단국대학교
SW중심대학

창의적 사고와 코딩

Lecture 4. 반복

반복의 필요성

- 회사에 중요한 손님이 오셔서 화면에 "환영합니다."을 5번 출력해야 한다고 가정하자. 이제까지 학습한 방법만을 사용하면 다음과 같이 print() 함수를 호출하는 문장을 5번 되풀이해야 한다.
- 일련의 명령문을 반복적으로 수행 할 수 있는 구조가 필요

```
print("환영합니다.")
print("환영합니다.")
print("환영합니다.")
print("환영합니다.")
print("환영합니다.")
```

같은 처리가 반복

```
환영합니다.
환영합니다.
환영합니다.
환영합니다.
환영합니다.
>>>
```

반복의 필요성

- 만약 1000번 반복해야 한다면?

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

1000번 복사해서 붙여넣기 ?

반복 구조를 사용하면

```
for x in range(5) :
    print("환영합니다.")
```

```
환영합니다.
환영합니다.
환영합니다.
환영합니다.
환영합니다.
>>>
```

Loops : 명령문의 반복

세 가지 종류의 Loop

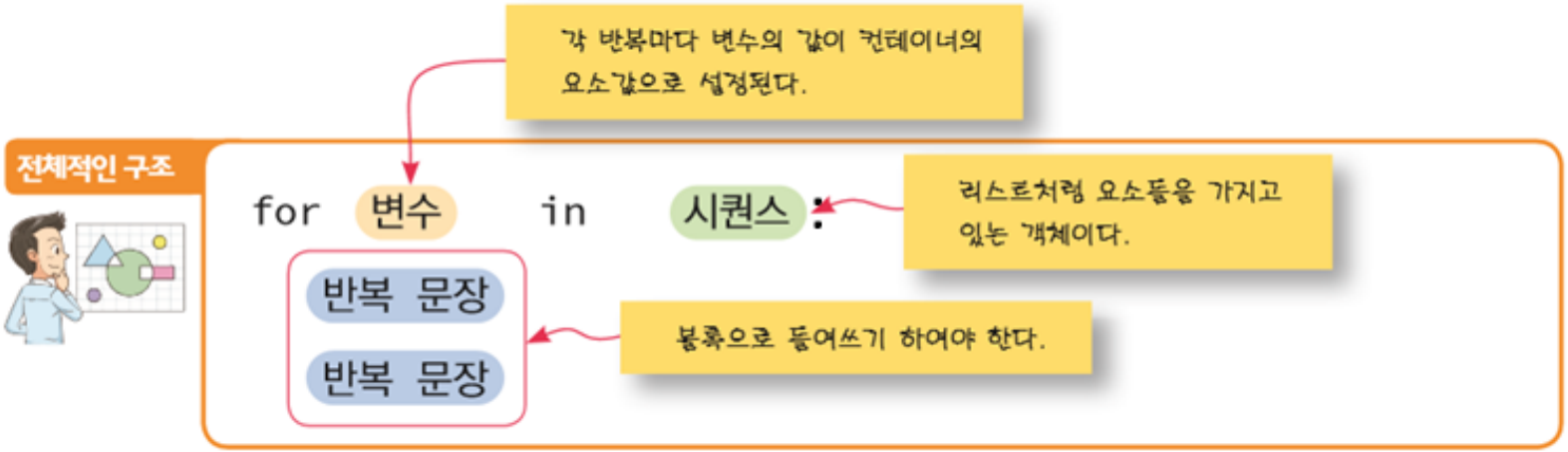
- 카운트 제어 루프(Count-controlled for loop) : 가장 일발적인 유형
  - 특정 횟수만큼 루프를 반복하기 위한 구조
  - 예: C언어 for(i = 0; i<= n; i++) , 초기화 부분, 종료 표현식, 증감 식
  - 파이썬은 이런 종류의 루프를 알지 못한다.
- 조건 제어 루프(Condition-controlled loop)
  - 주어진 조건이 변경(루프의 종류의 따라 True -> False, False -> True로)될 때까지 루프가 반복되는 구조
  - while loops(주어진 조건이 False가 될 때 까지 루프 구문 실행) , do while loops
- 콜렉션 제어 루프(Collection-controlled loop)
  - collection(배열, 리스트 또는 다른 ordered sequence)의 요소를 반복하는데 사용되는 특수한 구조
  - 예: bash shell 의 for loop , Perl의 foreach loop, Python 의 for loop

