

# Python 입문

## Python언어 개요

한승현

방문형SW교육 / SW중심대학지원사업

2017.04.07

# 목차

---

1 언어의 종류  
:왜 Python인가

---

2 Python 설치하기

---

3 Python은 어떤 언어인가

---

4 실습: 터틀 그래픽을 이용한  
Python 프로그래밍의 전반적 이해  
#2주차까지 지속  
#3주차부터는 Jupyter Notebook으로

# 언어의 종류

## :what is Programming

## Language

## #Why\_Python

## 1. 정의

1) 기계에게 명령 또는 연산을 시킬 목적으로 설계되어 기계와 의사소통을 할 수 있게 해주는 언어

- ① 쉽게 말하면 컴퓨터를 부러먹기 위한 언어
- ② 시초는 에이다 러브레이스(Ada Lovelace)로부터 시작



조지 고든 바이런  
엄친아  
시인/ 낭만주의 대가 중 한명



어거스타 에이다 킹  
엄친딸  
최초의 프로그래머

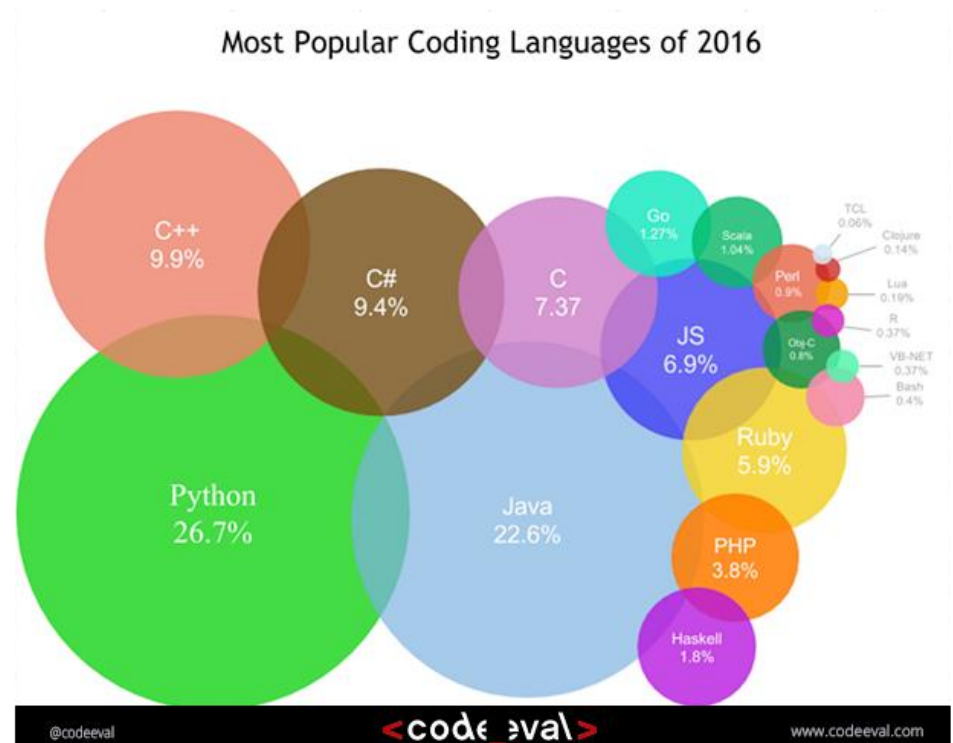
## 2. 언어의 종류

### 1) 시스템 프로그래밍용 언어

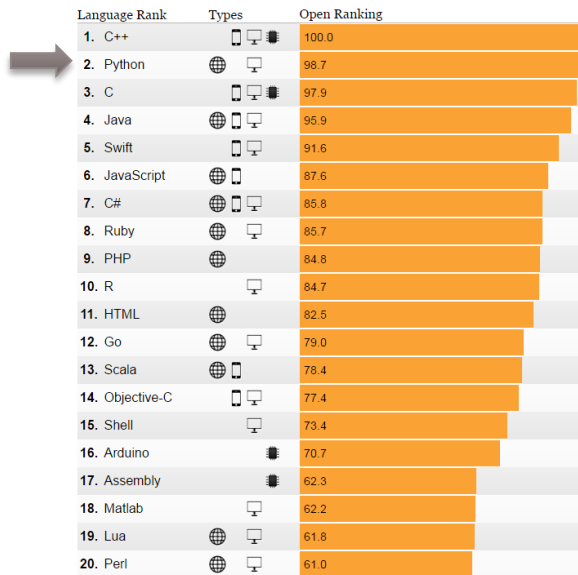
1) ① C, C++, C#, Java, Obj-C

### 2) 업무 자동화를 위한 스크립팅 언어

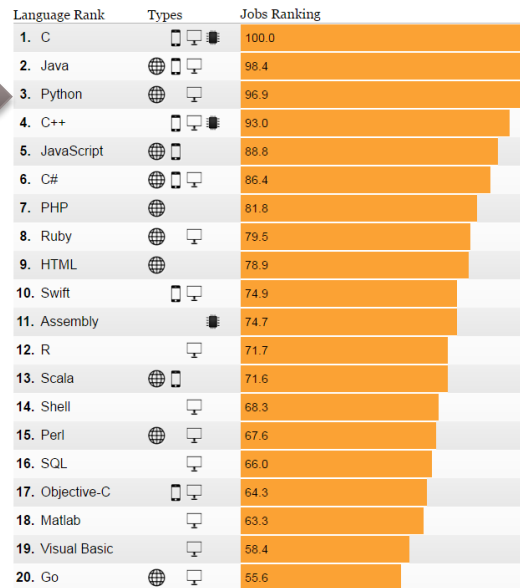
1) Toy Languages



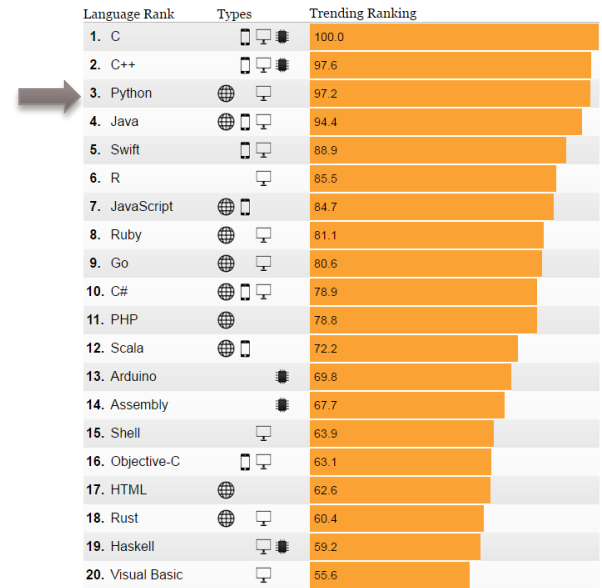
## 3. 왜 파이썬인가



대세 순위



언어 수요 순위



오픈소스 기여도 순위

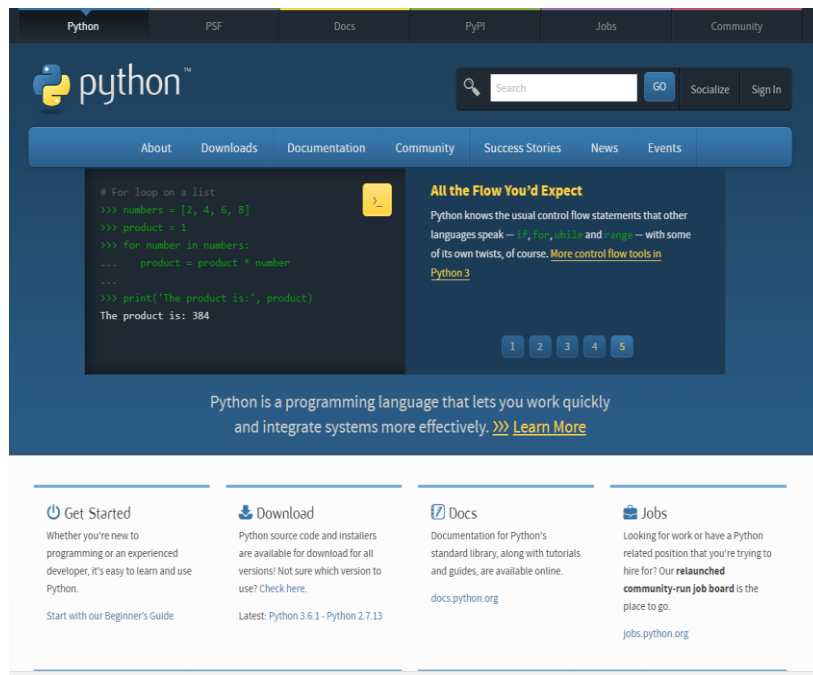
# Python 설치하기

02

