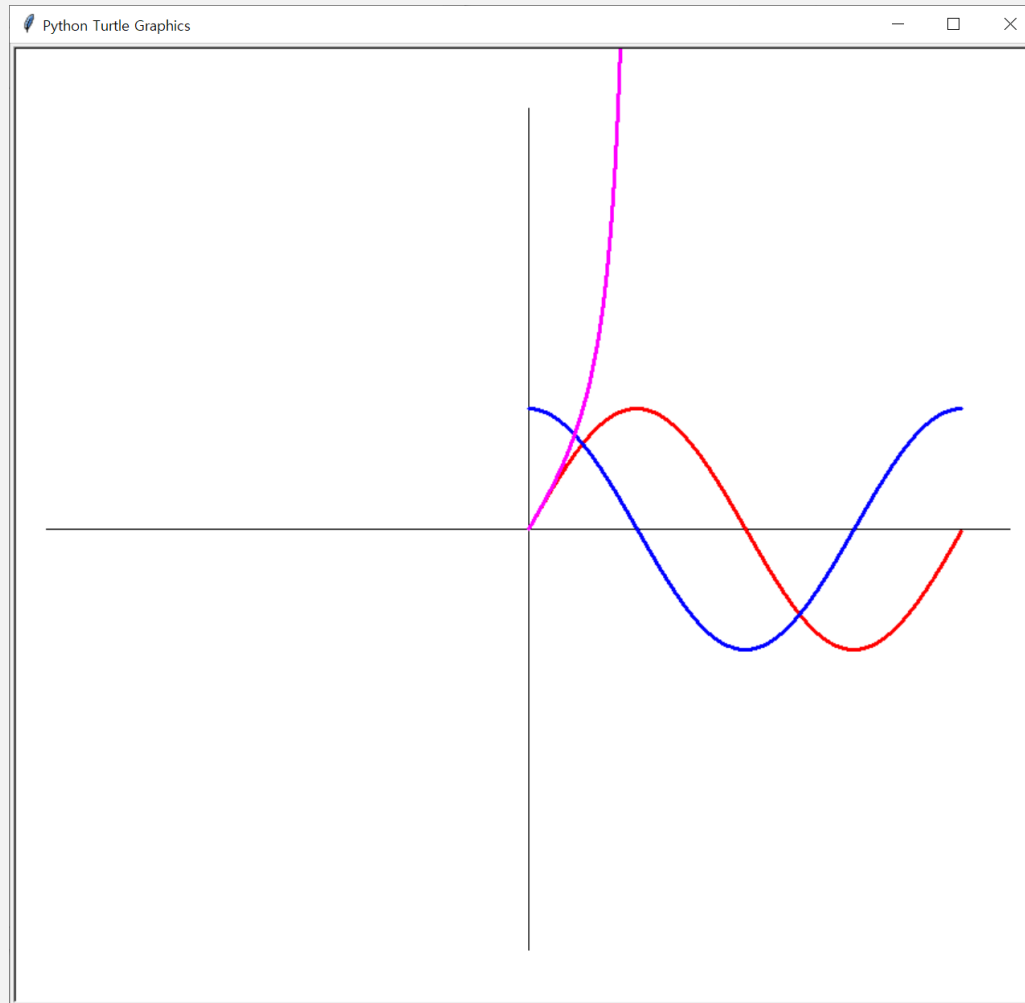
```
x = 322 y = -61.56614753256582
x = 323 y = -60.181502315204824
x = 324 y = -58.778525229247336
x = 325 y = -57.35764363510465
x = 326 y = -55.91929034707466
x = 327 y = -54.4639035015027
x = 328 y = -52.9919264233205
x = 329 y = -51.503807491005446
x = 330 y = -50.000000000000004
x = 331 y = -48.48096202463369
x = 332 y = -46.947156278589084
x = 333 y = -45.399049973954696
x = 334 y = -43.83711467890778
x = 335 y = -42.261826174069924
x = 336 y = -40.67366430758001
x = 337 y = -39.07311284892739
x = 338 y = -37.46065934159123
x = 339 y = -35.83679495453008
x = 340 y = -34.20201433256686
x = 341 y = -32.55681544571567
x = 342 y = -30.901699437494763
x = 343 y = -29.237170472273714
x = 344 y = -27.563735581699895
x = 345 y = -25.881904510252067
x = 346 y = -24.19218955996679
x = 347 y = -22.495105434386534
x = 348 y = -20.791169081775987
x = 349 y = -19.080899537654467
x = 350 y = -17.36481776669304
x = 351 y = -15.643446504023112
x = 352 y = -13.917310096006588
x = 353 y = -12.186934340514723
x = 354 y = -10.452846326765341
x = 355 y = -8.715574274765832
x = 356 y = -6.9756473744125636
x = 357 y = -5.233595624294437
x = 358 y = -3.4899496702500823
x = 359 y = -1.745240643728356
x = 360 y = -2.4492935982947064e-14
>>>
```

