DAY 04

for 명령을 사용하여 똑같은 작업 반복하기

likegnu@Facebook

모두의 파이썬 20일 만에 배우는 프로그래밍 기초

길벗

for 명령을 사용하여 똑같은 작업 반복하기

01 Hello!를 반복하여 출력하는 프로그램

02 반복 블록을 알아보는 프로그램

03 반복 기능으로 도형을 그리는 프로그램

01

1. Hello!를 반복하여 출력하는 프로그램

- 다음은 Hello!를 10번 반복하는 구문이다.
- IDLE에서 실행시켜 결과를 확인하자.

```
for x in range(10):
    print("Hello!")          # 반복할 내용은 네 칸 띄고 입력(들여쓰기)
```

01

1. Hello!를 반복하여 출력하는 프로그램

» 실행결과

```
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
```

01

1. Hello!를 반복하여 출력하는 프로그램

들여쓰기는 몇 칸 해야 하나요?

- 콜론(:)을 입력한 후 Enter 를 누르면 다음 줄부터는 IDLE이 자동으로 들여쓰기를 해 줍니다.
- 따라서 다음 줄에 편하게 반복할 내용을 입력할 수 있습니다.

파이썬의 추천하는 표준 들여쓰기는 몇 칸인가요?

- 프로그램을 편집하다가 들여쓰기를 직접 해야 할 때는 SpaceBar 를 네 번 눌러 '네 칸 띄어쓰기'를 하는 것입니다.

01

1. Hello!를 반복하여 출력하는 프로그램

반복을 멈추려면?

range(10) 안의 값을 바꿔 보라고 하면 굉장히 큰 수를 입력해서 테스트하는 사람이 많습니다. 10을 100000으로 바꾸면 어떻게 될까요?

```
for x in range(100000):
  print("Hello!")
```

- Hello!가 십만 번 출력됩니다. 당연히 출력을 마칠 때까지 시간이 오래 걸립니다. 언제 끝날지 모른다고 봐야 합니다.

- 이럴 때는 파이썬 IDLE을 강제로 닫을 수도 있지만, 다음과 같은 방법으로 프로그램을 멈출 수도 있습니다.

  ① 대화형 셸(Python 3.x.x Shell 창)을 클릭하여 선택합니다.
  ② **Ctrl + C** 를 누릅니다

02

2. 반복 블록을 알아보는 프로그램

```
for x in range(3):
        print(100)        # 들여쓰기(빈칸 네 개 뒤에 입력)
        print(200)        # 들여쓰기(빈칸 네 개 뒤에 입력)
print(300)        # 내어 쓰기(빈칸 없이 바로 입력)
```

» 주의!

- 04B-block.py로 파일을 저장한 후 실행할 것.
- 대화형 셀에서는 들여쓰기 블록 입력, 인식이 힘듭니다.

» 예상한 실행 결과는?

2. 반복 블록을 알아보는 프로그램

```
for x in range(3):
        print(100)        # 들여쓰기(빈칸 네 개 뒤에 입력)
        print(200)        # 들여쓰기(빈칸 네 개 뒤에 입력)
print(300)        # 내어 쓰기(빈칸 없이 바로 입력)
```

실행 결과

```
100
200
100
200
100
200
300
```

02

2. 반복 블록을 알아보는 프로그램

블록(Block)이 무엇인가요?

- 파이썬에서는 들여쓰기 해서 입력한 문장의 묶음을 블록이라고 부릅니다.
- 콜론(:) 다음에 들여쓰기로 입력한 두 문장이 바로 '블록'입니다.

```
    print(100)
    print(200)
```

들여쓰기 되지 않은 마지막 문장은 블록에 속하지 않습니다.

```
print(300)
```

- 블록은 오늘 배운 반복 외에도 여러 곳에서 쓰입니다.

