

파이썬 소개

파이썬은 1989년 네덜란드의 프로그래머 구도 반 로섬(Guido van Rossum)이 만든 컴퓨터 프로그래밍 언어입니다.

문법이 매우 쉬운 관계로 처음 프로그래밍 언어를 배울 때 많이 추천되는 언어이기도 합니다. 또한 학교에서만 사용되는 것이 아니라 회사에서도 쓰는 곳이 많습니다. 특히 최근에는 인공지능용 언어로 매우 각광받고 있습니다.

파이썬은 문법 구조를 최대한 단순화하려는 경향을 가지고 있습니다. 그러다 보니 다른 프로그래밍 언어에서 흔히 볼 수 있는 중괄호가 없고, 대신 들여쓰기로 중괄호의 역할을 대체하고 있습니다. 명시적인 중괄호가 없다보니 들여쓰기로 인해 코딩에 다소 어려움이 따를 수도 있겠습니다.

들여쓰기로 탭키를 쓸 수가 있는데, 파이썬에서는 탭을 공백 1문자로 처리하므로 탭의 정의가 다른 도구에서 코드를 열어볼 경우 들여쓰기가 망가질 수도 있음을 명심하세요. 들여쓰기를 위해 스페이스바를 4번 칠 수도 있겠는데, 좀 귀찮겠죠? 일단은 임시방편으로 탭으로 들여쓰기를 하도록 합니다. 높은 수준의 에디터에서는 탭을 자동으로 스페이스바 4번으로 자동변환해주기도 한답니다.

순수 객체지향

파이썬에는 원시 타입(Primitive Type)이 존재하지 않으며, 모든 것이 객체로 취급됩니다. 클래스나 함수는 물론 상수도 상수가 저장된 객체로 인식합니다. 다음과 같은 문장을 생각해 보죠:

`a = 109`

위 문장은 변수 `a`에 109가 할당된 것이 아니라 `a`가 109가 저장된 상수 객체를 가리킨다는 의미입니다. `a`에 대입된 값을 변경할 경우

`a = 109`

`a = 302`

`a`가 가리키는 대상이 109가 저장된 상수 객체에서 302가 저장된 상수 객체로 바뀐 것을 의미합니다. `a`의 값이 109에서 302로 바뀐게 아닙니다.

장점

인터프리터 언어지만 개발기간이 단축됩니다. 다른 언어와의 연계도 우수하므로 파이썬으로 빠르게 구현하고, 속도에 병목이 오는 부분을 더빠른 언어로 보완하는 경우도 있습니다.

문법이 통일되다 보니 프로그램 하나를 만들면 참고하는 사람이 대단히 많아서 최근 파이썬의 점유율이 매우 높아졌습니다. 물론 인공지능 광풍 때문이기도 하겠지만요.

파이썬은 과학, 공학 분야에서 필요한 여러 기능을 기본적으로 제공합니다. 언어 자체가 64비트를 넘어가는 매우 큰 정수를 지원합니다. 허수도 지원합니다. 그리고 1보다 작은 실수를 정밀하게 다루는 관계로 암호학과 통계 분야에도 적절한 언어입니다.

단점

멀티스레딩에 문제가 있습니다.

패키지 개발 및 배포가 복잡합니다.

