



AI 프로그래밍

융합학과 권오영

oykwon@koreatech.ac.kr



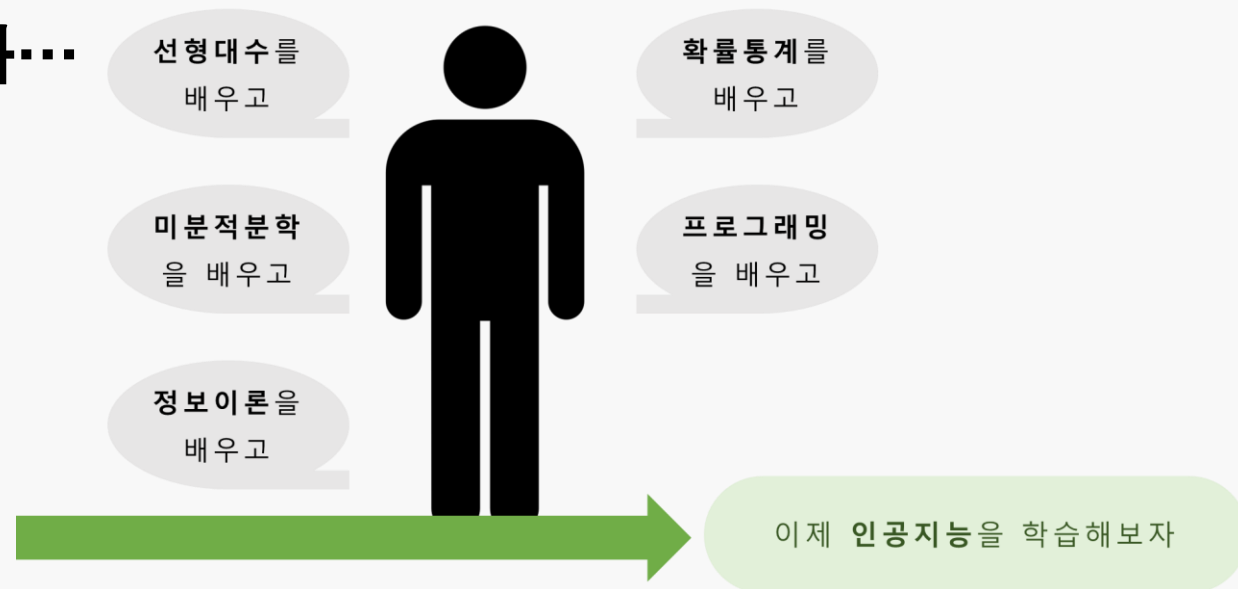
01

III. 소개

인공지능학습의 시작

인공지능응용 개발은 어디서부터 시작해야하나?

- Bottom up 접근
- 선형대수, 미분적분학, 확률과통계,
프로그래밍등 차근차근 배워나가자...



<https://machinelearningmastery.com/machine-learning-for-programmers/>

인공지능응용 개발은 어디서부터 시작해야하나?

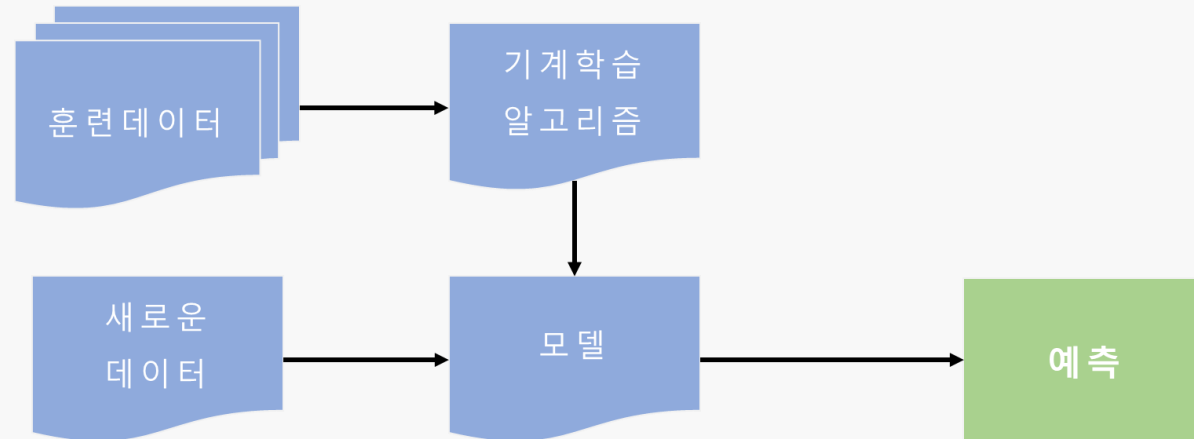
- Top down 접근
- 목표(결과물)를 먼저 설정하고, 목표 달성에 필요한 내용을 채워나감
 - Step 1: 마음가짐 조정 (신념:할 수 있다)
 - Step 2: 절차(프로세스) 선택
 - Step 3: 도구 선택 (구현을 하기 위해)
 - Step 4: 데이터세트를 갖고 연습
 - Step 5: 포트폴리오 구축 (실력을 보여주자).

출처 <https://machinelearningmastery.com/machine-learning-mastery-method/>

인공지능응용 가장 많이 쓰이는 분야

- 예측 모델링

- 데이터 수집
- 데이터를 활용해서 모델학습
- 새로운 데이터가 입력되었을 때 예측



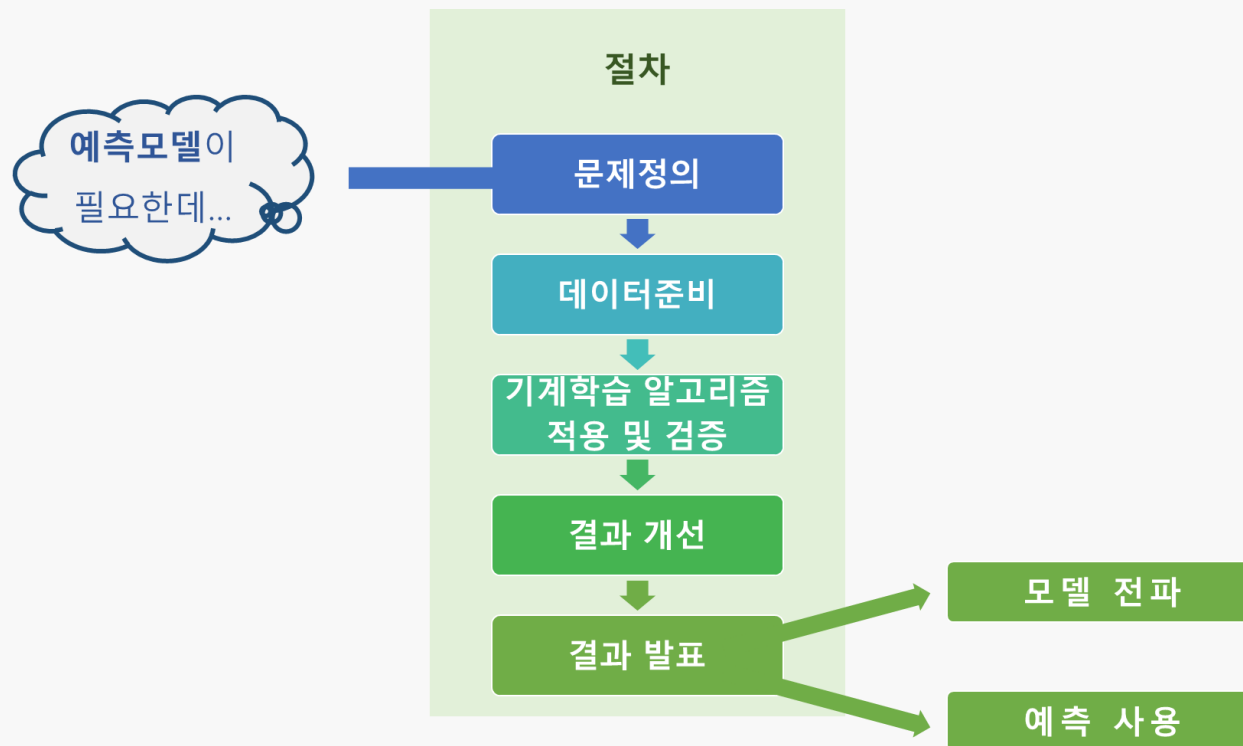
- 예) 구글 티처블머신

<https://teachablemachine.withgoogle.com>

인공지능응용 만들기

• 인공지능응용개발 프로세스는?

