



# 파이썬과 친해지기

time 모듈, range 함수 심화, 목록 변환,  
print 함수 심화

10주차

# 복습

while A:

A(조건)가 참인 동안 블록을 반복한다.

```
>>> while True:  
    print("안녕!")
```

안녕?

안녕?

안녕?

(무한반복)

# 복습

```
a = 0
while a < 3:
    print("꼬꼬댁!")
    a += 1
```



```
>>> %Run chicken.py
꼬꼬댁!
꼬꼬댁!
꼬꼬댁!
```

# 복습

```
while True:  
    print("꼬꼬댁!")
```



```
>>> %Run chicken.py
```

꼬꼬댁!

꼬꼬댁!

꼬꼬댁!

꼬꼬댁!

꼬꼬댁!

(무한반복)

# 복습

## break

반복문을 탈출한다.

```
>>> while True:  
    print("안녕?")  
    break  
    print("좋은 아침!")
```

안녕?

# 복습

```
a = 1
while True:
    a = a * 2
    if a > 1000:
        break
print(a)
```

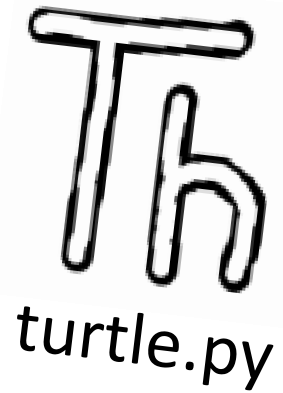


```
>>> %Run chicken.py
1024
```

# 모듈



→ 숫자를 임의로 뽑는다던지,  
뭔가 랜덤한 일을 할 수 있는 모듈



→ 거북이로 그림을 그려주는 모듈

# time 모듈

Th

time.py

“시간”에 관련된 일을 해주는 모듈

지금  
몇시 몇분 몇초?

5초 기다리기

오늘의 날짜는?





# time 모듈 불러오기

```
import time  
print("time 모듈 준비 완료!")
```

여느 모듈처럼 import 를 이용해 불러와주면 된다.

# time.sleep()

## time.sleep(A)

A 초 만큼 기다린다.

```
>>> import time
```

```
>>> time.sleep(5)
```

(5초간 멈춘다.)

```
>>> time.sleep(0.3)
```

(0.3초간 멈춘다.)

# time.sleep() 예제

```
import time
print("날씨 정보를 불러오는 중입니다...")
time.sleep(3)
print("날씨 정보 불러오기 완료!")
print("오늘은 치킨 시키기 좋은 날입니다.")
```

# time.sleep() 예제

```
import time
print("날씨 정보를 불러오는 중입니다...")
time.sleep(3)
print("날씨 정보 불러오기 완료!")
print("오늘은 치킨 시키기 좋은 날입니다.")
```



```
>>> %Run test.py
날씨 정보를 불러오는 중입니다...
```

# time.sleep() 예제

```
import time
print("날씨 정보를 불러오는 중입니다...")
time.sleep(3)
print("날씨 정보 불러오기 완료!")
print("오늘은 치킨 시키기 좋은 날입니다.")
```



```
>>> %Run test.py
날씨 정보를 불러오는 중입니다...
날씨 정보 불러오기 완료!
오늘은 치킨 시키기 좋은 날입니다.
```

# time.sleep() 예제

```
import time
for a in range(1, 11):
    print(a, "초...")
```

# time.sleep() 예제

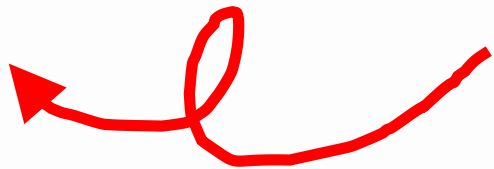
```
import time
for a in range(1, 11):
    print(a, "초...")
```



```
>>> %Run test.py
```

```
1 초...
2 초...
3 초...
4 초...
```

1부터 10까지  
한꺼번에 다 나온다.



# time.sleep() 예제

```
import time
for a in range(1, 11):
    print(a, "초...")
    time.sleep(1)
```



# time.sleep() 예제

```
import time
for a in range(1, 11):
    print(a, "초...")
    time.sleep(1)
```

>>> %Run test.py

1 초...  
2 초...

1부터 10까지  
천천히 센다.

# time.time()

## time.time()

1970년 1월 1일부터 지금까지 지난 초를 뱉는다.

```
>>> print(time.time())
```

```
1564156507.3043346
```

# time.time()

```
>>> print(time.time())  
1564156507.3043346
```

오전 12:55:19

2019년 7월 27일 토요일

1970년 1월 1일부터 선생님이 이 함수를 부른 순간까지  
15억 6415만 6507 초가 지났다.