Loops : 명령문의 반복

- **파이썬은 두 가지 종류의 루프를 제공**
  - 콜렉션 제어 루프: for loop
    - 콜렉션(데이터 집합)의 항목의 개수 만큼 반복하는 구조
  - 조건 제어 루프: while loop
    - 주어진 조건이 만족되는 동안, 반복을 계속하는 구조

for 문(for Loop)

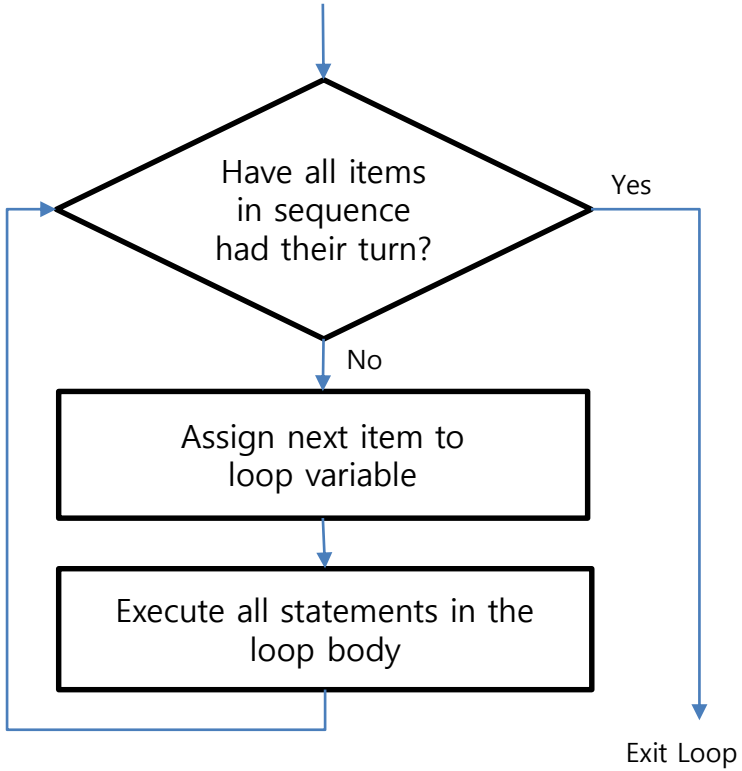
- for문은 반복의 횟수나 범위를 미리 알고 있을 경우 사용



- 시퀀스의 첫 번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 반복 문장들이 수행된다.
- 변수 : 루프 변수, 반복자 변수

for 문(for Loop)

for loop의 실행 흐름




```
code block
for x in S:
    loop code block
code block
```

==

```
code block
x = S[0]
loop code block
x = S[1]
loop code block
...
x = S[len(S)-1]
loop code block
code block
```