- 문제정의
- 데이터준비
- 알고리즘점검
- 결과개선
- 결과발표 (활용)



출처 <https://machinelearningmastery.com/machine-learning-mastery-method/>



02

III. 개발도구

인공지능응용개발에 도움을 주는 도구들

인공지능응용개발에 적절한 도구는?

• Python

- ✓ 1991년 귀도 반 로섬 (Guido van Rossum)이 개발 (Monty Pythons Flying Circus BBC 코미디 프로그램에서 영감)
- ✓ Easy to Learn, Easy to Read, Easy to Maintain
- ✓ 풍부한 라이브러리
(남들이 만들어 놓은 것이 아주 많다.)
- ✓ 다양한 플랫폼에서 사용가능하고,
다양한 분야에 활용
(Web, Game, GUI, Data Analytics,
Machine Learning, IOT...etc)



인공지능응용개발에 적절한 도구는?

- Python 패키지 (계산)

- NumPy: 수치 및 과학 계산을 위한 기본 라이브러리로 숫자 형 배열, 선형 대수 도구, 난수 기능 등을 위한 데이터 구조가 포함되어 있음
- SciPy: 최적화, 보간, 통계 및 신호 처리와 같은 과학 컴퓨팅을 위한 다양한 기능을 제공
- Matplotlib: Python의 핵심 plotting 라이브러리며 주피터 노트북에서 매직 명령어를 이용하여 `%matplotlib notebook` 또는 `%matplotlib inline` 형태로 사용가능
- Sympy: Symbolic 계산 지원
- Pandas: 데이터 분석을 위한 리소스와 레이블이 지정된 테이블 형식의 데이터를 위한 유연한 데이터 구조 제공

인공지능응용개발에 적절한 도구는?

- Python 패키지 (기계학습)

- Scikit-learn: 머신러닝 라이브러리(예측분석을 위한 각종 도구와 알고리즘 제공)
- Keras: TensorFlow위에서 수행할 수 있는 상위 수준의 오픈 소스 딥러닝 라이브러리 (초보자가 신경망을 쉽게 구성할 수 있다.)
- TensorFlow: 데이터 흐름(data flow) 프로그래밍을 위한 오픈소스 라이브러리로 인공 신경망과 같은 기계학습 프로그램에 널리 사용

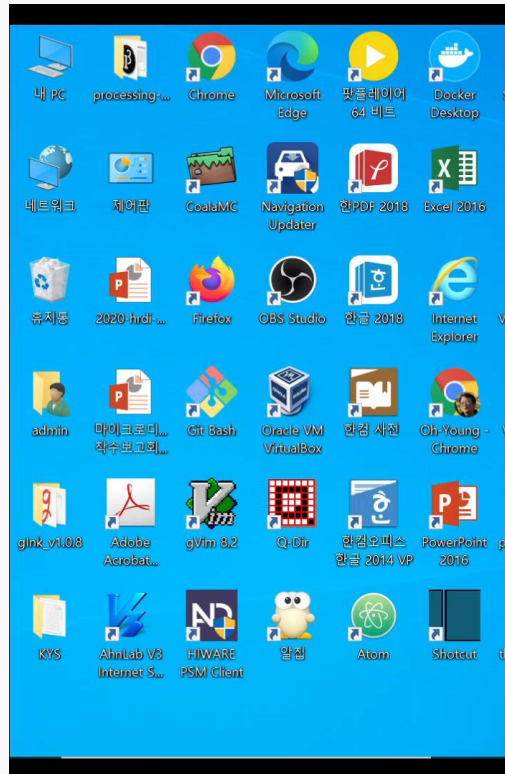


파이선 IDE (통합개발환경)

- Thonny (초보자용 <https://thonny.org/>)
 - MicroPython, Micro:Bit, Raspberry Pi 등도 쉽게 연결
- PyCharm (전문개발자용 <https://www.jetbrains.com/ko-kr/pycharm/>)
- Visual Studio Code (<https://code.visualstudio.com/>)
- Jupyter Notebook (<https://jupyter.org/>)
- Ananconda (<https://www.anaconda.com/>)

Thonny <https://youtu.be/nwlgxrXP-X4>

- <https://thonny.org/>
- 64/32비트 파이썬 번들을 기본으로 포함
 - 일반적인 파이썬 코딩
 - Micropython 지원
 - ✓ Pyboard
 - ✓ Micro:bit
- 라즈베리파이보드 설치
 - pip install thonny



Thonny
Python IDE for beginners

Download version **3.2.7** for
[Windows](#) • [Mac](#) • [Linux](#)

NB! Windows installer is signed with new identity and you may receive a warning dialog from Defender until it gains more reputation.

Just click "More info" and "Run anyway".