# time.localtime()

## time.localtime()

현재 시간을 뱉는다.

```
>>> a = time.localtime()
```

```
>>> a.tm_min
```

```
52
```

## ★ time.localtime() 속에 들어있는 변수들

이름	설명	예
tm_year	년도가 들어있음	2019 (년)
tm_mon	달이 들어있음	7 (월)
tm_mday	일이 들어있음	27 (일)
tm_hour	시가 들어있음	13 (시)
tm_min	분이 들어있음	52 (분)
tm_sec	초가 들어있음	12 (초)
tm_wday	요일(0~6)이 들어있음	5 (토요일)
tm_yday	1년의 몇번째 날인지가(1~365) 들어있음	208 (일)

(모두 tm\_ 이 붙는다.)

# time.localtime() 예제

```
import time
t = time.localtime()
year = t.tm_year
print("지금은", year, "년입니다.")
```

# time.localtime() 예제

```
import time
t = time.localtime()
year = t.tm_year
print("지금은", year, "년입니다.")
```



```
>>> %Run test.py
지금은 2019 년입니다.
```

```
import time
t = time.localtime()
print("오늘은",
      t.tm_year, "년",
      t.tm_mon, "월",
      t.tm_mday, "일입니다.")
```



```
import time
t = time.localtime()
print("오늘은",
      t.tm_year, "년",
      t.tm_mon, "월",
      t.tm_mday, "일입니다.")
```



```
>>> %Run test.py
오늘은 2019 년 7 월 27 일입니다.
```

# time.localtime() 예제

```
import time
t = time.localtime()
print("오늘은",
      t.tm_wday, "번째 요일입니다.")
```

# time.localtime() 예제

```
import time
t = time.localtime()
print("오늘은",
      t.tm_wday, "번째 요일입니다.")
```



```
>>> %Run test.py
오늘은 5 번째 요일입니다.
```

# 요일 세는 법

0	1	2	3	4	5	6
월	화	수	목	금	토	일

```
>>> t = time.localtime()
```

```
>>> print(t.tm_wday)
```

5

토요일!

# 같이 해보기

1. 오늘이 무슨 요일인지 알려주는 프로그램을 만들자.

힌트: `time.localtime()`과 그 속에 있는 `tm_wday`를 사용하자.

<code>tm_wday</code>	요일(0~6)이 들어있음
----------------------	---------------

```
>>> %Run test.py
토 요일입니다.
```

```
import time
```

time 모듈을 불러옴

```
import time
```

time 모듈을 불러옴

```
t = time.localtime()
```

```
w = t.tm_wday
```

현재 시간을 구하고

w에 요일을 넣음 (0~6 숫자)

```
import time
t = time.localtime()
w = t.tm_wday
if w == 0:
    print("월요일입니다.")
```

w가 0이면 월요일



```
import time
```

```
t = time.localtime()
```

```
w = t.tm_wday
```

```
if w == 0:
```

w가 0이면 월요일

```
    print("월요일입니다.")
```

```
elif w == 1:
```

w가 1이면 화요일

```
    print("화요일입니다.")
```

```
import time
```

```
t = time.localtime()
```

```
w = t.tm_wday
```

```
if w == 0:
```

w가 0이면 월요일

```
    print("월요일입니다.")
```

```
elif w == 1:
```

w가 1이면 화요일

```
    print("화요일입니다.")
```

근데 if문을 7개 쓰기 너무 귀찮다...

```
import time  
t = time.localtime()  
w = t.tm_wday
```

```
import time
t = time.localtime()
w = t.tm_wday
요일들 = ["월", "화", "수",
           "목", "금", "토", "일"]
```

이런 리스트를 만든다면?

```
import time
t = time.localtime()
w = t.tm_wday
요일들 = ["월", "화", "수",
           "목", "금", "토", "일"]
```


이런 리스트를 만든다면?

요일들[w] 를 하면 뭐가 나올까?

```
import time
t = time.localtime()
w = t.tm_wday
요일들 = ["월", "화", "수",
          "목", "금", "토", "일"]
```

0            1            2            3            4            5            6

["월", "화", "수", "목", "금", "토", "일"]



```
import time
t = time.localtime()
w = t.tm_wday
요일들 = ["월", "화", "수",
          "목", "금", "토", "일"]
print(요일들[w], "요일입니다.")
```

```
import time
t = time.localtime()
w = t.tm_wday
요일들 = ["월", "화", "수",
           "목", "금", "토", "일"]
print(요일들[w], "요일입니다.")
```



```
>>> %Run test.py
토 요일입니다.
```



# 직접 해보기

1. 현재 시간을 출력하는 프로그램을 만들자.