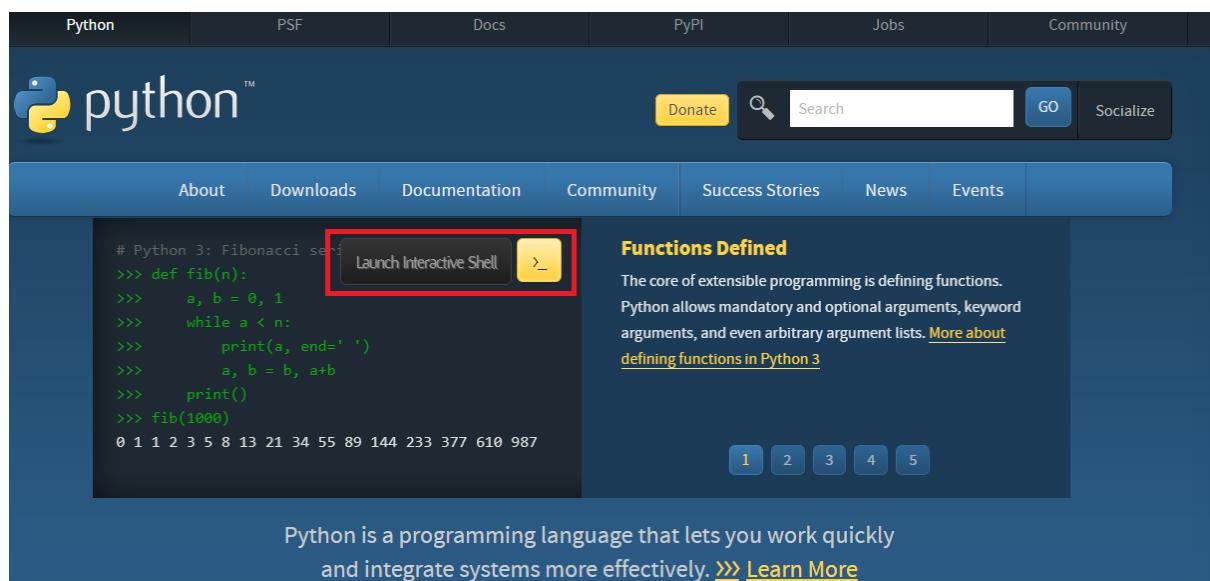
파이썬 시작하기

파이썬 개발환경은 여러가지 형태가 존재합니다.

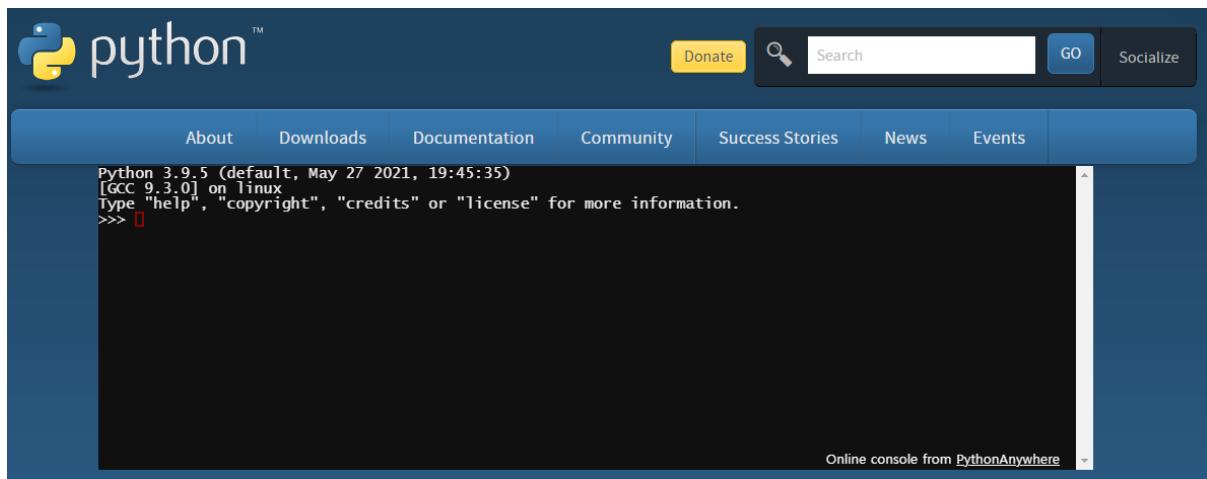
이 수업에서 파이썬을 연습하기 위해서는 아나콘다를 설치하고 주피터노트북으로 코딩하는 것입니다. 그런데, 처음부터 아나콘다 패키지를 설치하는 것은 시간이 조금 소요됩니다. 또한 우리 인공지능학부 실습실 환경은 일부 PC를 제외하고는 아나콘다가 설치되어 있습니다. 또한 인공지능서버의 주피터허브를 사용하여 실습할 수도 있게 준비가 되어 있습니다. 그러나 지난 여름방학 때 고교 특강에 따라 설치환경을 재정비해야 하는 상황이기도 합니다. 아나콘다를 삭제하고, 재설치하거나 신규설치를 해야 하는데, 수업 4번하는데 설치 과정이 들어가게 되면 다른 것을 할 시간이 부족할 것 같습니다.

당장 파이썬 코딩 연습을 하기 위해서 간편하게 사용하는 방법은 파이썬 홈페이지에서 파이썬을 실행하는 것입니다.

웹브라우저에서 “[python.org](https://www.python.org)”를 입력해보세요.