Ln: 1161 Col: 31



실습 함수 그래프 그리기

- Cosine, Tangent 그래프를 반복문을 이용하여 그리세요.
- Turtle graphics의 기능 사용



```
import math
import turtle

t = turtle.Turtle()
```

```
# x축 그리기
```

```
t.forward(400)
t.penup()
t.goto(0,0)
```

```
t.left(180)
t.pendown()
t.forward(400)
t.goto(0,0)
```

```
# y축 그리기
```

```
t.left(90)
t.pendown()
t.forward(350)
t.penup()
t.goto(0,0)
```

```
t.left(180)
t.pendown()
t.forward(350)
t.penup()
t.goto(0,0)
```

```
t.pensize(3)
t.shape('turtle')
```

```
# Sine wave
```

```
t.pencolor("red")
t.pendown()
```

```
for angle in range(360):          # 360번 반복

    y = math.sin(math.radians(angle)) # angle 값에 대응되는 sine값을 계산
    scaledX = angle                 # x축의 좌표값을 각도로 정함
    scaledY = y * 100               # y축의 좌표값을 싸인값으로 정함
    t.goto(scaledX, scaledY)        # 터틀 객체를 (scaledX, scaledY)로 이동
    print("x =", scaledX, "y =", scaledY)
```

```
# Cosine wave
t.goto(0, 0)
t.pencolor("blue")
```

```
for angle in range(360):          # 360번 반복
    y = math.cos(math.radians(angle)) # angle 값에 대응되는 sine값을 계산
    scaledX = angle                 # x축의 좌표값을 각도로 정함
    scaledY = y * 100               # y축의 좌표값을 싸인값으로 정함
    t.goto(scaledX, scaledY)        # 터틀 객체를 (scaledX, scaledY)로 이동
    t.pendown()
    print("x =", scaledX, "y =", scaledY)
```

```
t.penup()                          # 터틀 객체의 펜을 올린다.
```

```
# Tangent wave
```

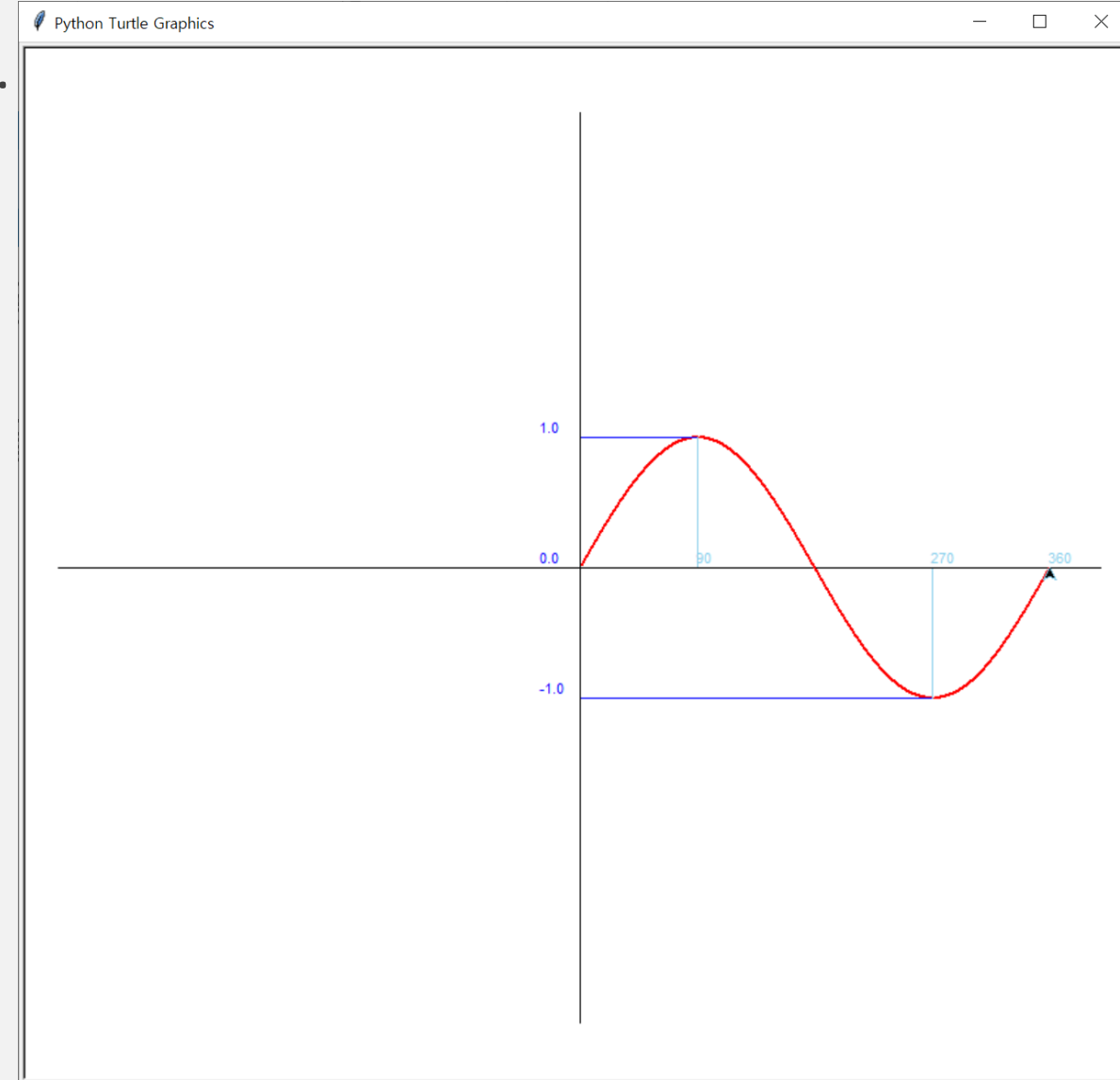
```
t.goto(0, 0)
t.pencolor("magenta")
t.pendown()
for angle in range(90):          # 90번 반복

    y = math.tan(math.radians(angle)) # angle 값에 대응되는 sine값을 계산
    scaledX = angle                 # x축의 좌표값을 각도로 정함
    scaledY = y * 100               # y축의 좌표값을 싸인값으로 정함
    t.goto(scaledX, scaledY)        # 터틀 객체를 (scaledX, scaledY)로 이동

    print("x =", scaledX, "y =", scaledY)
```