# Install Python

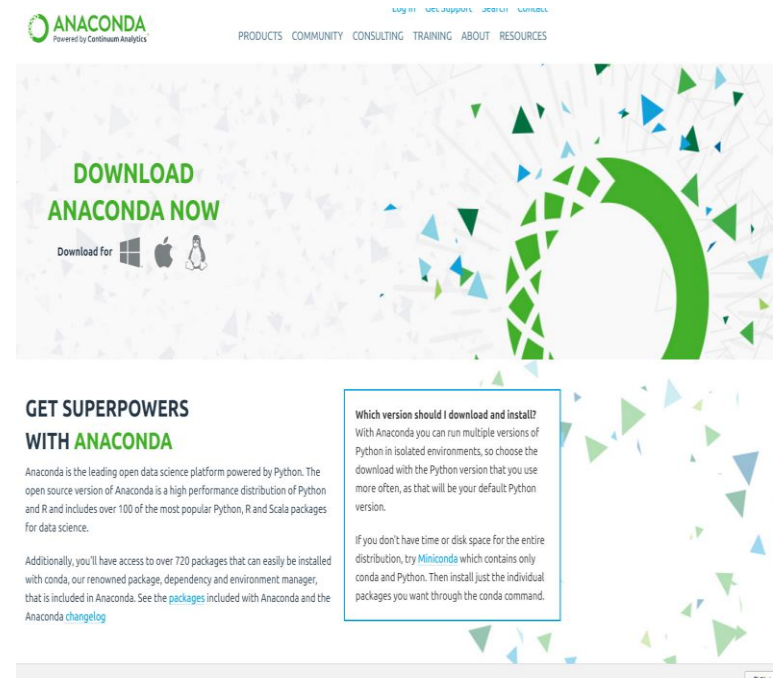
2-1

<https://www.python.org/>



The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links to Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a search bar and a 'GO' button. The main content area features a large blue banner with the Python logo and the text 'python™'. To the left of the banner is a code snippet demonstrating a for loop. To the right is a section titled 'All the Flow You'd Expect' with a brief description of Python's control flow statements. Below the banner, there's a section titled 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'. At the bottom, there are four columns: 'Get Started' (with a link to the Beginner's Guide), 'Download' (with a link to the latest version, Python 3.6.1 - Python 2.7.13), 'Docs' (with a link to the documentation), and 'Jobs' (with a link to the community-run job board).

<https://www.continuum.io/downloads>