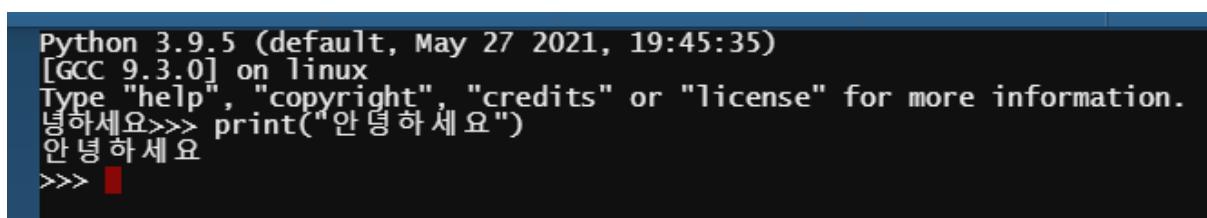
그러면 윗그림과 같은 모양의 파이썬 홈페이지가 뜹니다. 이 때 화면 중앙 부근의 노란 정사각형에 커서를 올리면 “Launch Interactive Shell”이라는 풍선 도움말이 나올 것입니다. 이 노란 정사각형을 클릭해보세요.



그러면 이전 그림처럼 파이썬 인터프리터 화면이 뜰 것입니다. 빨간 커서 위치에

```
print("안녕하세요")
```

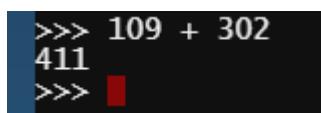
를 입력하고 엔터키를 눌러보세요.



위 그림처럼 “안녕하세요”를 보여줄 것입니다.

자, 그럼 코딩을 시작해 볼까요?

덧셈



위 그림처럼 숫자 + 숫자 형식으로 수를 입력하고 엔터키를 눌러보세요. 바로 답이 출력되는 것을 볼 수 있습니다. 참 쉽죠?

물론 이런 표현은 간단한 테스트를 위해서 사용하는 겁니다. 정식으로 사용하면 좀 곤란하겠죠?

빨셈, 곱셈, 나눗셈

```
>>> 109 - 302
-193
>>> 109 * 302
32918
>>> 109 / 302
0.3609271523178808
>>> █
```

이어서 뺄셈, 곱셈, 나눗셈도 위 그림처럼 두 수를 중심으로 -, *, / 연산기호를 사용하여 실행시켜 보기 바랍니다. 나눗셈의 경우 무한소수의 자릿수가 너무 길게 표현되는 것이 눈에 그슬리죠? 이것은 출력 자릿수를 제한하면 됩니다만, 일단은 그냥 넘어가도록 하겠습니다.

몫과 나머지

```
>>> 109 // 302
0
>>> 109 % 302
109
>>> █
```

몫을 구하려면 //를 사용하고 나머지는 % 기호를 씁니다.

변수

정수를 가리키는 변수

어떤 문구점을 가서 필요한 문구 몇점을 사려고 합니다.

우선 연필이 필요할 것 같군요. 300원이라고 적혀 있네요. 또 스케치북이 떨어졌습니다. 2000원짜리가 눈에 들어옵니다.

자, 연필과 스케치북을 구입하려면 얼마가 필요할까요? 이전에 했던 것처럼 300 + 2000이라고 하면 금방 답을 얻을 수 있습니다.

그런데, 나만 필요한 것이 아니라 다른 사람도 연필과 스케치북이 필요할 수도 있잖아요? 그래서 연필값을 따로 가리킬 수 있도록 변수를 활용해서 계산해보면 어떨까요?

```
>>> pencil = 300
>>> sketchbook = 2000
>>> pencil + sketchbook
2300
>>> █
```

위 그림처럼 변수와 변수끼리 덧셈이라는 연산을 할 수 있습니다.

또한, 계산 결과를 저장하려면, 다른 변수를 만들면 가능합니다.

```
>>> price = pencil + sketchbook
>>> price
2300
>>> █
```

총구입금액을 `price`가 보관중입니다. 여기서 크레용이 추가로 필요해서 크레용을 같이 구입하였다면 금액이 늘어나겠죠?

```
>>> creyon = 4000
>>> price = price + creyon
>>> price
6300
>>> □
```

문자열을 가리키는 변수

이번에는 글자를 넣는 변수를 생각해보겠습니다.

```
>>> a = 'bird'
>>>
>>> b = 'boy'
>>>
>>> a + b
'birdboy'
>>>
>>> a + ' ' + b
'bird boy'
>>> □
```

`a`가 `bird`를 가리키고, `b`가 `boy`를 가리킵니다. 이 때 문자열에는 반드시 따옴표 또는 이중 따옴표를 반드시 붙여서 문자열 상수임을 나타내도록 해야 합니다. 그냥 `bird`라고 하면 `a`가 `bird`라는 변수를 가리키는 것처럼 되어 버립니다. 이제 `a`와 `b`를 더하면 무슨 일이 일어날까요? 문자열과 문자열의 덧셈은 두 문자열의 접속(concatenation)으로 표현됩니다.