Features

Easy to get started. Thonny comes with Python 3.7 built in, so just one

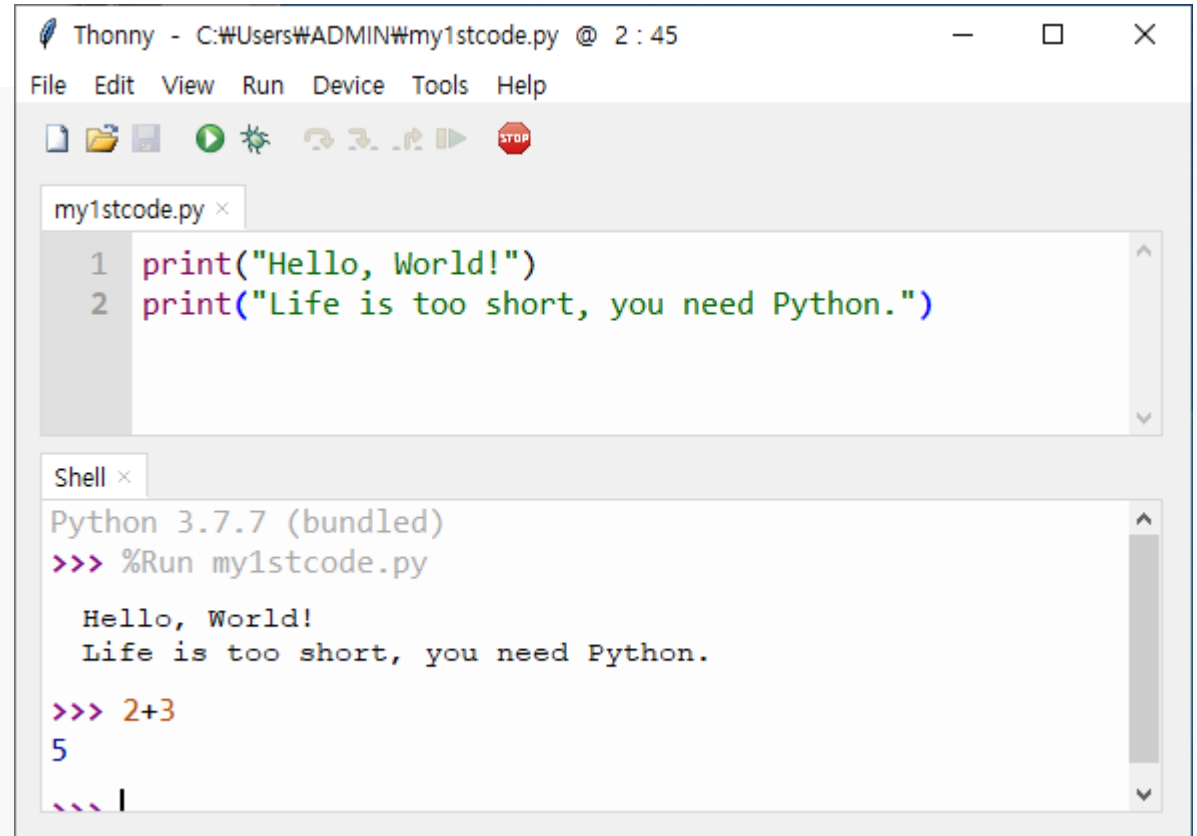
❖ 파이썬 실습

Thonny 설치 및 실행

my1stcode.py

저장 및 실행(Run button)

수행종료(Stop button)



The screenshot shows the Thonny Python IDE interface. The title bar reads "Thonny - C:\Users\ADMIN\my1stcode.py @ 2:45". The menu bar includes "File", "Edit", "View", "Run", "Device", "Tools", and "Help". The toolbar contains icons for file operations, running, and stopping. The main editor window, titled "my1stcode.py", contains the following Python code:

```
1 print("Hello, World!")
2 print("Life is too short, you need Python.")
```

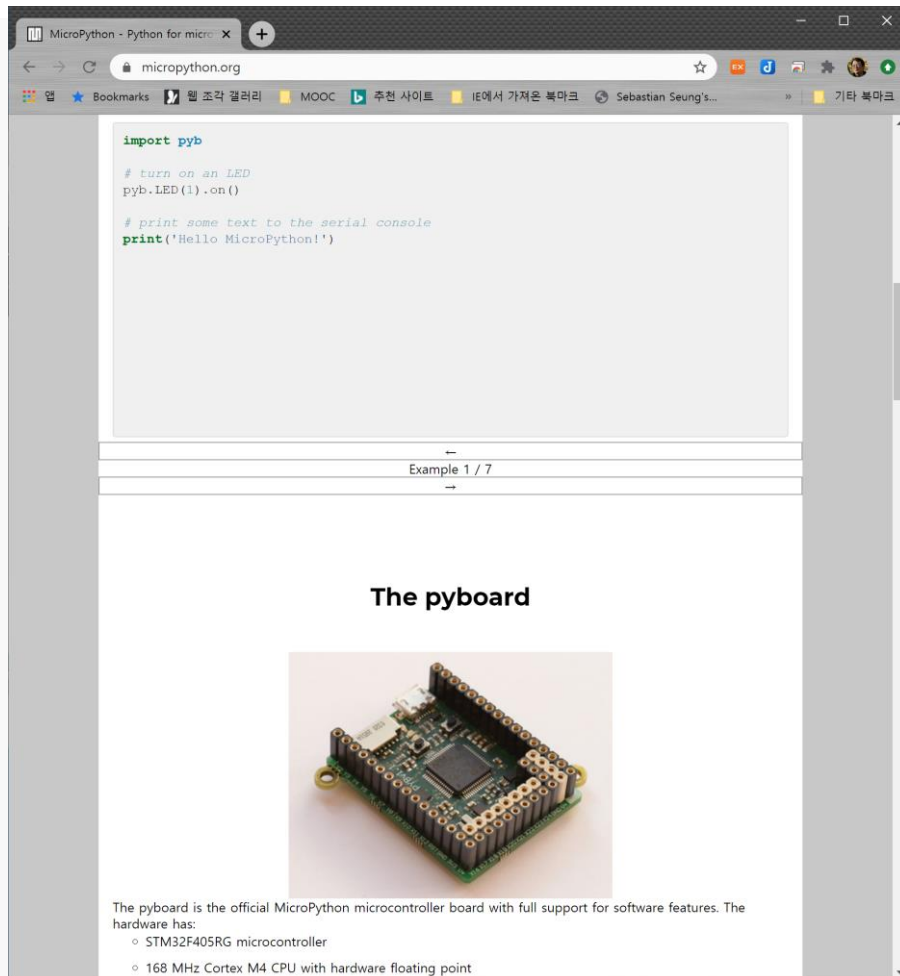
Below the editor is a "Shell" window titled "Shell ×". It shows the output of running the script using the command `>>> %Run my1stcode.py`. The output is:

```
Python 3.7.7 (bundled)
>>> %Run my1stcode.py

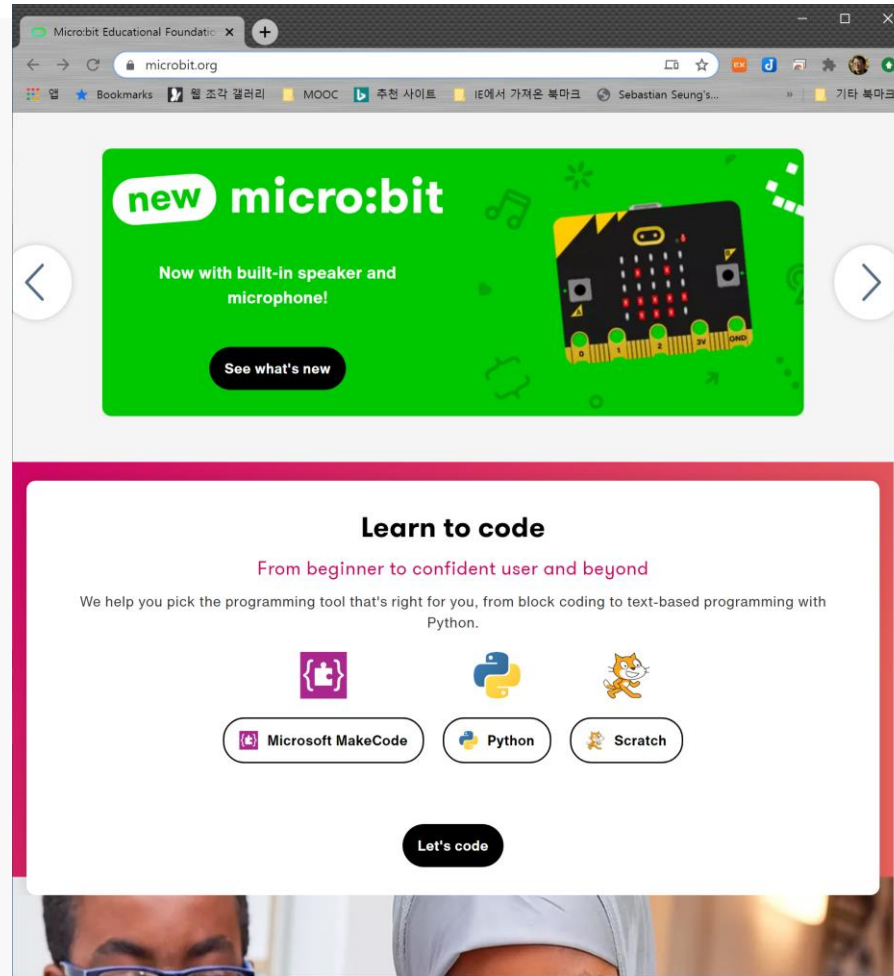
Hello, World!
Life is too short, you need Python.

>>> 2+3
5
>>> |
```