실습 함수 그래프 그리기 – drawing scale

- Sine 그래프를 반복문을 이용하여 그리시오.
- Turtle graphics의 기능 사용
- 그림과 같이 x, y 축의 scale 을 표시하시오.



```
1 import math
2 import turtle
3
4 t = turtle.Turtle()
5
6 # x축 그리기
7 t.forward(400)
8 t.penup()
9 t.goto(0,0)
10
11 t.left(180)
12 t.pendown()
13 t.forward(400)
14 t.goto(0,0)
15
16 # y축 그리기
17 t.left(90)
18 t.pendown()
19 t.forward(350)
20
21 t.penup()
22 t.goto(0,0)
23 t.left(180)
24 t.pendown()
25 t.forward(350)
26
27 t.penup()
28 t.goto(0,0)
29
30 t.pensize(3)
31 #t.shape('turtle')
32
33 # Sine wave
34 #t.pendown()
```

```
35
36 for angle in range(361):
```

```
37
38     t.pencolor("red")
39     t.pensize(2)
```

```
40
41     y = math.sin(math.radians(angle))
```

```
42
43     scaledX = angle
```

```
44     scaledY = y * 100.0
```

```
45     t.goto (scaledX, scaledY)
```

```
46     t.pendown()
```

```
47
48     if angle in [0, 90, 270]:
```

```
49         t.pencolor("blue")
```

```
50         t.pensize(1)
```

```
51         t.goto (0, scaledY)
```

```
52         t.penup()
```

```
53         t.goto (0-30, scaledY)
```

```
54         t.write(scaledY/100.0)
```

```
55
56     if angle in [90, 180, 270, 360]:
```

```
57         t.pencolor("skyblue")
```

```
58         t.pensize(1)
```

```
59         t.penup()
```

```
60         t.goto (0, scaledY)
```

```
61         t.goto (scaledX, scaledY)
```

```
62         t.pendown()
```

```
63         t.goto (scaledX, 0)
```

```
64         t.write(scaledX)
```

```
65         t.penup()
```

```
66
```

360번 반복

angle 값에 대응되는 sine값을 계산

x축의 좌표값을 각도로 정함

y축의 좌표값을 싸인값으로 정함

터틀 객체를 (scaledX, scaledY)로 이동

Loop variable names

- i, j, k 많이 사용

Counting by steps

- for loops
 - range(num1, num2, num3)
 - num1 < num2, 0 < num3
 - [num1, num1+num3, num1+2*num3, ..., num1+k*num3]
 - (num1+k*num3 < num2)까지 반복



3.2example5.py


File Edit Format Run Options Window Help

```
for i in range(1, 10, 3):  
    print(">> 1 부터 3씩 증가합니다. : ", i)  
  
print("\n>> list(range(1, 10, 3)) = ", list(range(1, 10, 3)))
```

```
>> 1 부터 3씩 증가합니다. : 1  
>> 1 부터 3씩 증가합니다. : 4  
>> 1 부터 3씩 증가합니다. : 7  
  
>> list(range(1, 10, 3)) = [1, 4, 7]
```

Counting by steps

- for loops
 - range(num1, num2, num3)
 - num1 > num2, 0 > num3
 - [num1, num1+num3, num1+2*num3, ..., num1+k*num3]
 - (num1+k*num3 > num2)까지 반복

 3.2example6.py

File Edit Format Run Options Window Help

```
for i in range(10, 1, -3):  
    print(">> 10 부터 3씩 감소합니다. :", i)
```

```
print("\n>> list(range(10, 1, -3)) = ", list(range(10, 1, -3)))
```

```
>> 10 부터 3씩 감소합니다. : 10  
>> 10 부터 3씩 감소합니다. : 7  
>> 10 부터 3씩 감소합니다. : 4
```

```
>> list(range(10, 1, -3)) = [10, 7, 4]
```

.소프트웨어학부

Counting without numbers

- 숫자 없는 List

```
3.2example2.py
File Edit Format Run Options Window Help
animal = ['dog', 'cat', 'pig', 'wolf', 'tiger']
print(">> animal = ", animal, "\n")
for i in animal:
    print(">> animal list의 값을 하나씩 출력 : ", i)
```

```
>> animal = ['dog', 'cat', 'pig', 'wolf', 'tiger']
>> animal list의 값을 하나씩 출력 : dog
>> animal list의 값을 하나씩 출력 : cat
>> animal list의 값을 하나씩 출력 : pig
>> animal list의 값을 하나씩 출력 : wolf
>> animal list의 값을 하나씩 출력 : tiger
```

실습

- 숫자 출력

- 1) 0부터 9까지 출력하시오.
- 2) 1부터 10까지 출력하시오.
- 3) 1부터 10까지 수중에 짝수만 출력하시오.

```
# 0부터 9까지 출력  
0 1 2 3 4 5 6 7 8 9
```

```
# 1부터 10까지 출력  
1 2 3 4 5 6 7 8 9 10
```

```
# 1부터 10까지의 숫자중에 짝수만 출력  
2 4 6 8 10  
2 4 6 8 10
```

```
# 1부터 10까지 홀수만 출력  
1 3 5 7 9
```

print_range_1.py

File Edit Format Run Options Window Help

```
print("# 0부터 9까지 출력")  
for i in range(10) :  
    print(i, end = ' ')
```

```
print("\n\n# 1부터 10까지 출력")  
for i in range(1, 11) :  
    print(i, end = ' ')
```

```
print("\n\n# 1부터 10까지의 숫자중에 짝수만 출력")  
for i in range(2, 12, 2):  
    print(i, end = ' ')
```

```
print()  
for i in range(2, 11, 2):  
    print(i, end = ' ')
```

```
print("\n\n# 1부터 10까지 홀수만 출력")  
for i in range(1, 11, 2) :  
    print(i, end = ' ')
```