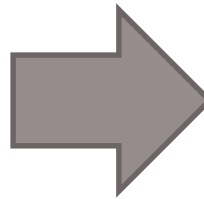
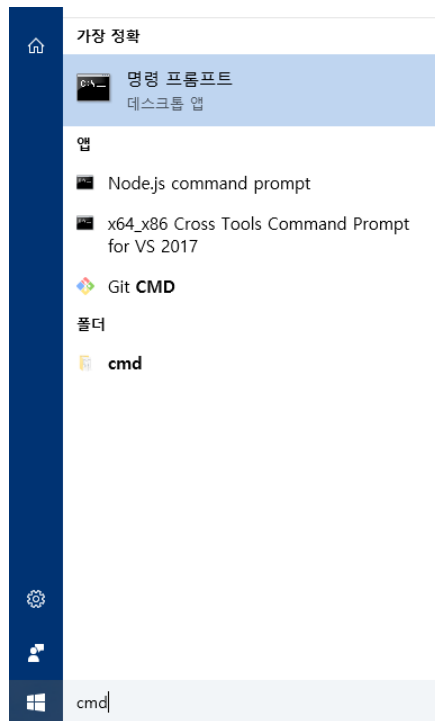
The screenshot shows the Anaconda website. At the top, there's a navigation bar with links to PRODUCTS, COMMUNITY, CONSULTING, TRAINING, ABOUT, and RESOURCES. Below this is a large green banner with the text 'DOWNLOAD ANACONDA NOW'. To the left of the banner is a section titled 'GET SUPERPOWERS WITH ANACONDA' with a brief description of Anaconda as a leading open data science platform. To the right is a section titled 'Which version should I download and install?' with a brief description of how to choose the right version. Below the banner, there's a section titled 'Additionally, you'll have access to over 720 packages that can easily be installed with conda, our renowned package, dependency and environment manager, that is included in Anaconda. See the [packages](#) included with Anaconda and the [Anaconda changelog](#)'.