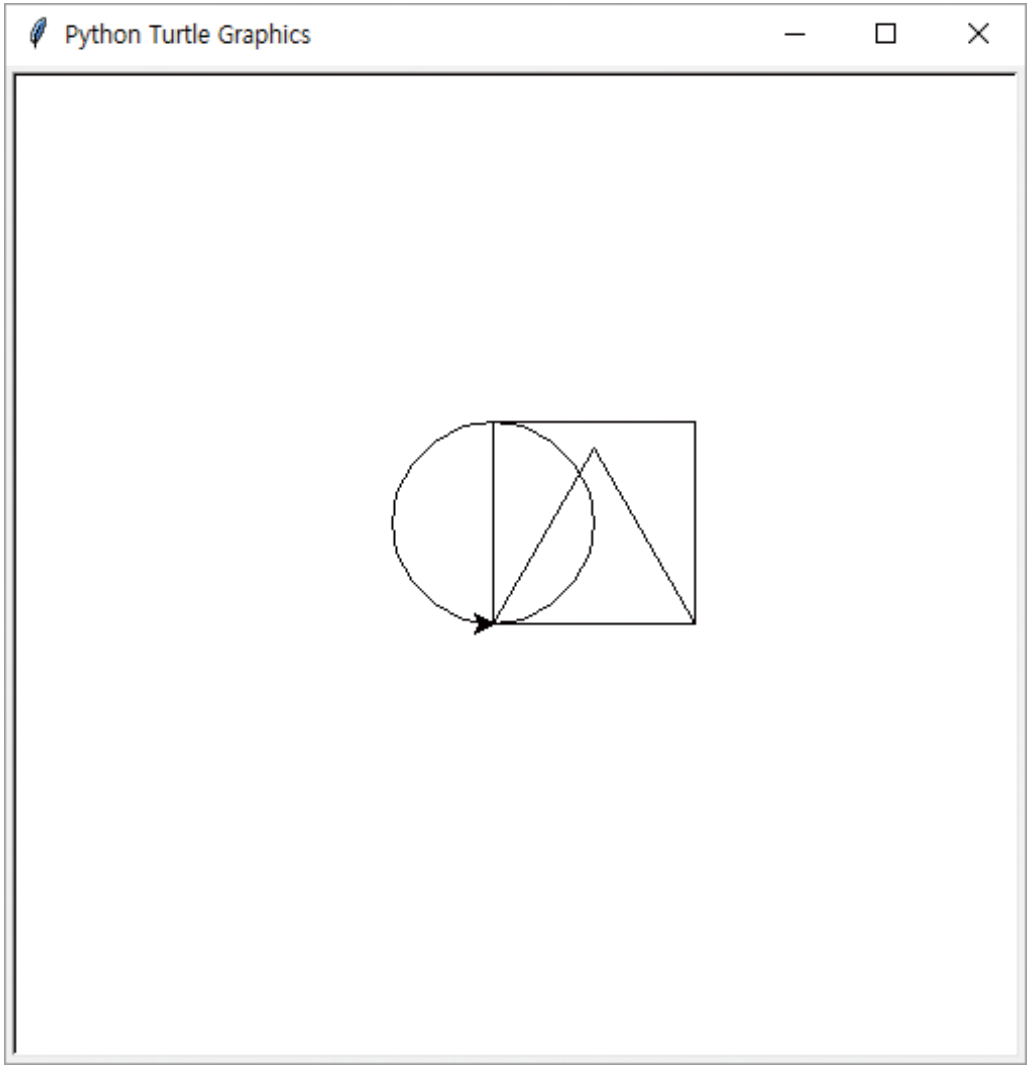
- 블록에서 기억할 사항!
  - **블록**: 프로그램의 실행에 사용되는 '프로그램의 한 묶음 또는 단위'

3. 반복 기능으로 도형을 그리는 프로그램

```
import turtle as t

# 삼각형 그리기
for x in range(3):          # 세 번 반복
    t.forward(100)          # 거북이가 100만큼 앞으로 이동
    t.left(120)             # 거북이가 왼쪽으로 120도 회전

# 사각형 그리기
for x in range(4):          # 네 번 반복
    t.forward(100)          # 거북이가 100만큼 앞으로 이동
    t.left(90)              # 거북이가 왼쪽으로 90도 회전

# 원 그리기
t.circle(50)                # 반지름이 50인 원을 그림
```