Micropython



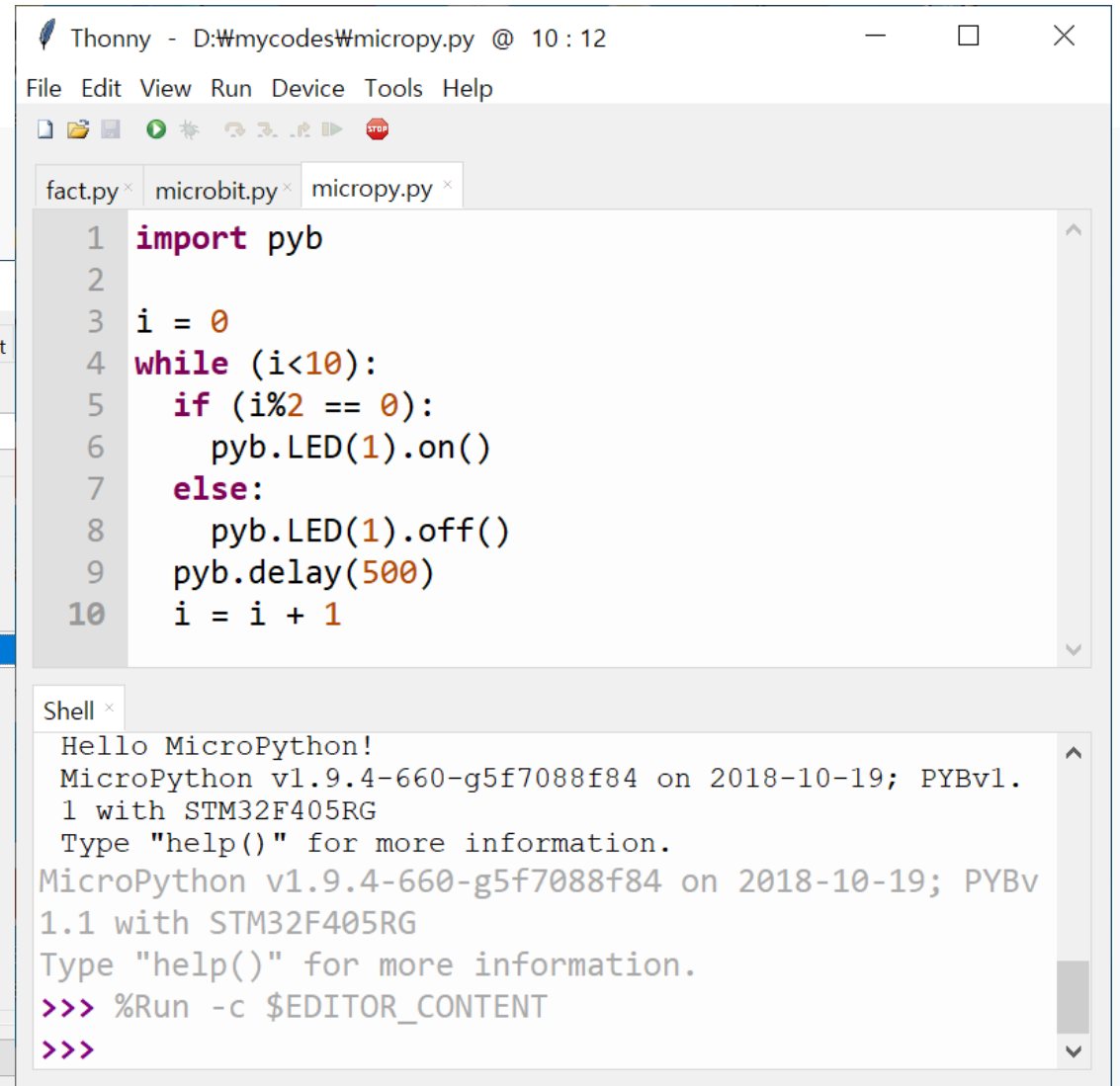
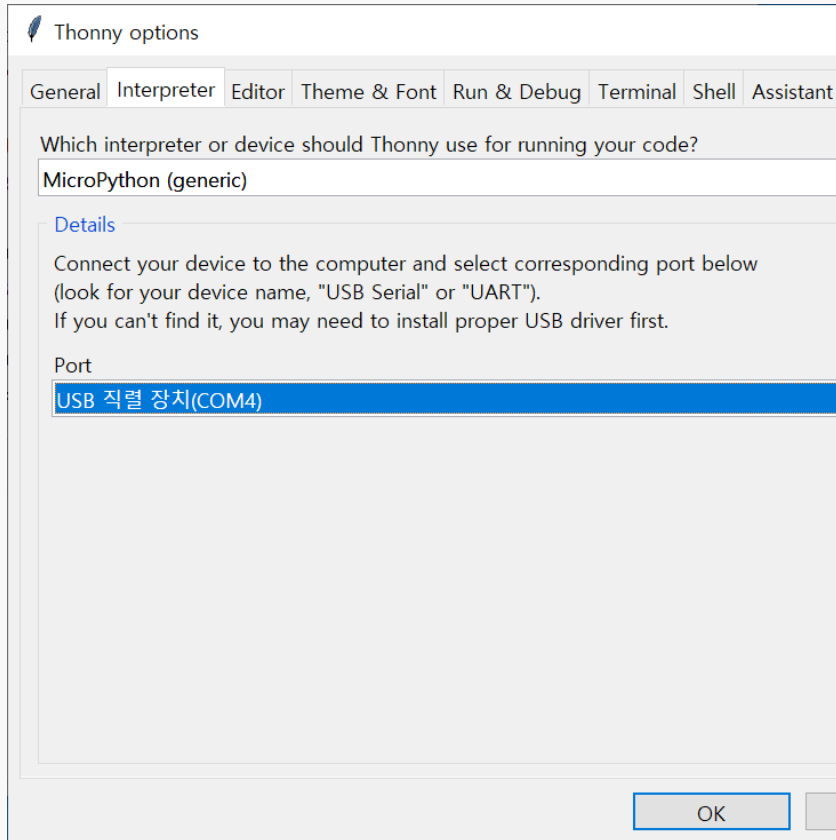
<https://micropython.org/>