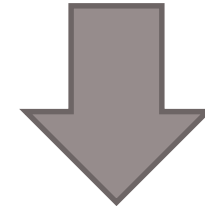
# HELLO WORLD!

2-2

## 1. console



```
C:\Users\Wuser>python
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:
18:10) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" fo
r more information.
>>> _
```



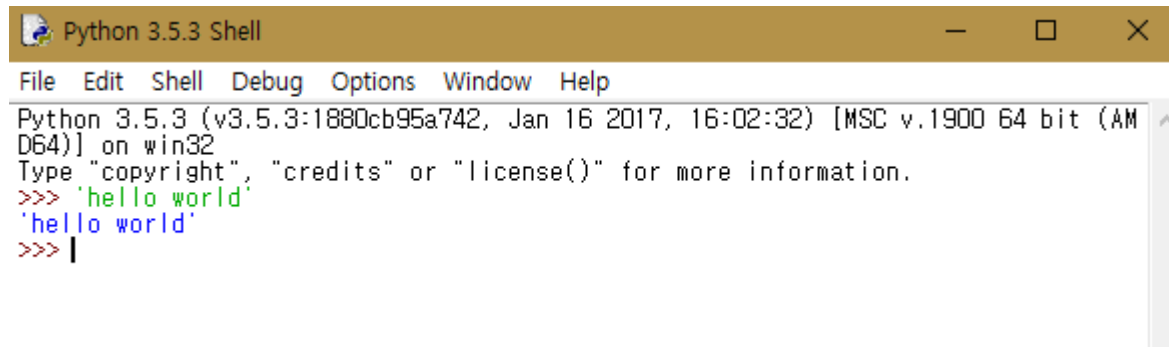
```
C:\Users\Wuser>python
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:
18:10) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" fo
r more information.
>>> print('Hello world!')_
```

# Dev Environment

2-3

## 2. Idle & Spyder

Idle = basic python interpreter

A screenshot of a Windows command prompt window titled "Python 3.5.3 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The text inside the window shows the Python version and build information: "Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32". It then prompts the user to type "copyright", "credits" or "license()" for more information. The user has entered ">>> 'hello world'" and the output is "'hello world'". The prompt ">>>" is followed by a cursor.

```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 'hello world'
'hello world'
>>> |
```



## 3. 3<sup>rd</sup> party IDE(integrated development environment) 통합 개발 환경

- Visual Studio
- Eclipse
- PyCharm



## 4. Text Editor

- Sublime Text
- Atom
- Vim
- ...