실습

- 소수(prime number) 판별

- 자연수 p를 입력 받고, p가 소수(1과 자신만으로 나누어지는 수)인지를 판별하는 프로그램을 작성하시오.
 - 예) 96818971 는 소수인가?
- 계산 시간을 측정 하시오.

```
# 시간과 관련된 기능을 가져옵니다.
import time

n = int(input("자연수 >"))

# 소수를 구하기전 시간을 저장 합니다.
start_time = time.time()

# whatever you want

# 걸린 시간을 출력합니다.
print("걸린 시간 :", time.time() - start_time)
```


실습

• 소수판별

- 자연수 p를 입력 받고, p가 소수(1과 자신만으로 나누어지는 수)인지를 판별하는 프로그램을 작성하시오.
 - 예) 96818971 는 소수인가?
- 계산 시간을 측정 하시오.

prime number_1.py

File Edit Format Run Options Window Help

시간과 관련된 기능을 가져옵니다.

import time, math

n = int(input(">> 자연수를 입력하세요 : "))

print(">> n = ", n)

소수를 구하기전 시간을 저장합니다.

start_time = time.time()

print("\n1) start time = ", start_time)

flag = 0 # 0인 경우 소수를 의미. 우선은 0으로 setting

print("2) flag = 0. 소수로 가정")

for i in range(2, n) :

#for i in range(2, int(math.sqrt(n)) + 1) :

print("3) i = ", i)

if n % i == 0 :

print("4) In if 문 : i = ", i)

print("5) {N}은 {i}로 나누어 떨어집니다.".format(N = n, i = i))

flag = 1 # 소수가 아님

print("6) flag = 1. 소수가 아님으로 값 변경")

break

if flag == 0 : # 소수

print("7) {N}은 소수입니다.".format(N = n))

else :

print("8) {N}은 소수가 아닙니다.".format(N = n))

걸린 시간을 출력합니다.

print("\n9) ending time = ", time.time())

print("10) 계산에 걸린 시간 :", time.time() - start_time)

>> 자연수를 입력하세요 : 12

>> n = 12

1) start time = 1586270643.157042

2) flag = 0. 소수로 가정

3) i = 2

4) In if 문 : i = 2

5) 12은 2로 나누어 떨어집니다.

6) flag = 1. 소수가 아님으로 값 변경

8) 12은 소수가 아닙니다.

9) ending time = 1586270643.2282884

10) 계산에 걸린 시간 : 0.08746504783630371

>>>

>> 자연수를 입력하세요 : 11

>> n = 11

1) start time = 1586270650.0078652

2) flag = 0. 소수로 가정

3) i = 2

3) i = 3

3) i = 4

3) i = 5

3) i = 6

3) i = 7

3) i = 8

3) i = 9

3) i = 10

7) 11은 소수입니다.

9) ending time = 1586270650.1840606

10) 계산에 걸린 시간 : 0.1921708583831787

>>>

실습

- 소수판별

- 자연수 p 를 입력 받고, p 가 소수(1과 자신만으로 나누어지는 수)인지를 판별하는 프로그램을 작성하시오.
 - 예) 96818971 는 소수인가?
- 반복 횟수를 $p/2$ 보다 적게 하시오.
 - 힌트: python 제곱근 함수 사용법

```
import math  
n = 10  
print (math.sqrt(n))
```

실습

- 소수판별
 - [증명] n 에 대한 소수판별은 \sqrt{n} 이하의 정수로만 나누면 된다.
 - “ n 이 소수가 아니면, n 은 \sqrt{n} 이하의 숫자들을 약수로 갖는다.” 증명
 - \sqrt{n} 이하의 숫자를 대입했을 때 나누어 떨어짐이 확인됨
 - $n=a*b$ (소수가 아니라고 가정)
 - a 와 b 가 모두 \sqrt{n} 보다 큰 숫자라고 가정하면, $a*b$ 는 n 보다 큰 수가 됨 → 모순 발생 → a 와 b 가 모두 \sqrt{n} 보다 큰 숫자는 아니다가 증명됨 → 적어도 하나는 \sqrt{n} 이하의 수임이 증명됨 → 2부터 \sqrt{n} 까지만 나누어 보면 됨

실습

- 소수판별

- 기존 코드에
비해서 얼마나
빨라 졌나요?

prime number_1.py

File Edit Format Run Options Window Help

시간과 관련된 기능을 가져옵니다.

import time, math

n = int(input(">> 자연수를 입력하세요 : "))

print(">> n = ", n)

소수를 구하기전 시간을 저장합니다.

start_time = time.time()

print("\n1) start time = ", start_time)

flag = 0 # 0인 경우 소수를 의미. 우선은 0으로 setting

print("\n2) flag = 0. 소수로 가정")

#for i in range(2, n) :

for i in range(2, int(math.sqrt(n)) + 1) :

print("\n3) i = ", i)

if n % i == 0 :

print("\n4) In if 문 : i = ", i)

print("\n5) {N}은 {i}로 나누어 떨어집니다.".format(N = n, i = i))

flag = 1 # 소수가 아님

print("\n6) flag = 1. 소수가 아님으로 값 변경")

break

if flag == 0 : # 소수

print("\n7) {N}은 소수입니다.".format(N = n))

else :

print("\n8) {N}은 소수가 아닙니다.".format(N = n))

걸린 시간을 출력합니다.

print("\n9) ending time = ", time.time())

print("\n10) 계산에 걸린 시간 :", time.time() - start_time)

>> 자연수를 입력하세요 : 12

>> n = 12

1) start time = 1586271145.2944808

2) flag = 0. 소수로 가정

3) i = 2

4) In if 문 : i = 2

5) 12은 2로 나누어 떨어집니다.