구글 코랩

전체 이미지 동영상 뉴스 도서 더보기 도구

검색결과 약 19,800개 (0.27초)

<https://colab.sandbox.google.com/notebooks>

Colaboratory에 오신 것을 환영합니다 - Colaboratory - Google

Google Drive 계정에서 스프레드시트를 비롯한 데이터를 Colab 메모장으로 가져오거나 GitHub 등의 여러 다른 소스에서 데이터를 가져올 수 있습니다. Colab을 데이터 과학 ...

<https://research.google.com/colaboratory>

Welcome To Colaboratory - Google Research

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To ...

구글 코랩

파이썬 홈페이지에서 간단한 테스트는 해볼 수 있겠습니다만, 아무래도 불편해보입니다. 한정된 수업시간으로 생각해볼 수 있는 파이썬을 사용한 데이터 분석방법으로는 구글 코랩을 활용하는 것입니다. 구글 검색창에 “구글 코랩”을 검색해 보세요. 그리고 위 그림과 같이 제일 처음 나오는 주소를 클릭하세요.

Colaboratory에 오신 것을 환영합니다

파일 수정 보기 삽입 런타임 도구 도움말

공유 설정

목차

+ 코드 + 텍스트 Drive로 복사

연결 수정 가능

Colaboratory란?

줄여서 'Colab'이라고도 하는 Colaboratory를 사용하면 브라우저에서 Python을 작성하고 실행할 수 있습니다. Colab은 다음과 같은 이점을 자랑합니다.

- 구성이 필요하지 않음
- GPU 무료 액세스
- 간편한 공유

학생이든, 데이터 과학자든, AI 연구원이든 Colab으로 업무를 더욱 간편하게 처리할 수 있습니다. [Colab](#)에서 자세한 내용을 확인하거나 아래에서 시작해 보세요.

시작하기

지금 읽고 계신 문서는 정적 웹페이지가 아니라 코드를 작성하고 실행할 수 있는 대화형 환경인 **Colab**입니다.

예를 들어 다음은 값을 계산하여 변수로 저장하고 결과를 출력하는 간단한 Python 스크립트가 포함된 셀입니다.

그러면 위 그림과 같이 구글 코랩 시작 화면이 뜹니다. 여기서 메뉴 중 파일-새 노트를 클릭합니다.

Untitled4.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

댓글 공유 설정

+ 코드 + 텍스트

연결 수정 가능

그러면 위 그림과 같이 새로운 탭으로 노트가 뜰 것입니다. 위 그림에서 빨간 직사각형을 셀이라고 하며 코드를 입력할 수 있습니다.

파이썬 제어구조

조건문

어느 프로그래밍 언어와 마찬가지로 파이썬을 사용해서 코딩을 하려면 기본적인 문법을 알고 있어야 합니다. 코드를 실행하면 첫 문장부터 순차적으로 실행이 됩니다. 그런데, 경우에 따라 조건에 맞으면 실행을 하고, 그렇지 않으면 실행하지 않는 코드를 사용할 수도 있습니다. 이 때 조건문을 씁니다.

if와 else

파이썬의 조건문은 `if`와 `else`로 구성됩니다. 예를 한 번 들어볼까요?
두 수 중에 어느 값이 큰지 비교를 해보도록 하죠.

```
✓ 0초 ▶
a = 365
b = 31
if a > b:
    print('a가 b보다 크다')
else:
    print('b가 a보다 크다')

a가 b보다 크다
```

위 그림은 365와 31 중에서 어느 수가 더 큰지를 비교하는 코드입니다. 여기서 주의할 점은 `if`문과 `else`문의 끝에 콜론이 들어간다는 점입니다. 다른 프로그래밍 언어에서는 볼 수 없는 독특한 형태를 가지고 있는데요, 문법에 어긋나지 않도록 꼭 집어넣어서 사용하기

바랍니다. 셀을 실행하기 위해서는  버튼을 누르거나(`ctrl + enter`), `shift + enter`를 치면 됩니다. `ctrl+enter`는 셀을 실행하고, `shift+enter`는 셀을 실행하면서 다음 셀을 나타냅니다.