We are going to use

## 1. 실습

1. lpython : 1, 2차시에만
2. Atom : 텍스트로 볼 때
3. Jupyter Notebook 3차시 이후로

## 2. 개발/실습/과제

1. Pycharm  
(With Anaconda Interpreter?)

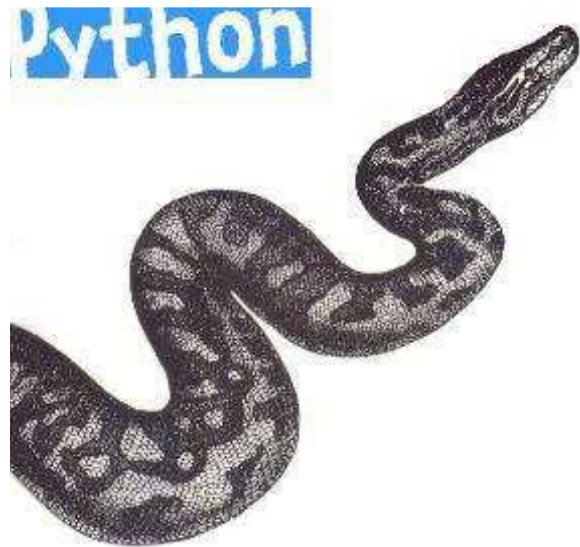
# Why Python

03

# What is Python

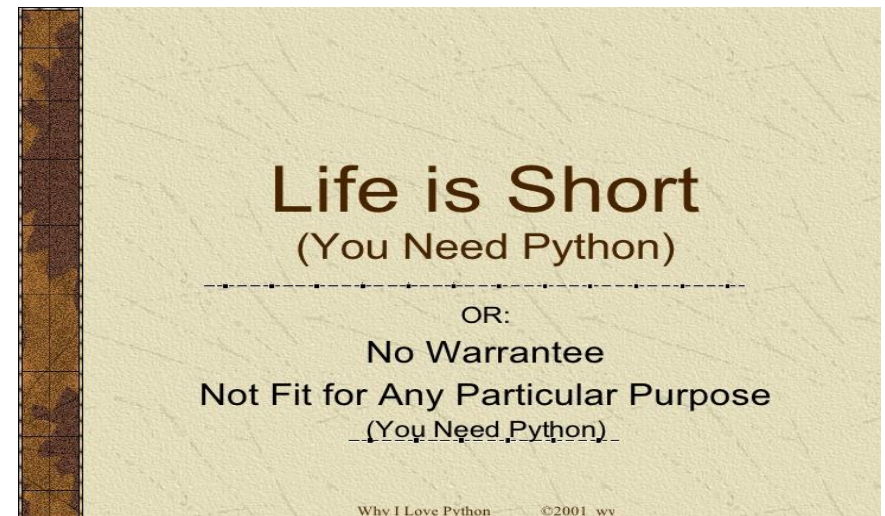
3-1

Python



- 1) "A large tropical snake that kills animals for food by winding itself around them and crushing them."

- from Longman Dictionary



# What is Python

3-1

Who made it?



Guido Van Rossum

- 1) 창시자는 귀도 판 로섬(Guido van Rossum).  
1989년 크리스마스 주에, 연구실이 닫혀있어서 심심한 김에 제작
- 2) 보통 말하는 Python은 C언어로 구현,  
C#으로 구현된 닷넷프레임워크 위에서 동작하는 IronPython  
CPython에서 C 스택을 없앤 Stackless Python  
Java로 구현되어 JVM위에서 돌아가는 Jython  
파이썬 자체로 구현된 PyPy 등이 있으며  
이 가운데 오리지널은 CPython

# What is Python

3-2

## Dev History



- 1989년 12월 개발 시작
- 1998년 4월 13일 - 파이썬 1.5.1 버전 발표
- 2000년 9월 6일 - 파이썬 1.6 최종 버전 발표
- 2000년 10월 16일 - 파이썬 2.0 최종 버전
- 2004년 11월 30일 - 파이썬 2.4 정식 버전
- 2007년 7월 18일 - 파이썬 2.5.1
- 2008년 9월 - 파이썬 2.6
- 2010년 7월 - 파이썬 2.7
- 2009년 1월 - 파이썬 3.0
- 2010년 - 파이썬 3.1.2
- 2011년 - 파이썬 3.2
- 2013년 - 파이썬 3.3
- 2014년 - 파이썬 3.4, ~2.7.9
- 2015년 - 파이썬 3.5, ~2.7.11
- 2016년 - 파이썬 3.6, ~2.7.13
- 2017년 - ...