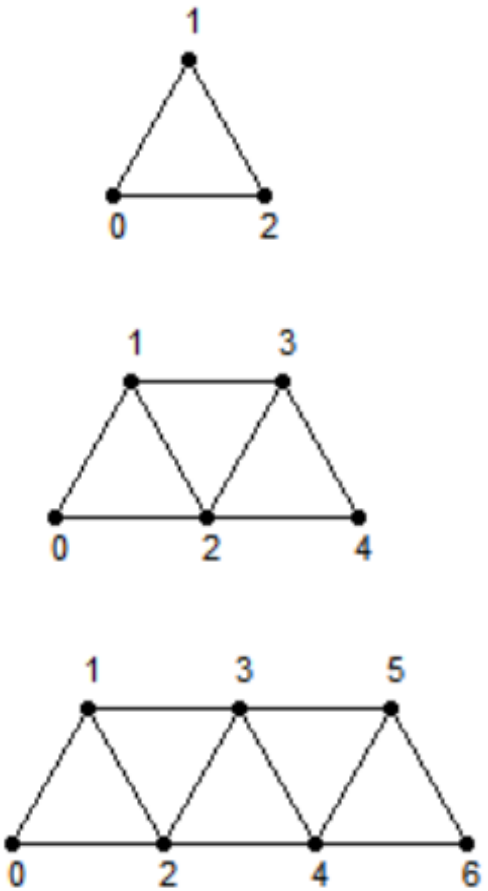
3. 반복 기능으로 도형을 그리는 프로그램

» 실행결과



숙제

- 다음과 같이 홀수 개의 삼각형 그리기



02

숙제에 참고할 명령어

자주 사용하는 거북이 그래픽 명령어

함수	설명	사용 예
forward(거리)/ fd(거리)	거북이가 앞으로 이동합니다.	t.forward(100) # 거북이가 100만큼 앞으로 이동합니다.
backward(거리) / back(거리)	거북이가 뒤로 이동합니다.	t.back(50) # 거북이가 50만큼 뒤로 이동합니다.
left(각도) / lt(각도)	거북이가 왼쪽으로 회전합니다.	t.left(45) # 거북이가 45도 왼쪽으로 회전합니다.
right(각도) / rt(각도)	거북이가 오른쪽으로 회전합니다.	t.right(45) # 거북이가 45도 오른쪽으로 회전합니다.
circle(반지름)	현재 위치에서 원을 그립니다.	t.circle(50) # 반지름이 50인 원을 그립니다.
down( ) / pendown( )	펜(잉크 묻힌 꼬리)을 내립니다.	t.down() # 이제 움직이면 그림이 그려집니다.
up( ) / penup( )	펜(잉크 묻힌 꼬리)을 올립니다.	t.up() # 거북이가 움직여도 선이 그려지지 않습니다.
shape("모양")	거북이 모양을 바꿉니다.	t.shape("turtle") # 진짜 거북이 모양으로 지정합니다. t.shape("arrow") # 화살표 모양의 거북이로 지정합니다. ※ 거북이 모양으로 "circle", "square", "triangle"을 사용할 수 있습니다.
speed(속도)	거북이 속도를 바꿉니다.	t.speed(1) # 가장 느린 속도 t.speed(10) # 빠른 속도 t.speed(0) # 최고 속도

DAY 05

range 명령을 사용하여 변화를 주면서 반복하기

likegnu@Facebook

모두의 파이썬 20일 만에 배우는 프로그래밍 기초

길벗

range 명령을 사용하여 변화를 주면서 반복하기

01 range란?

02 for 명령어로 숫자를 반복하여 출력하는 프로그램

03 1부터 10까지 숫자의 합계를 구하는 프로그램

04 변수 이름 정하기

01

1. range란?

- range : 파이썬에서의 range는 반복할 '범위'를 지정하는 명령어
- 대화형 셸에서 range를 확인하는 예제

```
>>> list(range(5))
[0, 1, 2, 3, 4]
```

range(5) : 0,1, 2, 3, 4로 값을 다섯 개 가짐.

for x in range(5) : '변수 x의 값을 차례대로 0, 1, 2, 3, 4로 바꾸면서 반복 블록을 실행'.

```
>>> list(range(0, 5))
[0, 1, 2, 3, 4]
```

range(a, b) :
range(a, b)의 값은 a에서 시작해서b 바로 앞의 값까지 1씩 늘리면서 반복.

range(0,5) :
0부터 시작해서 5 바로 앞의 값까지 반복
0, 1, 2, 3, 4를 출력.

01

1. range란?

- range : 파이썬에서의 range는 반복할 '범위'를 지정하는 명령어
- 대화형 셸에서 range를 확인하는 예제

```
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

range(1, 11) :
1에서 시작해서 11 바로 앞(10)까지를 반복
1, 2, 3, 4, 5, 6, 7, 8, 9, 10을 출력.