그런데, 단순히 ‘`a가 b보다 크다`’라고 표시하는 것보다 ‘`365가 31보다 크다`’로 나타내는 것이 더 나은 표현이겠죠? 이렇게 하려면 어떻게 어떻게 코딩하면 될까요? `print`문에서 문자열과 문자열의 덧셈을 사용하면 됩니다.

```
[3] 0초 a = 365
      b = 31
      if a > b:
          print(a + '가 ' + b + '보다 크다')
      else:
          print(b + '가 ' + a + '보다 크다')
```

```
TypeError Traceback (most recent call last)
<ipython-input-3-45ba9ab1e7a8> in <module>()
      2 b = 31
      3 if a > b:
----> 4     print(a + '가 ' + b + '보다 크다')
      5 else:
      6     print(b + '가 ' + a + '보다 크다')

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

[SEARCH STACK OVERFLOW](#)

위 그림은 '365가 31보다 크다'를 출력하기 위해 문자열과 문자열의 덧셈을 사용한 프린트 함수의 실행 결과를 나타냅니다. 그런데, 에러가 나타났습니다. 사유는 변수 **a**의 타입은 정수인데, 정수와 문자열의 덧셈은 허용이 되지 않는다는 겁니다.

당연한 얘기죠? 이 문제를 해결하려면 정수를 문자열로 변환해서 출력하는 방법을 생각할 수 있습니다.

```
[4] a = 365
      b = 31
      if a > b:
          print(a, '가 ', b, '보다 크다')
      else:
          print(b, '가 ', a, '보다 크다')
```

365 가 31 보다 크다

정수를 문자열로 변환하는 방법을 알아야 하겠죠? 그 전에 어떤 프로그래밍 언어든 간에 함수에서 매개변수를 여러개 넣으려면 콤마를 사용하게 되는데요, 프로그래밍 언어를 조금이라도 접해본 학생들은 금방 눈에 들어올거예요.

위 그림처럼 프린트 함수에서 출력하려는 항목이 여러개일 때 콤마로 나열을 하면 되는 겁니다.

```
[5] a = 365
    b = 31
    if a > b:
        print(str(a) + '가 ' + str(b) + '보다 크다')
    else:
        print(str(b) + '가 ' + str(a) + '보다 크다')
```

365가 31보다 크다

또한 조금전에 언급한 바와 같이 정수를 문자열로 변환하기 위해서 사용하는 함수가 있습니다. 바로 **str** 함수입니다. 위 그림에서는 정수형 변수를 문자열로 변환하여 문자열의 덧셈을 이용해서 출력하는 모습을 나타내고 있습니다.

elif

조건이 ‘예’와 ‘아니오’로 끝나는 경우도 있지만 여러개의 조건을 비교할 때도 많습니다. 여러가지 조건을 비교할 때는 **if**와 **else** 사이에 **elif**를 집어넣습니다.

```
[7] if a > b:
    print(str(a) + '가 ' + str(b) + '보다 크다')
elif a == b:
    print(str(a) + '와 ' + str(b) + '의 크기가 같다')
elif a < b:
    print(str(a) + '가 ' + str(b) + '보다 작다')
else:
    print(str(b) + '와 ' + str(a) + '의 크기를 비교할 수 없다')
```

365가 31보다 크다

while 문

while 문은 조건에 맞으면 **while**내의 내용을 언제까지라도 반복하는 반복문입니다. 예를 들어 1부터 10까지 더하여 결과를 출력하는 코드를 **while**문을 사용하여 표현해볼까요?

```
[8] c = 1
    result = 0
    while c <= 10:
        result = result + c
        c = c + 1
    print('1부터 10까지의 합 = ', result)
```

1부터 10까지의 합 = 55

위 그림은 1부터 10까지의 합을 `while`문으로 표현한 코드입니다. `while` 문도 `if` 문처럼 끝에 콜론을 넣는다는 것을 알 수가 있습니다. 지금은 `while` 문에 1부터 10까지 지정된 수로 계산하고 있죠? 그런데 m부터 n까지의 합을 구하고, m과 n을 입력으로 받아서 계산한다면 코드를 어떻게 수정하면 될까요?

위 코드에서 c에 m을 대입하고, `while` 문의 조건을 `c <= n`으로 변경해야 할 것입니다:

```
[9] m = int(input("두 수 중 작은 수를 입력하시오: "))
n = int(input("두 수 중 큰 수를 입력하시오: "))
c = m
result = 0
while c <= n:
    result = result + c
    c = c + 1
print(m, '부터 ', n, '까지의 합 = ', result)
```

두 수 중 작은 수를 입력하시오: 1
 두 수 중 큰 수를 입력하시오: 100
 1부터 100 까지의 합 = 5050

`m`과 `n` 입력을 받으면 숫자를 입력하더라도 문자열로 인식되므로 `int` 함수를 사용하여 정수로 변환한 후 `m`과 `n`에 집어넣도록 합니다.

또한 두 수를 입력할 때 작은 수부터 입력하라고 명시했음에도 불구하고 큰 수를 먼저 집어넣을 수도 있을 것입니다. 이런 사고를 미연에 방지하려면 두 수를 입력받은 후 대소비교를 하고나서 총합을 구하는 것이 안전하겠습니다?

다음 그림은 큰 수를 먼저 입력했을 때를 가정하여 입력된 두 수를 서로 비교한 후 큰 수가 먼저 입력되었다면 두 수를 교환하고 나서 진행할 수 있도록 수정된 프로그램을 나타냅니다.

```
[9] m = int(input("두 수 중 작은 수를 입력하시오: "))
n = int(input("두 수 중 큰 수를 입력하시오: "))

if m < n:
    print('정상적으로 입력되었습니다')
else:
    print('큰 수가 먼저 입력되었습니다')
    temp = n
    n = m
    m = temp
```

for 문

파이썬에서의 `for` 문은 다른 언어의 `for` 문과는 차이가 있습니다. 그래서 당황할 수도 있겠는데, 요즈음에는 다른 언어에서도 범위기반 `for` 문을 즐겨쓰기도 합니다.

```
[10] total = 0
for x in range(1, 11):
    total = total + x
print(total)
```

55

위 코드는 1부터 10까지의 합을 구하기 위해 `for` 문을 사용하여 작성한 프로그램입니다. 여기서 `range(10)`은 최대 10까지의 시퀀스를 나타냅니다. `range` 함수는 `range(min, max, step)`으로 표기할 수 있습니다. `min`부터 `max`까지 `step` 만큼 건너뛰면서 오름차순으로 숫자를 나열합니다. 시작값과 스텝수는 생략이 가능하며 생략할 경우 시작값과 스텝수는 1로 자동 설정됩니다.

따라서 `range(1, 11)`이라고 하면 1부터 시작하여 2, 3, 4..., 10까지를 나타냅니다. 아... 그리고 보니 10이 빠졌죠? `range` 함수에서는 최대값 - 1까지의 수가 나열됩니다. 자칫하면 혼동할 수 있으니 주의 바랍니다.

다음 그림은 '*'를 사용하여 삼각형을 출력해본 것입니다. `while` 문을 사용하였습니다.

```
[11] wstar = 1
    while wstar <= 5:
        print('*' * wstar)
        wstar = wstar + 1
```

```
*
**
***
****
*****
```

여기서 잠깐! 문자열 연산 중에 덧셈은 몇 번 경험해서 알게 되었는데, 문자열 곱셈은 좀 생소하죠? 문자열 곱셈도 대상 문자열에 반복횟수를 곱하여 출력할 수 있습니다. 대상 문자열을 일일이 타이핑하지 않고도 원하는 반복값만큼 되풀이하는 것입니다.

```
[12] for x in range(1, 6):
    print('*' * x)
```

```
*
**
***
****
*****
```

위 그림은 같은 직삼각형 출력 결과를 `for` 문을 사용하여 실행한 코드를 나타냅니다.

오늘의 과제

이제부터 문제를 하나 풀어보고 넘어가도록 하겠습니다. 방금 별모양으로 삼각형 모양을 구성해보았습니다. 같은 방식으로 별 또는 다른 기호 혹은 반복횟수의 숫자(1, 2, 3, ...) 등을 활용하여 여러가지 형태의 삼각형을 출력해보세요. 역삼각형이나 정삼각형도 괜찮고, 삼각형을 270도 회전하여 두 개의 삼각형으로 출력해도 좋습니다. 단, 프로그램은 두 가지로 작성해야 합니다. 하나는 `while` 문으로 하고, 또하나는 `for` 문으로 작성해보세요. 결과는 두 코드가 동일한 출력을 내도록 해야 합니다.