6) flag = 1. 소수가 아님으로 값 변경

8) 12은 소수가 아닙니다.

9) ending time = 1586271145.3796635

10) 계산에 걸린 시간 : 0.1020817756652832

>>>

>> 자연수를 입력하세요 : 11

>> n = 11

1) start time = 1586271160.3301127

2) flag = 0. 소수로 가정

3) i = 2

3) i = 3

7) 11은 소수입니다.

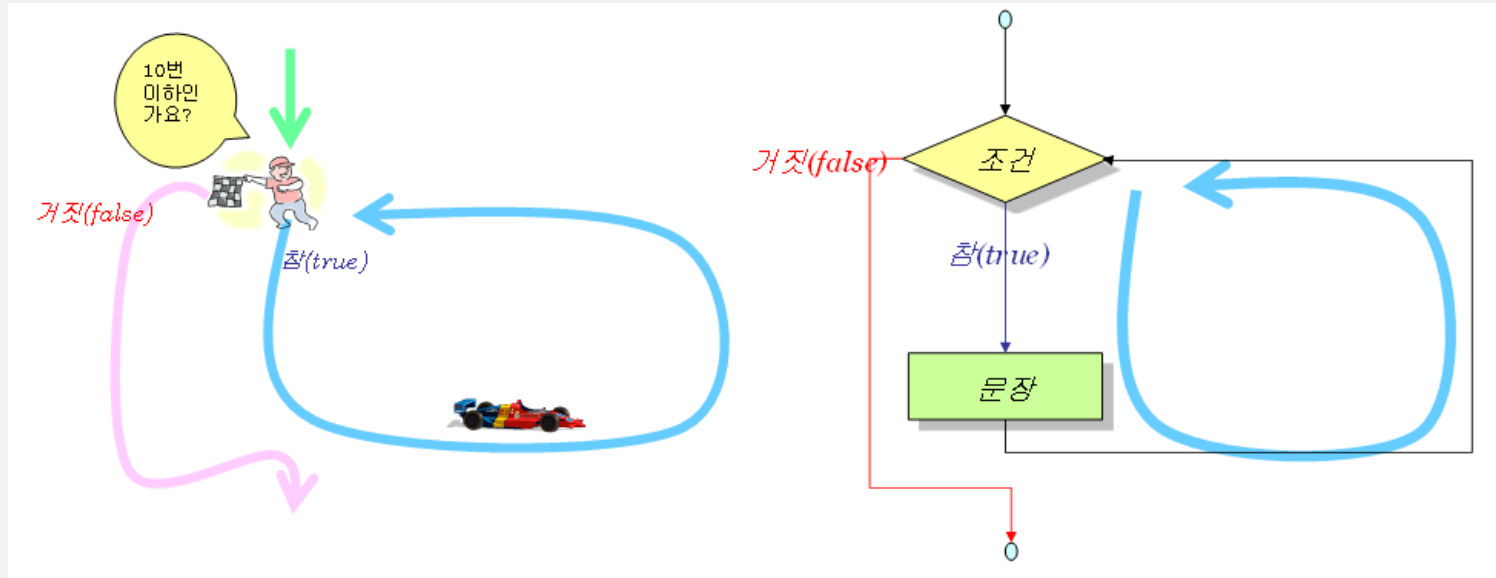
9) ending time = 1586271160.4139273

10) 계산에 걸린 시간 : 0.10473513603210449

...

while loops

- while 문은 조건을 정해놓고 반복을 하는 구조이다.



while loops

- while loops

while 조건문:
실행문

```
3.2example7.py
File Edit Format Run Options Window Help
someInput = input(">> Type 3 to continue, anything else to quit : ")
while someInput == '3': # loop condition
    print(">> Thank you for the 3. Very kind of you.")
    someInput = input("\n>> Type 3 to continue, anything else to quit : ")
print("\n\n>> That's not 3, so I'm quitting now.")
```

```
>> Type 3 to continue, anything else to quit : 3
>> Thank you for the 3. Very kind of you.

>> Type 3 to continue, anything else to quit : 2
```

while loops : 배수의 합 계산 프로그램

- 1부터 100사이의 모든 7의 배수의 합을 계산하여 출력하는 프로그램을 반복 구조를 사용하여 작성하라.

1부터 100 사이의 모든 7의 배수의 합은 735 입니다.

While loops

- 실습

- 은행 이율이 2.5%일 때, 100만원으로 200만원을 만들려면 몇 년을 저금해야 하는지 while 문으로 계산해보자.

While loops

- 실습

- 은행 이율이 2.5%일 때, 100만원으로 200만원을 만들려면 몇 년을 저금 해야 하는지 while 문으로 계산해보자.

```
interest.py
File Edit Format Run Options Window Help
year = 0
balance = 1000000

print("\n>>> 초기 입금액 = {}원".format(balance), "\n")

while balance < 2000000 :
    year = year + 1
    balance = int(balance * 1.025)
    print(">> {}년 후 {}원".format(year, balance))
```

```
>>> 초기 입금액 = 1000000원
>> 1년 후 1024999원
>> 2년 후 1050623원
>> 3년 후 1076888원
>> 4년 후 1103810원
>> 5년 후 1131405원
>> 6년 후 1159690원
>> 7년 후 1188682원
>> 8년 후 1218399원
>> 9년 후 1248858원
>> 10년 후 1280079원
>> 11년 후 1312080원
>> 12년 후 1344881원
>> 13년 후 1378503원
>> 14년 후 1412965원
>> 15년 후 1448289원
>> 16년 후 1484496원
>> 17년 후 1521608원
>> 18년 후 1559648원
>> 19년 후 1598639원
>> 20년 후 1638604원
>> 21년 후 1679569원
>> 22년 후 1721558원
>> 23년 후 1764596원
>> 24년 후 1808710원
>> 25년 후 1853927원
>> 26년 후 1900275원
>> 27년 후 1947781원
>> 28년 후 1996475원
>> 29년 후 2046386원
```