<https://microbit.org/>

Micropython

- Interpreter 변경
(python)
- 포트설정



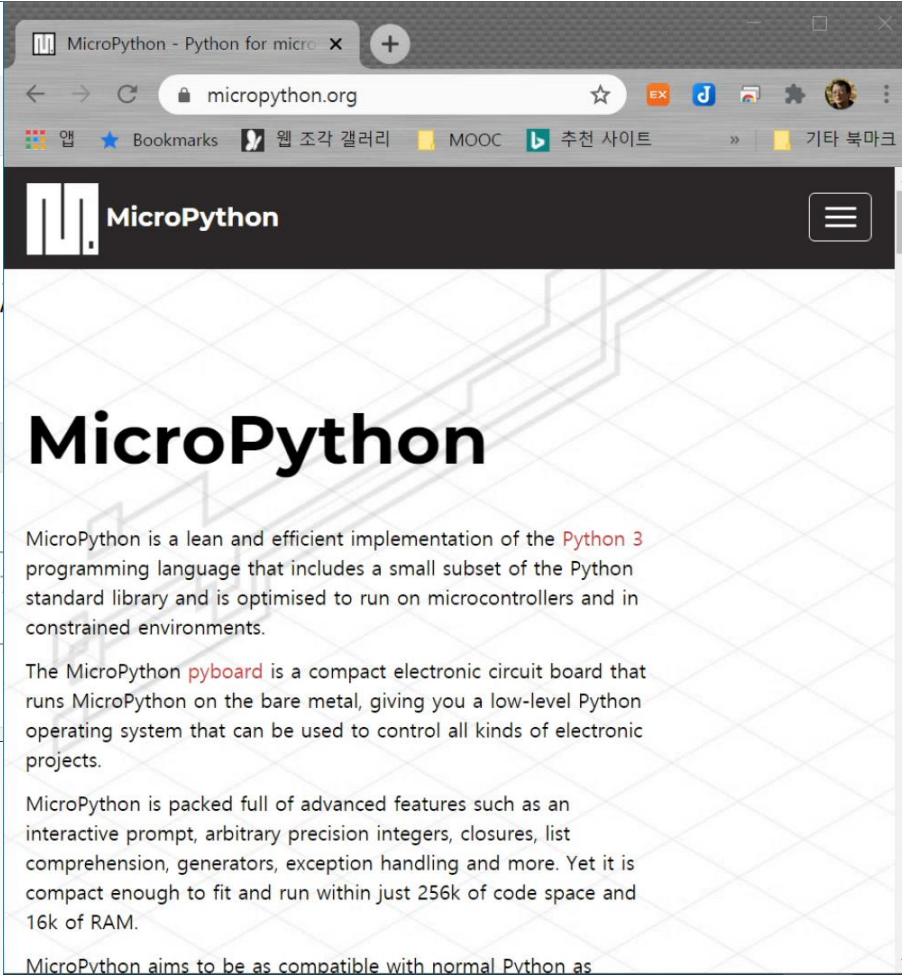
Micropython

```
Thonny - D:\mycodes\microbit.py @ 6:16
File Edit View Run Device Tools Help

fact.py x microbit.py x
1 from microbit import *
2
3 while True:
4     display.scroll('Hello')
5     display.show(Image.HEART)
6     sleep(2000)

Shell x
6
>>>

MicroPython v1.9.2-34-gd64154c
:bit v1.0.0-rc.5 with nRF51822
Type "help()" for more information
>>> %Run -c $EDITOR_CONTENT
```



The screenshot shows the MicroPython website in a web browser. The page has a dark header with the MicroPython logo and a hamburger menu. The main content area has a light background with a circuit board pattern. The title "MicroPython" is prominently displayed. Below the title, there is a paragraph describing MicroPython as a lean and efficient implementation of the Python 3 programming language. Another paragraph describes the MicroPython pyboard as a compact electronic circuit board. The bottom of the page mentions that MicroPython aims to be as compatible with normal Python as possible.

MicroPython - Python for micro x

micropython.org

MicroPython

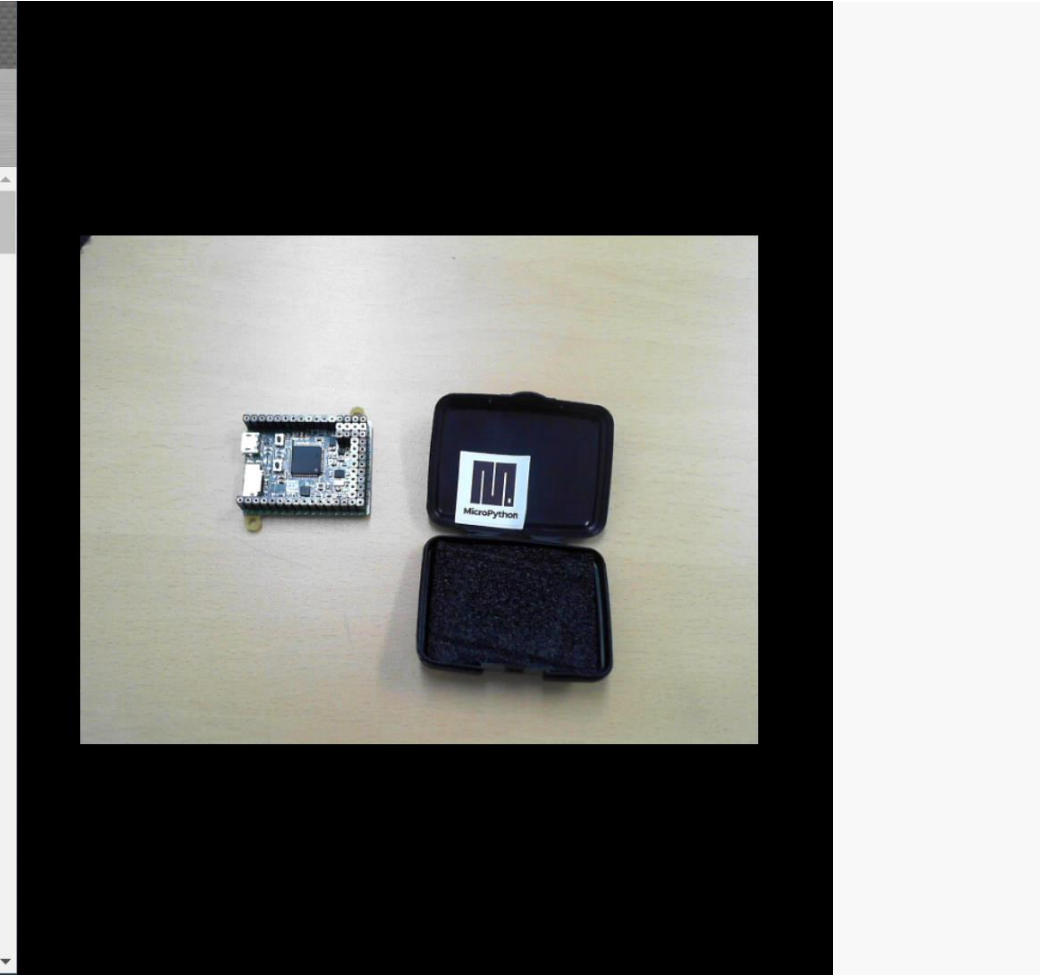
MicroPython

MicroPython is a lean and efficient implementation of the **Python 3** programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments.

The MicroPython **pyboard** is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as



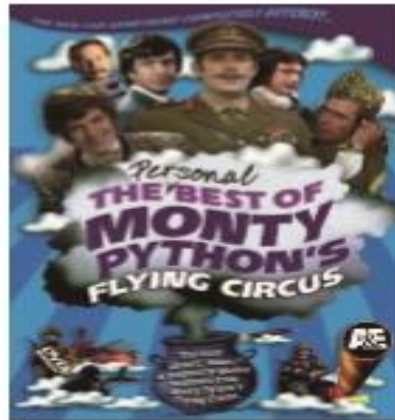
파이썬

파이썬

- ❖ 1991년 귀도 반 로섬(Guido van Rossum)이 개발
 - Monty Pythons Flying Circus BBC 코미디
- ❖ 컴퓨터 프로그래밍 언어, 사람과 친화적인 언어, 비교적 쉬운 언어, 인터프리터 언어(바로 결과), 가장 활용도가 높은 언어....etc.