# What is Python

3-2

## Dev History



# What is Python

3-3

Why Python?

```
In [4]: import antigravity
```

Python Philosophy

```
In [6]: import this_
```

아름다운 것이 추한 것보다 낫다.  
(Beautiful is better than ugly)  
명시적인 것이 암시적인 것보다 낫다.  
(Explicit is better than implicit)  
간결한 것이 복잡한 것보다 낫다.  
(Simple is better than complex)  
정교한 것이 난잡한 것보다 낫다.  
(Complex is better than complicated)

# 실습: 터틀 그래픽

03

## import

- 모듈 가져오기

```
import math  
from math import pi, sin  
from math import *  
import numpy as np
```



```
math.sin(math.pi/2)  
sin(pi / 2)  
log(e)  
np.sin(np.pi/2)
```

## 터틀 그래픽 실행

- 윈도우키 - IDLE [Enter]
- 윈도우키 + R, IDLE [Enter]

```
from turtle import *  
reset()
```

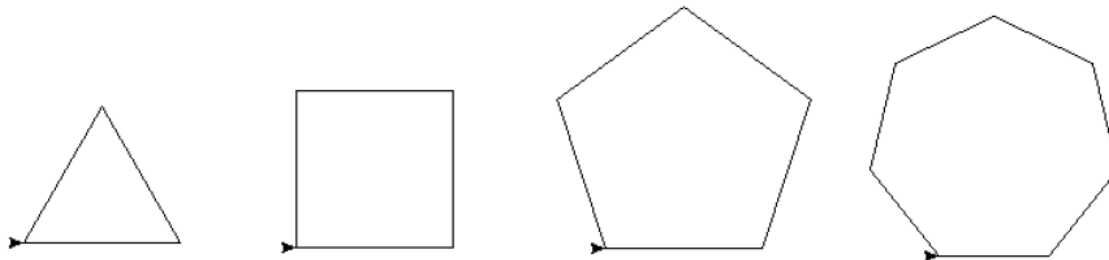
## 기본 함수

```
forward(x)
backward(x)
left(a)
right(a)
circle(radius, extent=None)
goto(x, y)
up(), down()
color(), width()
...

position()
heading()
```

## Turtle graphics

```
>>> from turtle import *  
>>> reset()  
  
>>> forward(100)  
>>> left(90)  
>>> circle(90)  
>>> circle(100, 90)  
>>> width(5)  
>>> circle(100, 90)
```



# TURTLE

4-1

[https://opentechschoool.github.io/python-beginners/ko/simple\\_drawing.html](https://opentechschoool.github.io/python-beginners/ko/simple_drawing.html)

## Code

```
# Step 1: Make all the "turtle" commands available to us.  
import turtle  
  
# Step 2: Create a new turtle. We'll call it "bob"  
bob = turtle.Turtle()  
  
# Step 3: Move in the direction Bob's facing for 50 pixels  
bob.forward(50)  
  
# Step 4: We're done!  
turtle.done()
```

## Expected Output





# TURTLE

4-1

## Code

```
import turtle

silly = turtle.Turtle()

silly.forward(50)
silly.right(90)    # Rotate clockwise by 90 degrees

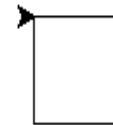
silly.forward(50)
silly.right(90)

silly.forward(50)
silly.right(90)

silly.forward(50)
silly.right(90)

turtle.done()
```

## Expected Output



## 너무 긴데? 줄여볼까? - 1 조건문

### 03-1 if문

다음과 같은 상상을 해보자.

"돈이 있으면 택시를 타고, 돈이 없으면 걸어 간다."

우리 모두에게 언제든지 일어날 수 있는 상황 중 하나이다. 프로그래밍도 사람이 하는 것이므로 위의 문장처럼 주어진 조건을 판단한 후 그 상황에 맞게 처리해야 할 경우가 생긴다. 이렇듯 프로그래밍에서 조건을 판단하여 해당 조건에 맞는 상황을 수행하는 데 쓰이는 것이 바로 if 문이다.

위와 같은 상황을 파이썬에서는 다음과 같이 표현할 수 있다.

```
>>> money = 1
>>> if money:
...     print("택시를 타고 가라")
... else:
...     print("걸어 가라")
...
택시를 타고 가라
```

money에 입력된 1은 참이다. 따라서 if문 다음의 문장이 수행되어 '택시를 타고 가라'가 출력된다.

## if문의 기본 구조

다음은 if와 else를 이용한 조건문의 기본 구조이다.

```
if 조건문:  
    수행할 문장1  
    수행할 문장2  
    ...  
else:  
    수행할 문장A  
    수행할 문장B  
    ...
```

조건문을 테스트해서 참이면 if문 바로 다음의 문장(if 블록)들을 수행하고, 조건문이 거짓이면 else문 다음의 문장(else 블록)들을 수행하게 된다. 그러므로 else문은 if문 없이 독립적으로 사용할 수 없다.