힌트: `time.localtime()` 과 그 속의 변수들을 이용하자.

```
>>> %Run test.py
```

현재 시간은 13 시 53 분 12 초입니다.

참고

tm_hour	시가 들어있음
tm_min	분이 들어있음
tm_sec	초가 들어있음
tm_wday	요일(0~6)이 들어있음

# 직접 해보기

## 2. 현재 시간을 1초마다 출력하는 시계를 만들자.

힌트: 1번 코드와 `time.sleep()`을 사용하자.

또 `while True:` 로 무한반복문을 만들 수 있다.

```
>>> %Run test.py
```

```
현재 시간은 13 시 53 분 12 초입니다.
```

```
현재 시간은 13 시 53 분 13 초입니다.
```

```
현재 시간은 13 시 53 분 14 초입니다.
```

```
현재 시간은 13 시 53 분 15 초입니다.
```

```
현재 시간은 13 시 53 분 16 초입니다.
```

```
현재 시간은 13 시 53 분 17 초입니다.
```

# 직접 해보기

3. 오늘이 평일인지 주말인지 알려주는 프로그램을 만들자.

힌트: `time.localtime()`과 그 속의 변수, `if`와 `else`를 이용하자.

```
>>> %Run test.py
오늘은 주말입니다. 야호!
```

참고

tm_hour	시가 들어있음
tm_min	분이 들어있음
tm_sec	초가 들어있음
tm_wday	요일(0~6)이 들어있음

0	1	2	3	4	5	6
월	화	수	목	금	토	일
		(평일)			(주말)	

# range

## range(a, b)

“a 이상 b 미만인 모든 정수”라는 표현을 뱉는다.

```
>>> for a in range(1, 6):  
        print("양", a, "마리...")
```

# range - 2

## range(a, b, c)

“a부터 b까지 c 간격으로 뛰어 세는 정수”  
c는 쓰지 않으면 기본적으로 1이다.

```
>>> for a in range(2, 100, 2):  
        print(a, "는 짝수입니다.")
```

range(2, 10, 3)

시작  
(포함)

끝  
(미포함)

간격

-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(2, 10, 3)

시작  
(포함)

끝  
(미포함)

간격

시작  
(포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(2, 10, 3)

시작  
(포함)

끝  
(미포함)

간격

시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12



range(2, 10, 3)

시작 (포함)      끝 (미포함)      간격

시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(2, 10, 3)

시작 (포함)      끝 (미포함)      간격

시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12



+3

range(2, 10, 3)

시작  
(포함)

끝  
(미포함)

간격

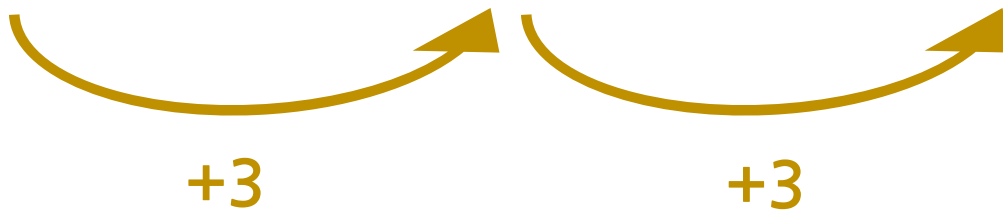
시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12



range(2, 10, 3)

시작 (포함)      끝 (미포함)      간격

시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12



+3



+3



+3

range(2, 10, 2)

시작  
(포함)

끝  
(미포함)

간격

시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(2, 10, 2)

시작 (포함)      끝 (미포함)      간격

시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(2, 10, 2)

시작 (포함)      끝 (미포함)      간격

시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12



+2

range(**2**, **10**, **2**)

시작 (포함)      끝 (미포함)      간격

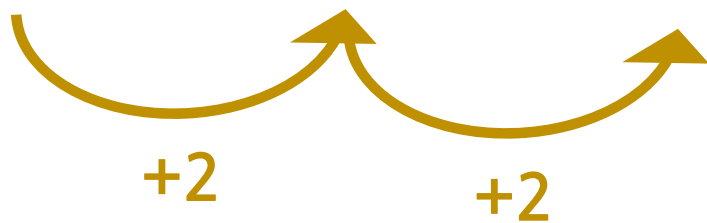
시작  
(포함)