파이썬은 제가 좋아하는 영국
코미디 프로 이름이었어요!

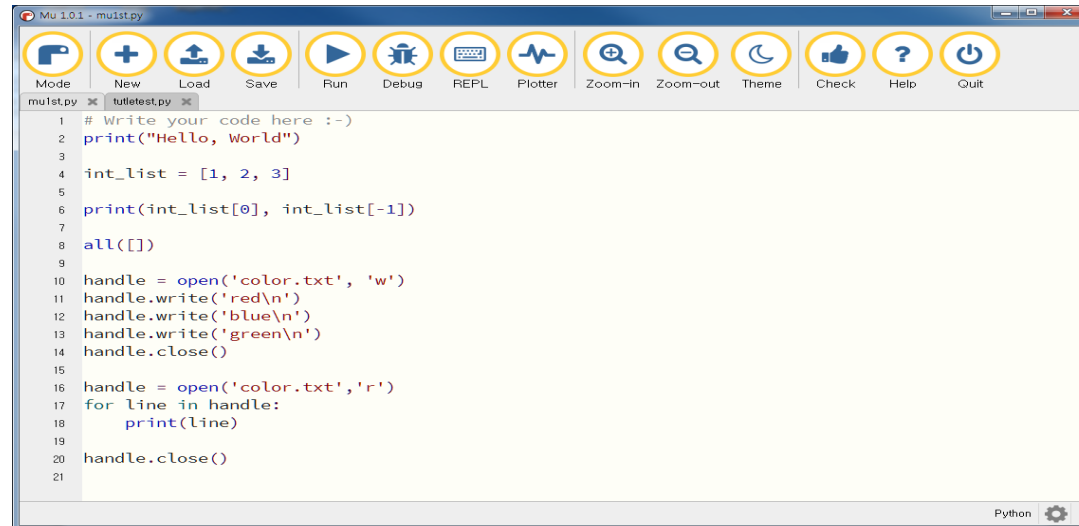
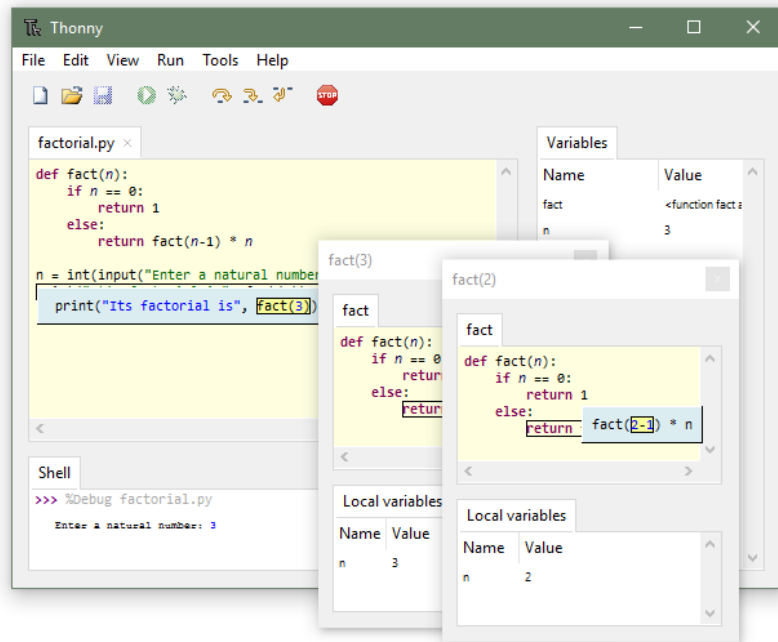


나하고는 관계가
없어!^^



파이썬

- 파이썬 프로그래밍 환경 (PC환경에서 초보인 경우)
- mu editor 활용(<https://codewith.mu/en/download>)



추천: <https://thonny.org/>

PYTHON 기본 문법

기본 자료형 (Built-in Atomic Data Types)

기본 데이터 타입(자료형)

❖ Python에 사용하는 기본 자료형

- Integer (정수)
- Floating point (실수)
- Bool (논리)
- None

❖ 자료형의 실체를 object라 하고, object들을 연산자(+,-,* 등)을 연결한 형태를 expression(표현식)이라 한다.

기본 연산자

연산자	기호	사용예	결과값
덧셈	+	7 + 4	11
뺄셈	-	7 - 4	3
곱셈	*	7 * 4	28
나눗셈	//	7 // 4	1
나눗셈	/	7 / 4	1.75
나머지	%	7 % 4	3

지수승은 ** 사용: 2**3은 2³의미

```
>>> 3 + 2
```

```
5
```

```
>>> 3.0 + 2.0
```

```
5.0
```

```
>>> 3 != 2
```

```
True
```

```
>>> type (3)
```

```
<type 'int '>
```

```
>>> type (3.0)
```

```
<type 'float '>
```

비교 연산자

== (equal)

!= (not equal)

> (greater)

>= (at least)

< (less)

<= (at most)

논리 연산자

a and b

a or b

not a

변수

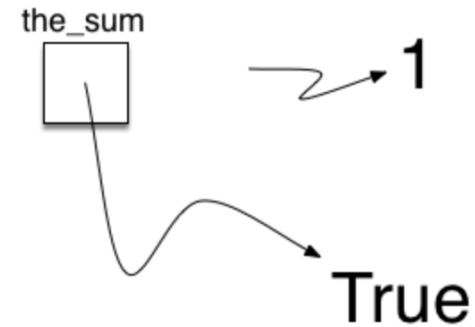
❖ 변수: 자료형을 저장하는 공간

the_sum = 0

the_sum = the_sum + 1

the_sum = True

(왼쪽에 있는 이름이 가리키는 장소에
오른쪽에 있는 값을 넣는다.)



pi = 3

radius = 11

area = pi * (radius ** 2)