리스트를 이용한 반복

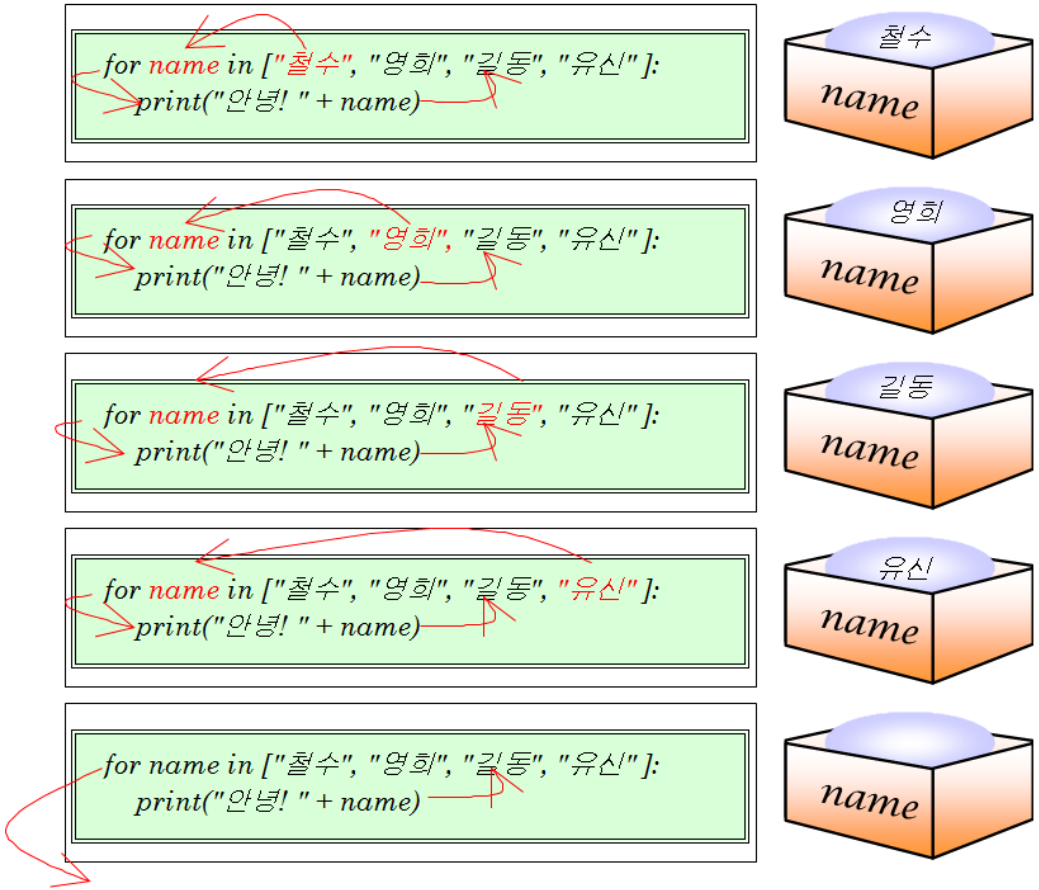
```
for name in ["철수", "영희", "길동", "유신" ]:
    print("안녕! " + name)
```

끝에 :이 있어야 함
들여쓰기 하여야 함

```
안녕! 철수
안녕! 영희
안녕! 길동
안녕! 유신
```

name이 '철수'부터 '유신'까지 변경되면서 반복된다.

리스트 반복 이해하기


for name in ["철수", "영희", "길동", "유신"]:
print("안녕! " + name)
철수
name
영희
name
길동
name
유신
name
name

리스트를 이용한 반복

* 리스트 에 있는 내용을 반복 !! *

```
for x in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] :
    print(x, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
>>>
```

print(x, end=“ ”) 함수 내에 end=" "는 x 출력후 다음 줄로 넘기지 말고 “ ”를 출력하라는 의미

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

```
hello
hello
hello
hello
hello
```

리스트를 이용한 반복

```
for val in [1, 2, 'a', "Python", '대한', "민국" ]:
    print(val)
```

```
lista = [1, 2, 'a', "Python", '대한', "민국" ]
for val in lista:
    print(val)
```

for val in [1, 2, 'a', "Python", '대한', '민국']:

출력결과

리스트를 이용한 반복

```
for j in ['서울', '부산', '대구']:
    print('도시 :', j)
```

```
도시 : 서울
도시 : 부산
도시 : 대구
```

```
city = ['서울', '부산', '대구']
no = 1

for i in city:
    print('도시 ',no,':', i)
    no+=1
```

```
city = ['서울', '부산', '대구']
no = 0

for i in city:
    no +=1
    print('도시 ',no,':', i)
```

```
도시  1 : 서울
도시  2 : 부산
도시  3 : 대구
```

```
city = ['서울', '부산', '대구']

for j in range(len(city)):
    print('도시', j+1, ':', city[j])
```

```
city = ['서울', '부산', '대구']

for j in range(1, len(city)+1):
    print('도시', j, ':', city[j-1])
```

문자열을 이용한 반복

- 문자열도 시퀀스의 일부분이다. 따라서 문자열을 대상으로 반복문을 만들 수 있다.

```
for c in  "abcdef":
            print(c, end=" ")
```

```
a b c d e f
```

```
for char in 'PYTHON':
    print(char)
```