While loops

- 실습
 - 여러분이 가지고 있는 컴퓨터가 3초 동안 몇 번의 덧셈 연산을 반복하는지 실험해보자.
 - 이 숫자가 컴퓨터의 계산 능력을 실제로 반영하는가? 아니면 이러한 측정 방식의 문제는 무엇인가?

While loops

- 실습

- 여러분이 가지고 있는 컴퓨터가 3초 동안 몇 번의 덧셈 연산을 반복하는지 실험해보자.
- 이 숫자가 컴퓨터의 계산 능력을 실제로 반영하는가? 아니면 이러한 측정 방식의 문제는 무엇인가?

```
no_of_counter.py
File Edit Format Run Options Window Help
# 시간과 관련된 기능을 가져옵니다.
import time

# n(초)를 설정합니다.
n = 3
cnt = 0

target_time = time.time() + n
print(">> 현재시간 :", time.time())

while time.time() < target_time :
    cnt += 1

print("\n>> {N}초 동안 {CNT} 만큼 실행".format(N = n, CNT = cnt))
print(">> 실행 후 시간 :", time.time())
```

```
>> 현재시간 : 1586778646.7749228
```

```
>> 3초 동안 14235198 만큼 실행
```

```
>> 실행 후 시간 : 1586778649.8526745
```

while loops : 숫자 맞추기 게임

1부터 100 사이의 숫자를 맞추시오

숫자를 입력하시오: 50

낮음!

숫자를 입력하시오: 75

낮음!

숫자를 입력하시오: 82

낮음!

숫자를 입력하시오: 91

높음!

숫자를 입력하시오: 86

낮음!

숫자를 입력하시오: 87

축하합니다. 시도횟수= 6

while loops : Solution

```
guess.py
File Edit Format Run Options Window Help

import random

tries = 0
number = random.randint(1, 100)

print("\n>>> 1부터 100 사이의 숫자를 맞추시오. 10번의 기회가 있습니다.\n")

while tries < 10:
    guess = int(input(">> 숫자를 입력하시오: "))
    print()

    tries = tries + 1

    print(">> {N} 번째 시도 : " .format(N = tries), end='')
    if guess < number:
        print(" 낮음 !")
    elif guess > number:
        print(" 높음!")
    else:
        break


if guess == number:
    print("\n>> 축하합니다. 시도횟수 = ", tries)

else:
    print("\n>> 아쉽게 되었습니다. 정답은 {} 입니다" .format(number))
```

```
>>> 1부터 100 사이의 숫자를 맞추시오. 10번의 기회가 있습니다.
>> 숫자를 입력하시오: 50
>> 1 번째 시도 : 높음!
>> 숫자를 입력하시오: 25
>> 2 번째 시도 : 낮음 !
>> 숫자를 입력하시오: 36
>> 3 번째 시도 : 낮음 !
>> 숫자를 입력하시오: 44
>> 4 번째 시도 : 높음!
>> 숫자를 입력하시오: 40
>> 5 번째 시도 :
>> 축하합니다. 시도횟수 = 5
...
!
```

Bailing out of a loop – break and continue

- for loop 이나 while loop이 끝나기 전에 빠져 나오고 싶을 때
 - **continue**
 - 현재 반복 중지 후 다음 반복 시행 (바로 for loop으로 올라감)

 3.2example8.py

File Edit Format Run Options Window Help

```
for i in range (1, 6):
```

```
    print ('>> i =', i, " :", end=" ")
    print (' Hello, how', end=" ")
```

```
    if i == 3:
```

```
        print("\n\n>>> i가 {} 일때 continue 문 실행. for로 돌아갑니다." .format(i), "\n\n")
        continue
```

```
    print ('are you ?')
```

```
>> i = 1 : Hello, how are you ?
>> i = 2 : Hello, how are you ?
>> i = 3 : Hello, how
```

```
>>> i가 3 일때 continue 문 실행. for로 돌아갑니다.
```

```
>> i = 4 : Hello, how are you ?
>> i = 5 : Hello, how are you ?
... i
```

Bailing out of a loop – break and continue

- for loop 이나 while loop이 끝나기 전에 빠져 나오고 싶을 때
 - **break**
 - 현재 반복문을 중지하고 빠져나감

```
3.2example9.py
File Edit Format Run Options Window Help
for i in range (1, 6):
    print ('>> i =', i, end=" ")
    print ('Hello, how', end=" ")

    if i == 3:
        print("\n>> i = {} 일때, for loop실행을 중지하고 exit합니다.".format(i))
        break

    print ('are you ?')

print("\n>> for loop을 빠져나왔습니다,")

>> i = 1 Hello, how are you ?
>> i = 2 Hello, how are you ?
>> i = 3 Hello, how
>> i = 3 일때, for loop실행을 중지하고 exit합니다.
>> for loop을 빠져나왔습니다,
```

Loop else

- Python은 loop 문에서 else 지원
 - Conditional statement가 false일 때 else문 수행
 - 마지막에 수행