끝  
(미포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12





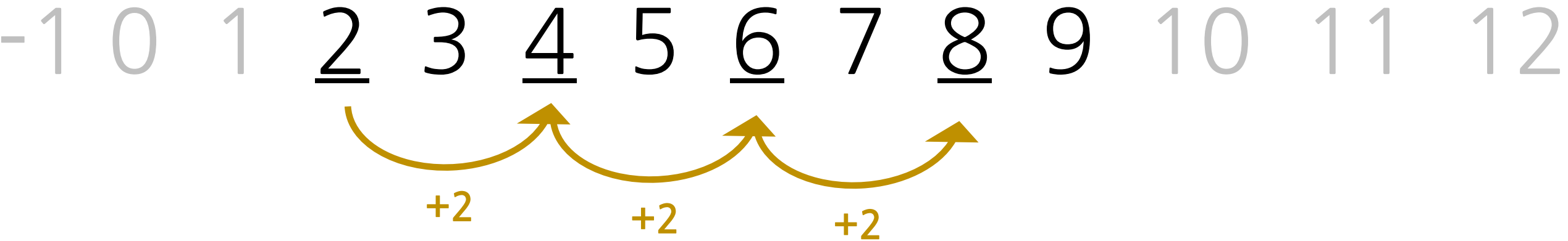
range(**2**, **10**, **2**)

시작 (포함)      끝 (미포함)      간격

시작  
(포함)



끝  
(미포함)



range(10, 1, -3)

시작  
(포함)

끝  
(미포함)

간격

-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(10, 1, -3)

시작 (포함)      끝 (미포함)      간격

끝  
(미포함)



시작  
(포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(10, 1, -3)

시작 (포함)      끝 (미포함)      간격

끝  
(미포함)



시작  
(포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12

range(10, 1, -3)

시작 (포함)      끝 (미포함)      간격

끝  
(미포함)



시작  
(포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12



-3

range(**10**, **1**, **-3**)

시작 (포함)      끝 (미포함)      간격

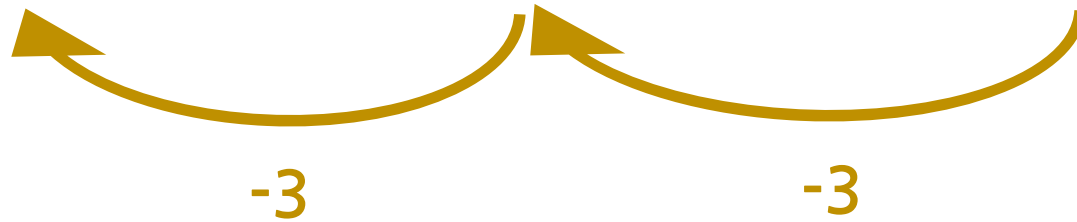
끝  
(미포함)



시작  
(포함)



-1 0 1 2 3 4 5 6 7 8 9 10 11 12



# range 예제

```
for a in range(0, 10, 2):  
    print(a)
```

# range 예제

```
for a in range(0, 10, 2):  
    print(a)
```

```
>>> %Run test.py
```

0

2

4

6

8



# range 예제

```
for a in range(5, 0, -1):  
    print(a)
```

# range 예제

```
for a in range(5, 0, -1):  
    print(a)
```

```
>>> %Run test.py
```

```
5  
4  
3  
2  
1
```

# list 변환

## list(A)

A를 리스트(목록)으로 바꿔 뱉는다.

```
>>> list(range(10))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> list("안녕하세요?")
```

```
['안', '녕', '하', '세', '요', '?']
```

# list 변환 예제

```
>>> print(range(0, 10))  
range(0, 10)
```

# list 변환 예제

```
>>> print(range(0, 10))
```

```
range(0, 10)
```

```
>>> print(list(range(0, 10)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# list 변환 예제

```
>>> print(range(0, 10))
```

```
range(0, 10)
```

```
>>> print(list(range(0, 10)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

어떤 range가 어떤 숫자를 뜻하는 지  
알아보고 싶을 때에 유용!

# 직접 해보기

1. 다음 range() 들이 어떤 숫자를 의미하는지 리스트로 변환해 출력해서 알아보자.

힌트: list(range(a, b, c)) 처럼 range 함수가 받은 값(수의 표현)을 리스트로 변환할 수 있다.

```
range(1, 7, 2)
```

```
range(1, 10, 999)
```

```
range(10, 1, -1)
```

```
range(10, 1, 1)
```

## 직접 해보기

2. 로켓 카운트다운을 만들어보자. **10**부터 **1**까지 **-1**의 간격으로 세다가 마지막에 “**발사!**”를 출력하자.