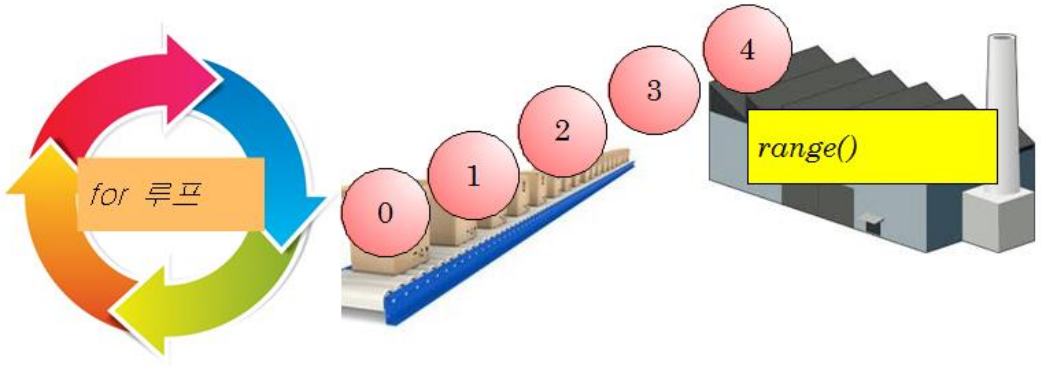
```
P
Y
T
H
O
N
>>>
```

'PYTHON'이 하나씩 char 변수에 저장되어 for 블록을 수행한다.

range() 함수

start(0) ~ stop - 1

- range() 함수를 이용하면 특정 구간의 정수들을 생성할 수 있다.
  예를 들어서 range(10)하면 0부터 9까지의 정수가 생성된다.




```
for x in range(10):
    print(x, end=' ')
```

```
0 1 2 3 4 5 6 7 8 9
```

range() 함수

- range(start, stop)와 같이 호출하면 start부터 시작하여서 (stop-1)까지의 정수가 생성된다. 이때 stop은 포함되지 않는다.

```
for x in range(1, 10):
    print(x, end=' ')
```

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

range() 함수 정리

- range() 함수는 start부터 stop-1까지 step의 간격으로 정수들을 생성한다. start와 step이 대괄호로 싸여져 있는데 이는 생략할 수 있다는 의미이다. start나 step이 생략되면 start는 0, step은 1로 간주된다.

전체적인 구조

```
range( [start ,] stop [, step ])
```

range() 함수 예제

```
range( [ start ,] stop [, step ])
```

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

list(A) : A를 리스트로 변환

```
>>> list('python')
['p', 'y', 't', 'h', 'o', 'n']
>>>
>>> a = range(10)
>>> type(a)
<class 'range'>
>>> a = list(a)
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

range() 함수 예제

```
for i in range(1, 10, 2) :
    print(i)
```

```
1
3
5
7
9
```

```
for i in range(10, 1, -1) :
    print(i)
```

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

yunyi96@gmail.com

Lab: 정수들의 합

- 1부터 사용자가 입력한 수 n까지 더해서 (1+2+3+...+n)을 계산하는 프로그램을 작성하여 보자. for 문을 사용하면 간명하게 합계를 구할 수 있다.

```
어디까지 계산할까요: 10
1부터  10 까지의 정수의 합=  55
```

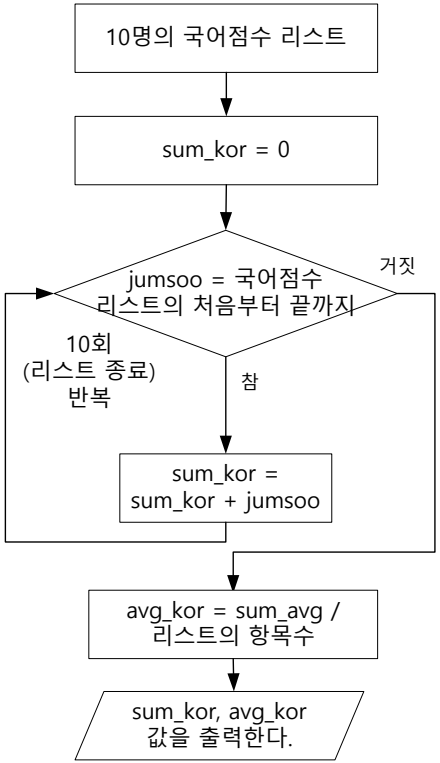
Solution: 파일명 for_sum.py

```
#  반복을 이용한 정수합 프로그램
sum = 0
limit=int(input("어디까지 계산할까요: "))
for i in range(1, limit+1) : # 1 ~ limit
    sum += i
print("1부터 ", limit, "까지의 정수의 합= ", sum)
#print("1부터 %d까지의 정수의 합= %d" % (limit,sum))
```

	i의 값	sum의 값
1번째 반복	1	0+1
2번째 반복	2	0+1+2
3번째 반복	3	0+1+2+3
…	…	…
10번째 반복	10	0+1+2+3+…+10

Lab 학생들의 국어 점수 합계와 평균 구하기

- 10명의 학생에 대한 국어 점수를 가지고 전체 합계와 평균을 구하는 프로그램을 작성해 보자.