radius = 14

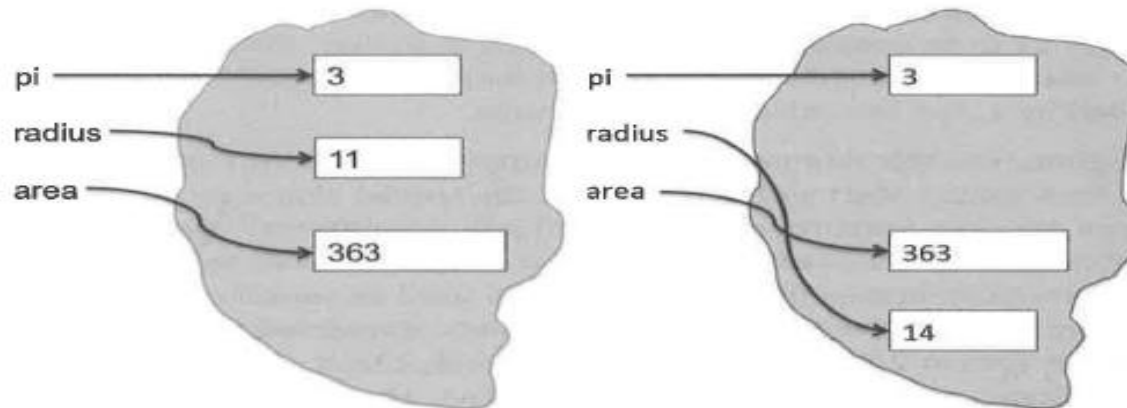


Figure 2.2 Binding of variables to objects

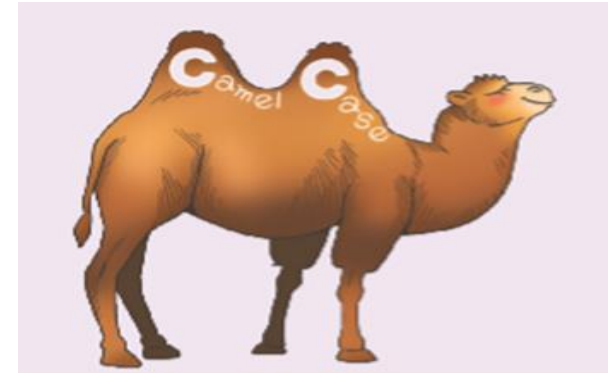
변수

❖ 변수 이름을 지을 때 주의 사항

- 의미 있는 이름을 사용하자
 - a, b, c 보다는 year, month, date과 같은 변수 명이 좋다
- 소문자와 대문자는 서로 다르다.
 - a와 A는 다른 변수로 취급
- 변수의 이름은 영문자와 숫자, 밑줄(_) 만 허용
(기호, 띄어쓰기 X)
 - boxVolume, numberOfPictures, king3, money# (X)

■ 예약어는 사용할 수 없다

and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, with, while, yield



myNewCar

변수

- ❖ 변수는 다른 변수의 값도 저장 가능

```
>>> score = 20  
>>> x = score  
>>> print(x)
```

- ❖ 수학에서의 = 와 python에서 = 의 차이

- Equal vs. Assignment
- 다중 할당을 허용 ($x, y = 2, 3$)

- ❖ 변수는 어떤 데이터이든 저장 (다양한 자료 형태 지원)

```
value = 3  
value = 3.14  
value = "hello"
```


주석

- ❖ 소스코드에 붙이는 설명글
- ❖ 주석은 #로 시작되는 문자열로 python이 해석하지 않음

```
# 이 프로그램은 사용자로 부터 2개의 정수를 받아 합을 구한다.  
# 첫번째 정수 받기  
x= int(input("첫번째 정수:"))  
# 두번째 정수 받기  
y= int(input("두번째 정수:"))  
# 정수의 합을 계산  
sum = x+y  
# 결과 값을 화면에 출력  
print("합은", sum)
```

```
#subtract area of square 5 from area of circle c  
areaC = pi*radius**2  
areaS = side*side  
Difference = areaC-areaS
```

입력

❖ 입력 : input()

```
a,b = input("please input").split()  
print (type(a), type(b))
```

```
a,b = input("please input").split(",")
```

```
a,b = map(int, input("please input").split(","))  
print (type(a), type(b))
```

출력

❖ 출력 : print()

값을 여러 개 출력 => 여러 값은 공백으로 구분되어 출력됨

```
print (값1, 값2, 값3)
```

```
print (변수1, 변수2, 변수3)
```

```
print ("%d * %05d = %5.1f" % (100, 200, 300.0))
```

참고 : %d, %x, %o, %f %c, %s

: %5.1f

```
print("{0:d} {1:5d} {2:05d}".format(100, 200, 300))
```

```
print("{2:d} {0:d} {1:d}".format(100, 200, 300))
```

여러 값을 공백이 아닌 다른 문자로 구분 => sep='문자 또는 문자열'

```
print (값1, 값2, 값3, sep=",")
```

Print without the Newline => end='문자 또는 문자열'

```
print (값1, 값2, 값3, end=",")
```

Print Escape Sequence

```
print ("줄바꿈\n연습")
```

Python 기초

선택과 반복

Flow Control

IF 문 (조건문)

if .. else
한 가지 조건 분기(선택)

if .. else
한 가지 조건 분기(선택)

if .. else
한 가지 조건 분기(선택)

반복문 (loop)

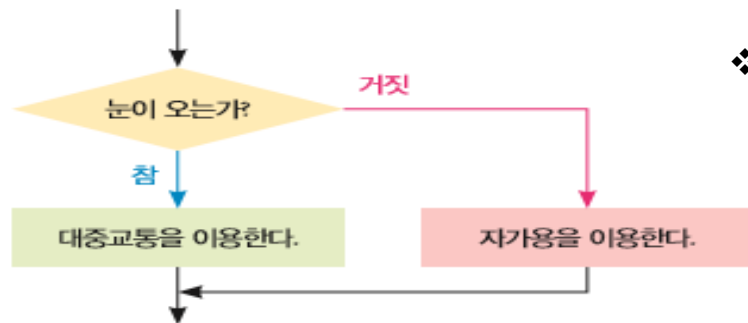
while
조건이 참인 동안 반복

for
순서열의 처음부터 끝까지

continue, break
반복문 중단,
해당 반복만 건너뛰기

if ~ else 문

❖ 조건의 참/거짓에 따라 할 일을 선택

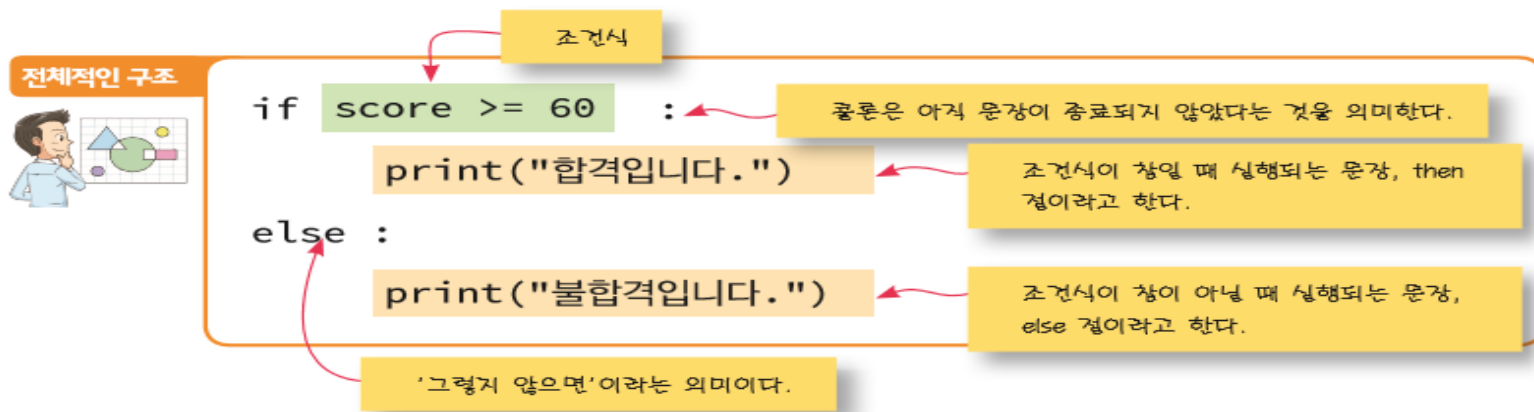


❖ 거짓(False)로 취급하는 것

- ✓ None, False, 숫자 0 (0, 0.0, 0j)
- ✓ 비어있는 문자열, 리스트, 튜플, 세트 : "", [], (), set()
- ✓ 클래스 인스턴스의 __bool__(), __len__() 메서드가 0 또는 False를 반환할 때
- ✓ 그 외에는 True로

❖ Syntax

- 참 영역이나 거짓 영역의 하위 블록을 표시하기 위해 꼭 들여쓰기를 해야함



조건식

❖ if 문 안의 조건식을 구성하는 연산자

표 3.1 관계 연산자

연산	의미
$x == y$	x와 y가 같은가?
$x != y$	x와 y가 다른가?
$x > y$	x가 y보다 큰가?
$x < y$	x가 y보다 작은가?
$x >= y$	x가 y보다 크거나 같은가?
$x <= y$	x가 y보다 작거나 같은가?