힌트: for문과 range(a, b, c)를 사용하자.

```
>>> %Run chicken.py
```

```
10 ...
```

```
9 ...
```

(생략)

```
2 ...
```

```
1 ...
```

```
발사!
```



# 직접 해보기

3. 2부터 100까지 짝수를 모두 출력해보자.

힌트: for문과 range(a, b, c)를 사용하자. 짝수만 출력하려면 간격이 몇이 되어야 할까? 또 시작과 끝은 몇이어야 할까?

```
>>> %Run test.py
```

```
2
```

```
4
```

```
....
```

(생략)

```
96
```

```
98
```

```
100
```

## 직접 해보기

4. 1부터 99까지 홀수를 모두 출력해보자.

힌트: 3번 코드에서 시작, 끝, 간격 중 하나만 바꾸면 된다.

```
>>> %Run test.py
```

```
1
```

```
3
```

```
....
```

(생략)

```
95
```

```
97
```

```
99
```

# print

```
print(A)
```

A를 화면에 출력한다.

```
print("안녕하세요!")
```

```
print(1 + 2)
```

# print - 2

```
print(A, B, C...)
```

A, B, C...를 띄어쓰기를 간격으로 모두 출력한다.

```
>>> print(1, 2, 3)
```

```
1 2 3
```

# print - 3

```
print(A, sep=B, end=C)
```

B 를 간격으로 뒤에 C 를 붙여 출력한다.

```
>>> print(1, 2, 3, sep='랑')
```

1랑2랑3

```
>>> print(1, 2, 3, end='입니다.')
```

1 2 3입니다.

# print - sep 예제

```
print("삼각", "김밥", "을 먹자")
```

→ `>>> %Run test.py`  
삼각 김밥 을 먹자

# print - sep 예제

```
print("삼각", "김밥", "을 먹자")
```

→ `>>> %Run test.py`  
삼각 김밥 을 먹자

```
print("삼각", "김밥", "을 먹자", sep="")
```

→ `>>> %Run test.py`  
삼각김밥을 먹자

# print - sep 예제

```
age = input("몇 살인가요? ")  
print("당신은 ", age, "살입니다.", sep="")
```



```
>>> %Run test.py  
몇 살인가요? 10  
당신은 10살입니다.
```



# print - end 예제

```
for a in range(0, 5):  
    print(a)
```

# print - end 예제

```
for a in range(0, 5):  
    print(a)
```



```
>>> %Run test.py
```

0

1

2

3

4

# print - end 예제

```
for a in range(0, 5):  
    print(a, end="\n")
```



end="Wn" 이 생략되어있었다.

```
>>> %Run test.py
```

```
0  
1  
2  
3  
4
```

# print - end 예제

```
for a in range(0, 5):  
    print(a, end=" ")
```

# print - end 예제

```
for a in range(0, 5):  
    print(a, end=" ")
```



```
>>> %Run test.py  
0 1 2 3 4
```

# print - end 예제

```
print(0, end="")  
print(1, end="")  
print(2, end="")  
print(3, end="")  
print(4, end="")
```



```
>>> %Run test.py  
01234
```

# 직접 해보기

1. 현재 시간을 출력하는 프로그램을 수정해 띄어쓰기가 자연스럽게 만들자.

힌트: print 함수를 부를 때 sep=??? 을 같이 줘보자.

```
>>> %Run test.py
```

현재 시간은 13 시 53 분 12 초입니다. (x)

```
>>> %Run test.py
```

현재 시간은 13시 53분 12초입니다. (o)

## 직접 해보기

2. `a = ["치", "킨", "좋", "아", "요"]` 인 리스트가 있다고 하자. 리스트 `a` 안에 있는 글자를 이어서 출력해보자.

힌트: for문과 print 함수를 이용하자. print 함수를 부를 때에 `end=???`를 같이 줘보자.

```
>>> %Run test.py
치킨좋아요
```



## 직접 해보기

3. 어떤 문장을 입력받고 그 문장을 한 글자마다 뒤에 “★”를 넣어 출력해보자.

힌트: 어떤 문장을 a에 입력받았다고 하면, for b in a 로 한 글자 한 글자를 모두 반복해 출력하자. 단 print 함수를 부를 때에 end=??? 를 같이 줘야 한다.

```
>>> %Run test.py
```

```
문장을 입력해주세요. 받아삼겹살미사일!  
받★아★라★삼★겹★살★미★사★일★!★
```



# 파이썬과 친해지기

time 모듈, range 함수 심화, 목록 변환,  
print 함수 심화

10주차

## References

왕초보를 위한 Python 2.7  
뱀 인형 이미지  
Stock Photos

<https://wikidocs.net/145>  
<https://bit.ly/2WINL65>  
<https://unsplash.com/>