3.2example10.py

File Edit Format Run Options Window Help

```
for i in range (1, 6):
```

```
    print('>> i =', i, end=" ")
    print('Hello, how', end=" ")
    print('are you ?')
```

```
else:
```

```
    print("\n>> Conditional statement가 false일 때 else문 수행")
    print('>> else : i =', i)
```

```
>> i = 1 Hello, how are you ?
>> i = 2 Hello, how are you ?
>> i = 3 Hello, how are you ?
>> i = 4 Hello, how are you ?
>> i = 5 Hello, how are you ?
```

```
>> Conditional statement가 false일 때 else문 수행
>> else : i = 5
```


Loop else

- Python은 loop 문에서 else 지원
 - Conditional statement가 false일 때 else문 수행
 - 마지막에 수행
 - 단, break를 만나면 else문 수행하지 않음

3.2example10-1.py

File Edit Format Run Options Window Help

```
for i in range (1, 6):
```

```
    print('>> i =', i, end=" ")
    print('Hello, how', end=" ")
```

```
    if i == 3 :
```

```
        print("\n>>> break 문을 만나면 else:를 실행하지 않습니다.")
        break
```

```
    print('are you ?')
```

```
else:
```

```
    print("\n>> Conditional statement가 false일 때 else문 수행")
    print('>> else : i =', i)
```

```
print("\n>> for 문을 탈출했습니다.")
```

```
>> i = 1 Hello, how are you ?
```

```
>> i = 2 Hello, how are you ?
```

```
>> i = 3 Hello, how
```

```
>>> break 문을 만나면 else:를 실행하지 않습니다.
```

```
>> for 문을 탈출했습니다.
```

```
^^^
```

실습

- 유클리드 알고리즘 (Euclidean algorithm)
 - 최대공약수(Greatest Common Factor, Greatest Common Denominator) 구하기
 - $40(=2*2*2*5)$ 와 $24(=2*2*2*3)$ 의 최대 공약수 = $8(=2*2*2)$
 - 정수 a 와 b 의 최대 공약수 ($a \geq b > 0$)를 $G(a,b)$ 라고 하면,
 - if $b==0$
 $G(a,b) = a,$
 - else: $G(a,b) = G(b, a \% b)$
 - 최대공약수 구하기.py

실습

- 유클리드 알고리즘

- 증명

1. $G(a,b)=k$ 라고 하자 $\rightarrow a=m*k, b=n*k$ (m 과 n 은 서로 소)
2. $a=b*q+r$ (quotient : 몫, residue: 나머지) 라고 하면,
 - Python 문법으로는, $r=a\%b, q=a//b$ 성립
3. $m*k=n*k*q+r \rightarrow r=(m-n*q)*k \rightarrow r$ 은 k 의 배수. 따라서 k 는 b 와 r 의 공약수
4. k 가 b 와 r 의 최대공약수임을 증명해야 함!
 - 위에서 $b=n*k$ 이며 $r=(m-n*q)*k$ 이므로, k 가 b 와 r 의 최대공약수임을 증명하기 위해서는 n 과 $(m-n*q)$ 가 서로 소임을 증명하면 됨
 - 귀류법 사용
 - 1) n 과 $(m-n*q)$ 가 서로 소가 아니라고 가정하자. 두 수의 최대공약수를 G 라고 하면, $n=x*G, m-n*q=y*G$ (x 와 y 는 서로 소)
 - 2) 그런데, $m-n*q=m-x*G*q=y*G \rightarrow m=G*(x*q+y) \rightarrow m$ 과 n 이 G 를 공통으로 갖게 됨 \rightarrow 1.에서 m 과 n 은 서로 소라고 했던 부분과 모순 발생
 - 3) 따라서, 1)의 가정은 틀렸다 $\rightarrow n$ 과 $(m-n*q)$ 는 서로 소이다!
 - 4) 따라서, k 는 b 와 r 의 최대공약수이다. 증명 끝.

실습

- 유클리드 알고리즘 (Euclid algorithm)
 - 정수 a 와 b 의 최대 공약수 ($a \geq b > 0$)를 $G(a,b)$ 라고 하면,
 1. if $b == 0$
 - 종료. a 가 최대공약수
 2. $a = b, b = a \% b$
 3. 1,2반복

실습

- 유클리드 알고리즘
 - 다음을 이용해서 알고리즘을 개선하시오
 - while(b)
 - a, b = b, a%b

```
Euclid .py
File Edit Format Run Options Window Help
print(">> 두개의 수를 입력하시오. 큰 수를 먼저 입력하시요. 작은 수는 다음으로 입력하시오.")

large = int(input(">>> 큰 수 (large) : "))
small = int(input("작은 수 (small) : "))

print("\n>> large = {}, small = {}".format(large, small))
print("\n>> {L}와 {S}의 최대공약수(GCD) : ".format(L = large, S = small), end="")

while(small != 0):
    x = large
    large = small
    small = x%large

print(large)
```