```
kor_jumsoo = [98, 88, 65, 70, 83, 91, 62, 53, 75, 30]

sum_kor = 0

for jumsoo in kor_jumsoo:
    sum_kor = sum_kor + jumsoo        # 합계를 누적해 간다.

avg_kor = sum_kor / len(kor_jumsoo)  # 평균을 계산한다.

print("전체 합계 :", sum_kor)
print("전체 평균 :", avg_kor)
```

```
전체 합계 : 715
전체 평균 : 71.5
```

yunyi96@gmail.com

Lab: 팩토리얼 계산

- for문을 이용하여서 팩토리얼을 계산해보자. 팩토리얼 n!은 1부터 n까지의 정수를 모두 곱한 것을 의미한다. 즉, n! = 1×2×3×……×(n-1)×n이다.

```
10!은 3628800입니다.
```

Solution : 파일명 for_facto.py

Lab: 온도 변환 테이블 출력

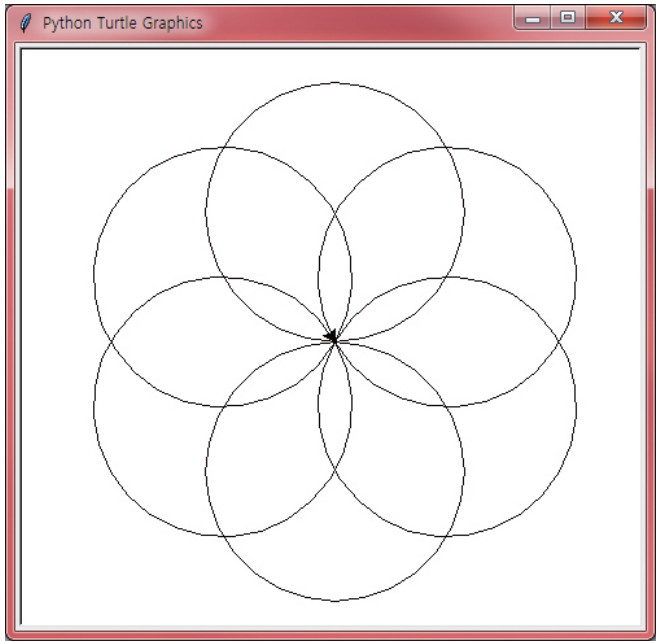
- 화씨온도-섭씨온도 변환 테이블을 출력하는 프로그램을 작성하여 보자.
- 반복 구조를 사용하여야 하고 정수보다는 실수로 출력하는 편이 정확하다. 화씨 0도부터 100도까지, 10도 단위로 증가시키면서 대응되는 섭씨온도를 옆에 출력한다.

$$C = (F-32)\times 5/9$$

```
0   -> -17.78
10  -> -12.22
20  -> -6.67
30  -> -1.11
40  -> 4.44
50  -> 10.0
60  -> 15.56
70  -> 21.11
80  -> 26.67
90  -> 32.22
100  -> 37.78
```

Solution : 파일명 temp_table.py

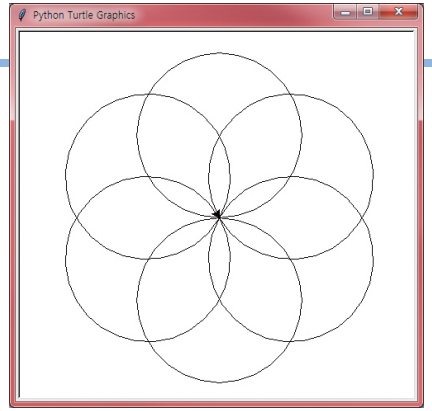
예제 : 6 개의 원 그리기



Solution