```
>>> list(range(1, 4))
[1, 2, 3]
```

range(1, 4) : 1에서 4 바로 앞(3)까지 반복
1, 2, 3을 출력.

02

2. for 명령어로 숫자를 반복하여 출력하는 프로그램

```
print("[0-4]")

for x in range(5):          # range(5)로 0, 1, 2, 3, 4까지 다섯 번 반복
    print(x)                # 변수 x 값을 출력

print("[1-10]")

for x in range(1, 11):      # 1, 2, ..., 10까지 열 번 반복(11은 제외).
    print(x)                # 변수 x 값을 출력
```

- range 명령어를 쓸 때 두 가지는 꼭 기억하세요!

① range(5) : 0부터 시작해서 4까지 다섯 번 반복한다(5는 제외한다).
② range(1, 11) : 1부터 시작해서 10까지 열 번 반복한다(11은 제외한다).

03

3. 1부터 10까지 숫자의 합계를 구하는 프로그램

```
s = 0                              # 합계를 구하는 변수 s, 처음 값은 0을 입력
for x in range(1, 11):             # 1, 2, ..., 10으로 열 번 반복(11은 제외).
    s = s+x                        # 현재의 s 값에 x 값을 더함
    print("x:", x, " sum:", s)     # 현재의 x 값과 s 값을 출력
```

실행결과

```
x: 1 sum: 1
x: 2 sum: 3
x: 3 sum: 6
x: 4 sum: 10
x: 5 sum: 15
x: 6 sum: 21
x: 7 sum: 28
x: 8 sum: 36
x: 9 sum: 45
x: 10 sum: 55
```

03

3. 1부터 10까지 숫자의 합계를 구하는 프로그램

```
s = 0                              # 합계를 구하는 변수 s, 처음 값은 0을 입력
for x in range(1, 10+1):           # 1, 2, ..., 10으로 열 번 반복(11은 제외).
    s = s+x                        # 현재의 s 값에 x 값을 더함
    print("x:", x, " sum:", s)     # 현재의 x 값과 s 값을 출력
```

실행결과

```
x: 1 sum: 1
x: 2 sum: 3
x: 3 sum: 6
x: 4 sum: 10
x: 5 sum: 15
x: 6 sum: 21
x: 7 sum: 28
x: 8 sum: 36
x: 9 sum: 45
x: 10 sum: 55
```

04

4. 변수 이름 정하기!

- 변수는 변할 수 있는 값, 즉 컴퓨터가 정보를 보관하는 곳입니다.

- 컴퓨터는 수많은 정보를 보관하고, 이러한 정보를 구분하려면 변수 이름이 필요하다고 배웠습니다.

- 우리는 지금까지 a, b, c, d, x, s처럼 매우 단순한 이름으로 변수 이름을 지었습니다.

- 변수 이름은 프로그램을 만드는 사람이 자유롭게 정할 수 있지만, **모든 이름을 변수 이름으로 쓸 수 있는 것은 아닙니다.**

04

4. 변수 이름 정하기!

» 변수 이름을 정할 때 다음과 같은 규칙을 따라야 합니다.

① 변수 이름은 **영문 대/소문자, 숫자, 밑줄(_)**로만 만들 수 있습니다.
변수 이름에는 공백을 사용할 수 없습니다.

② 변수 이름은 숫자로 시작할 수 없습니다.
변수 이름으로 3name이나 150m은 변수로 사용할 수 없습니다.

③ 영문 대/소문자를 구분합니다. 즉, A와 a는 다른 변수입니다.

④ 파이썬이 문법으로 사용하는 단어는 헷갈릴 수 있으므로 사용할 수 없습니다.
예: False, None, True, and, as, assert, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield 는 변수로 사용할 수 없습니다.

04

4. 변수 이름 정하기!

- 이러한 규칙을 지키면서 자유롭게 변수 이름을 정할 수 있습니다.

- 너무 길지 않은 것이 좋고, 변수 이름을 봤을 때 변수에 저장된 정보가 무엇인지 예상할 수 있는 이름이라면 더욱 좋습니다.

- 낙타등 표기법 (CamelCase)
  - 합성어에서 구분이 되는 부분을 대문자로 시작하고, 그 구분자 이외의 부분에서는 소문자표기를 함으로서 단어간 구분을 해주는 방식
  - 한개의 단어에서 대문자, 소문자가 울룩불룩하게 표기되면서 낙타의 등에 있는 혹처럼 보인다고 해서 CamelCase라고 부름
  - 예: McDonald, PlayStation, GetInputReader