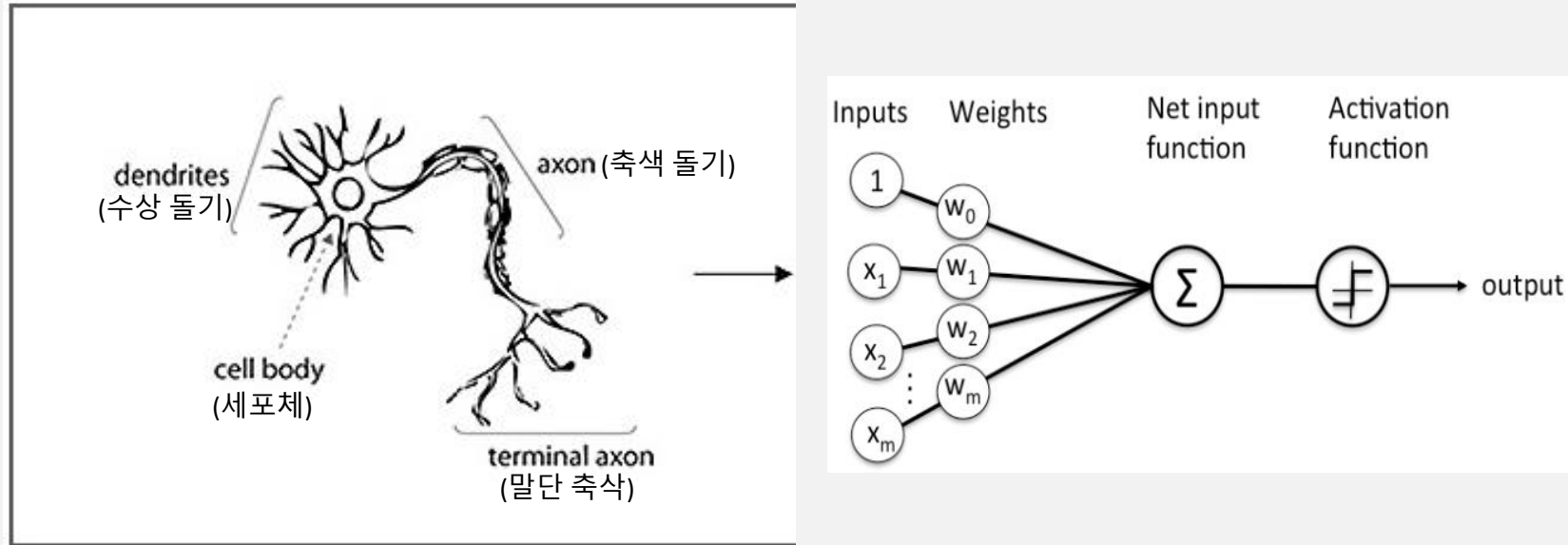
```
----
>> 두개의 수를 입력하시오. 큰 수를 먼저 입력하시요. 작은 수는 다음으로 입력하시오.
>>> 큰 수 (large) : 40
        작은 수 (small) : 24

>> large = 40, small = 24

>> 40와 24의 최대공약수(GCD) : 8
```

실습

- 활성화 함수 (activation function)
 - 인공신경망 (artificial neural network)
 - 활성화 함수(activation)에 의해 신호 생성 여부 결정



실습

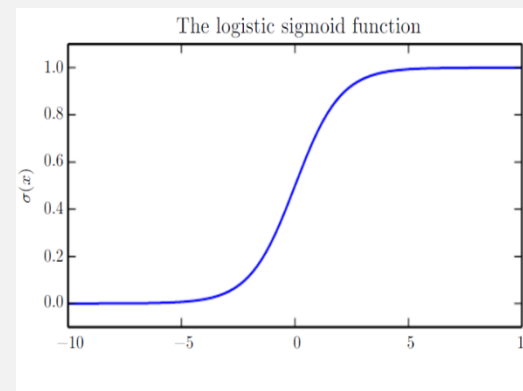
- 활성화 함수
 - Sigmoid 함수

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- 아래 식을 for loop으로 구현하고, 시그모이드 함수도 구현하시오.
 - 항의 개수는 10개로 한정함

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

```
import math  
  
print(math.factorial(3))
```



숙제

- 요일 구하기 (Finding the day of the week)
 - 사용자로부터 날짜(연,월,일) 정보를 입력 받아서 해당 날짜에 해당하는 요일(월~일)을 출력하는 프로그램을 작성하시오.
 - 힌트: 1년 1월 1일은 월요일이다.

숙제

- 요일 구하기
 - 예) 2015년 3월 3일 → 화요일
 - 1년 1월 1일은 월요일이므로, 2015년 3월 3일까지 몇 일이 지났는지 알아내면(이 값을 n 이라고 하자), $(n \% 7)$ 의 값을 구하면 요일을 알아낼 수 있다
 - $(n \% 7) == 0 \rightarrow$ 월요일
 - $(n \% 7) == 1 \rightarrow$ 화요일
 - $(n \% 7) == 2 \rightarrow$ 수요일
 - $(n \% 7) == 3 \rightarrow$ 목요일
 - $(n \% 7) == 4 \rightarrow$ 금요일
 - $(n \% 7) == 5 \rightarrow$ 토요일
 - $(n \% 7) == 6 \rightarrow$ 일요일

숙제

- 요일 구하기

- 1년 1월 1일~2015년 3월 3일까지 날짜수 = (1년 날짜수+2년 날짜수+...2014년 날짜수) + 2015년 3월 3일까지의 날짜수
 - 연도별 날짜수는 윤년에 따라 변함
 - 윤년일 때는 366일
 - 윤년이 아닐 때는 365일
 - 2015년 3월 3일까지의 날짜수는 달과 윤년에 따라 변함
 - 1월: 31일
 - 2월: 28일(윤년이 아닐때), 29일(윤년일때)
 - 3월: 31일
 - 4월: 30일
 - 5월: 31일
 - 6월: 30일
 - 7월: 31일
 - 8월: 31일
 - 9월: 30일
 - 10월: 31일
 - 11월: 30일
 - 12월: 31일

Homework

- 다음 수업 시작 전까지 ecampus로 제출
- 제출방법

파일명 :

이름-학번-Euclidean-algorithm

이름-학번-activation-function

이름-학번-finding-the-day-of-the week

- 파일이 여러 개일 경우 zip으로 묶어서 제출