```
import turtle
t = turtle.Turtle()

t.circle(100)
t.left(360/6)
t.circle(100)
t.left(360/6)
t.circle(100)
t.left(360/6)
t.circle(100)
t.left(360/6)
t.circle(100)
t.left(360/6)
t.circle(100)
t.left(360/6)
```



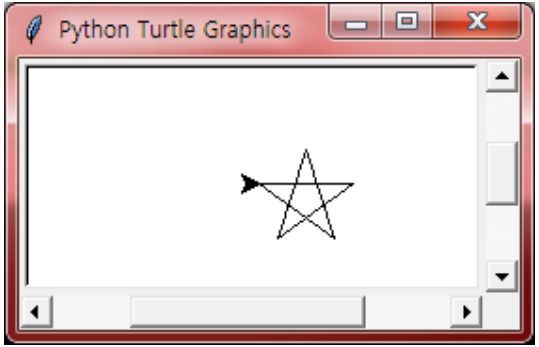

```
import turtle
t = turtle.Turtle()

for count in range(6):
        t.circle(100)
        t.left(360/6)
```

yunyi96@gmail.com

Lab: 화면에 별 그리기

- 터틀 그래픽을 이용하여 별을 화면에 그려보자.



Solution : 파일명 turtle_star.py

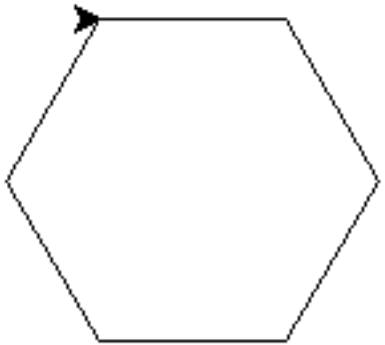
```
import turtle

star = turtle.Turtle()