## 들여쓰기

if문을 만들 때는 if 조건문: 바로 아래 문장부터 if문에 속하는 모든 문장에 들여쓰기(indentation)를 해주어야 한다. 다음과 같이 조건문이 참일 경우 "수행할 문장1"을 들여쓰기 했고 "수행할 문장2"와 "수행할 문장3"도 들여쓰기를 해주었다. 다른 프로그래밍 언어를 사용했던 사람들은 파이썬에서 "수행할 문장"들을 들여쓰기 하는 것을 무시하는 경우가 많으니 더 주의해야 한다.

```
if 조건문:  
    수행할 문장1  
    수행할 문장2  
    수행할 문장3
```

다음과 같이 작성하면 오류가 발생한다. "수행할 문장2"를 들여쓰기 하지 않았기 때문이다.

```
if 조건문:  
    수행할 문장1  
수행할 문장2  
    수행할 문장3
```

## while문의 기본 구조

반복해서 문장을 수행해야 할 경우 while문을 사용한다. 그래서 while문을 반복문이라고도 부른다.

다음은 while문의 기본 구조이다.

```
while <조건문>:  
    <수행할 문장1>  
    <수행할 문장2>  
    <수행할 문장3>  
    ...
```

while문은 조건문이 참인 동안에 while문 아래에 속하는 문장들이 반복해서 수행된다.

"열 번 찍어 안 넘어 가는 나무 없다" 라는 속담을 파이썬 프로그램으로 만든다면 다음과 같이 될 것이다.

```
>>> treeHit = 0  
>>> while treeHit < 10:  
...     treeHit = treeHit + 1  
...     print("나무를 %d번 찍었습니다." % treeHit)  
...     if treeHit == 10:  
...         print("나무 넘어갑니다.")  
...  
나무를 1번 찍었습니다.  
나무를 2번 찍었습니다.  
나무를 3번 찍었습니다.  
나무를 4번 찍었습니다.  
나무를 5번 찍었습니다.  
나무를 6번 찍었습니다.  
나무를 7번 찍었습니다.  
나무를 8번 찍었습니다.  
나무를 9번 찍었습니다.  
나무를 10번 찍었습니다.  
나무 넘어갑니다.
```

## 너무 긴데? 줄여볼까? - 2 반복문

### for문의 기본 구조

for문의 기본적인 구조는 다음과 같다.

```
for 변수 in 리스트(또는 튜플, 문자열):  
    수행할 문장1  
    수행할 문장2  
    ...
```

리스트나 튜플, 문자열의 첫 번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 "수행할 문장1", "수행할 문장2" 등이 수행된다.

### 예제 이용해 for문 이해하기

for문은 예제를 통해서 살펴보는 것이 가장 알기 쉽다. 다음 예제를 직접 입력해 보자.

#### 1. 전형적인 for문

```
>>> test_list = ['one', 'two', 'three']  
>>> for i in test_list:  
...     print(i)  
...  
one  
two  
three
```

['one', 'two', 'three']라는 리스트의 첫 번째 요소인 'one'이 먼저 i 변수에 대입된 후 print(i)라는 문장을 수행한다. 다음에 'two'라는 두 번째 요소가 i 변수에 대입된 후 print(i) 문장을 수행하고 리스트의 마지막 요소까지 이것을 반복한다.

4회 반복



`for i in range(4):`  
`print(i)`

공급 데이터

## Code

```
import turtle

smart = turtle.Turtle()

# Loop 4 times. Everything I want to repeat is
# *indented* by four spaces.
for i in range(4):
    smart.forward(50)
    smart.right(90)

# This isn't indented, so we aren't repeating it.
turtle.done()
```

## Expected Output



## Code

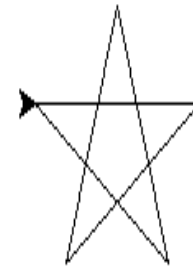
```
import turtle

star = turtle.Turtle()

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

turtle.done()
```

## Expected Output





# TURTLE - 실습(과제?)

4-1