for i in range(5):
    star.forward(50)
    star.right(144)
```

Lab: 화면에 다각형 그리기

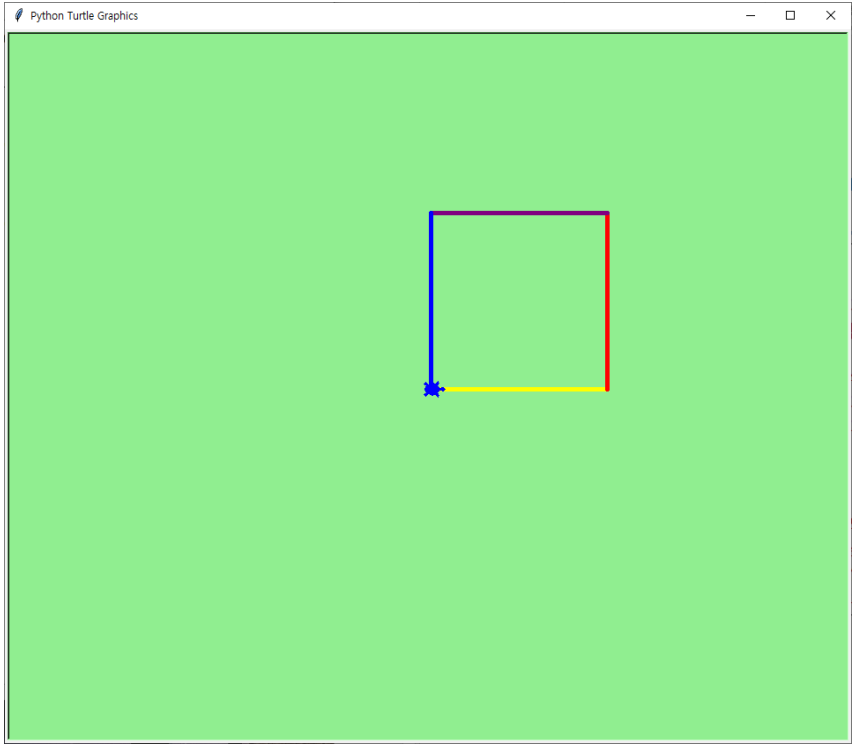
- 터틀 그래픽을 이용하여 다각형을 화면에 그려보자.



Solution : 파일명 turtle_polygon.py

Lab: 화면에 컬러 사각형 그리기

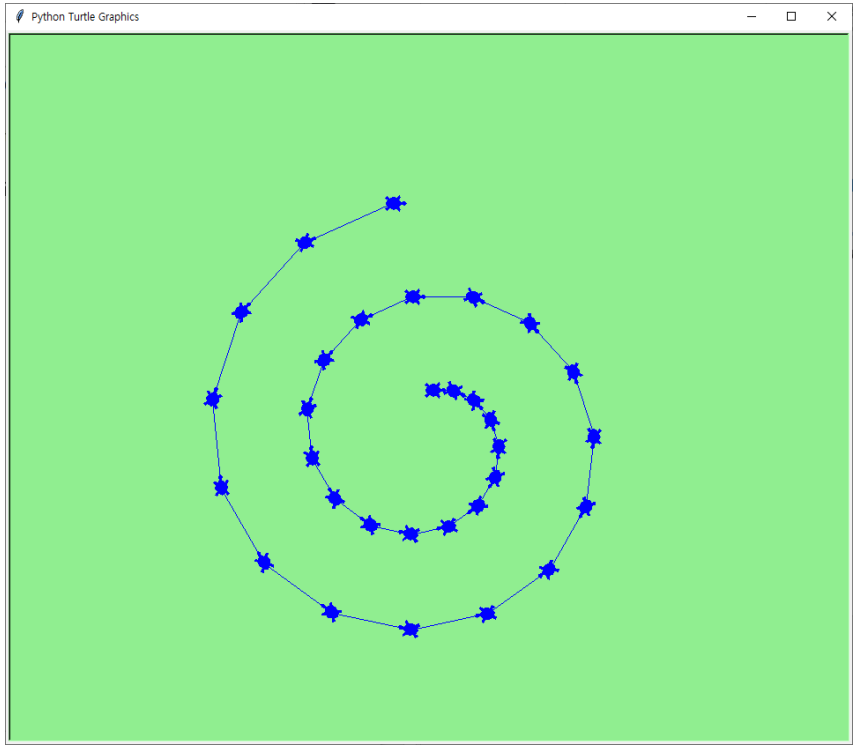
- **터틀 그래픽을 이용하여 컬러 사각형을 그려보자.**
  - 스크린 배경색 : lightgreen 로 설정
  - colors = ["yellow", "red", "purple", "blue"] colors 리스트를 이용하여 사각형의 선분을 그릴 때마다 다른 색으로 그려지도록 한다.
  - 그리는 펜의 두께 : 5 로 설정



Solution : 파일명 turtle_rect.py

Lab: 화면에 거북이 그리기

- 터틀 그래픽을 이용하여 거북이를 회전시키면서 화면에 그려보다.
- 터틀 객체의 stamp() 함수를 사용하면 화면에 터틀의 모양을 도장찍는 것처럼 출력할수 있다.

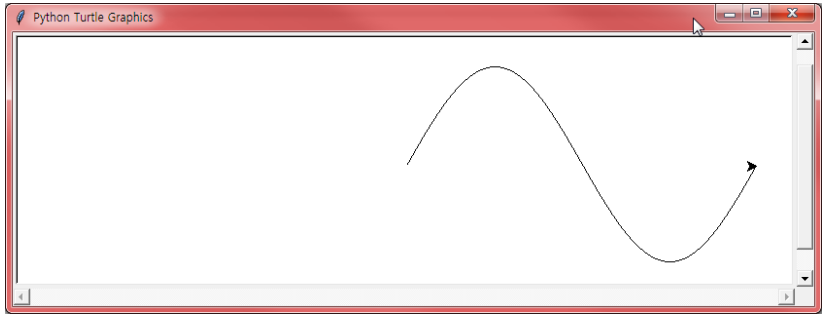


yunyi96@gmail.com

Solution : 파일명 turtle_spiral.py

Lab: 함수 그래프 그리기

- 싸인(sine) 그래프를 반복문을 이용하여서 그려보자. 싸인 함수는 수학, 물리학, 공학에서 아주 많이 나타나는 함수이다. 이번에도 터틀 그래픽의 기능을 사용하여 본다.



Solution : 파일명 turtle_sin.py

```
import math
import turtle

t = turtle.Turtle()

t.pendown()                                  # 터틀 객체의 펜을 내린다.
for angle in range(360):                     # 360번 반복한다.
    y = math.sin(math.radians(angle))  # angle 값에 대응되는 싸인값을 계산한다.
    scaledX = angle                        # x축의 좌표값을 각도로 한다.
    scaledY = y * 100                            # y축의 좌표값을 싸인값으로 한다.
    t.goto (scaledX, scaledY)              # 터틀 객체를 (scaledX, scaledY)로 이동시킨다.
t.penup()                                    # 터틀 객체의 펜을 올린다.
```