C: block을 { c_statements; }

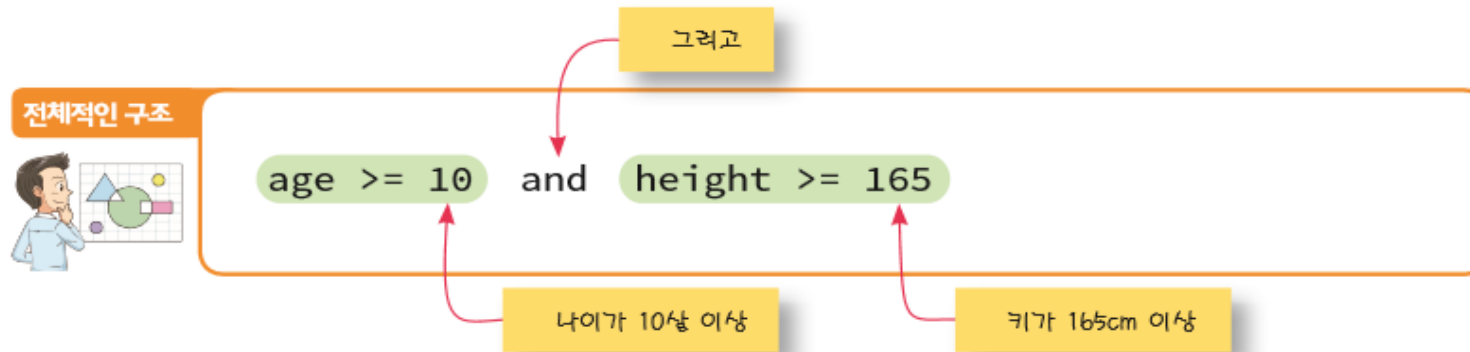
Python: indentation level(들여쓰기)로 블록표현

```
if x < y :
    들여쓰기 print('x is min')
else:
    들여쓰기 print('y is min')
```

조건식 연습 문제:
짝수, 홀수를 판단하려면?

복합 조건식

- ❖ 조건식을 and, or, not 등을 활용하여 확장
 - 연산 순서를 명확히 하기 위하여 괄호를 적극적으로 이용



```
age = 20
height = 180
if( (age>=10) and (height>=165)) :
    print("놀이 기구를 탈 수 있습니다.")
else :
    print("놀이 기구를 탈 수 없습니다.")
```

if ~ else의 내포

❖ if 문 안에 다시 if 문이 내포되는 형태

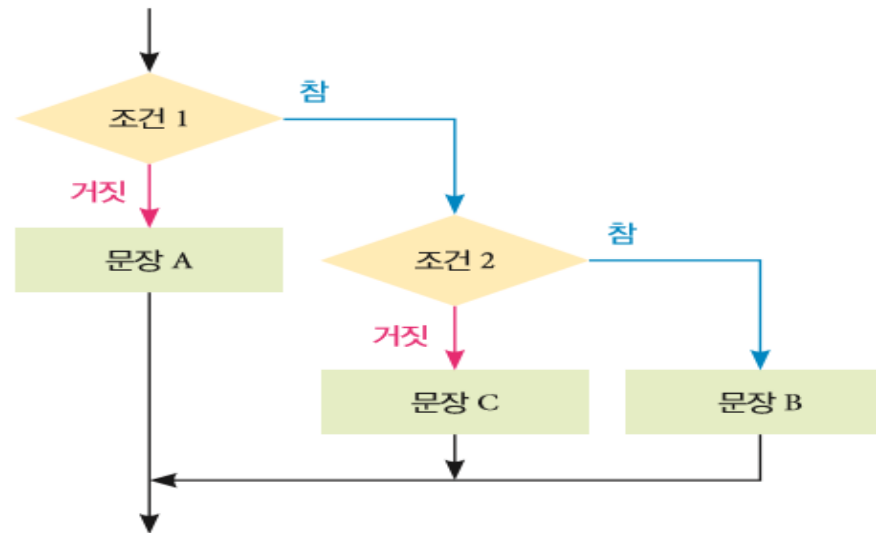
전체적인 구조



```
if 조건1 :  
    문장_A
```

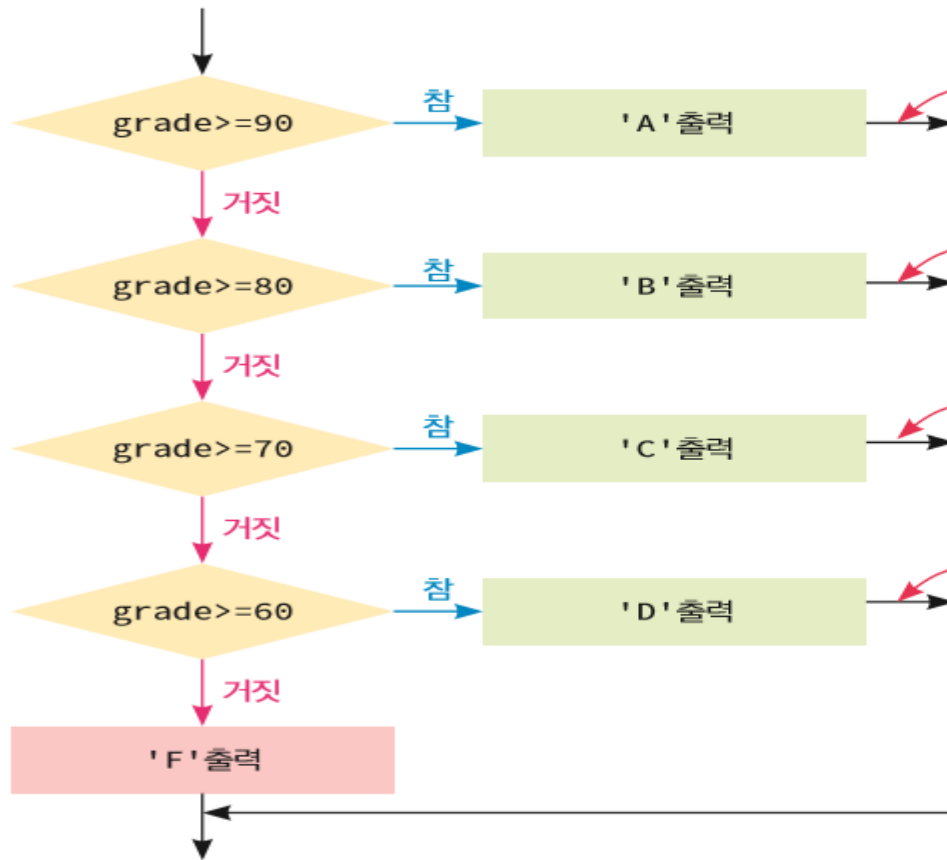
```
else:
```

```
    if 조건2 :  
        문장_B  
    else:  
        문장_C
```



if ~ elif ~ ~ else 구조

❖ 조건구간의 세분화 (elif : else if 의 축약형)

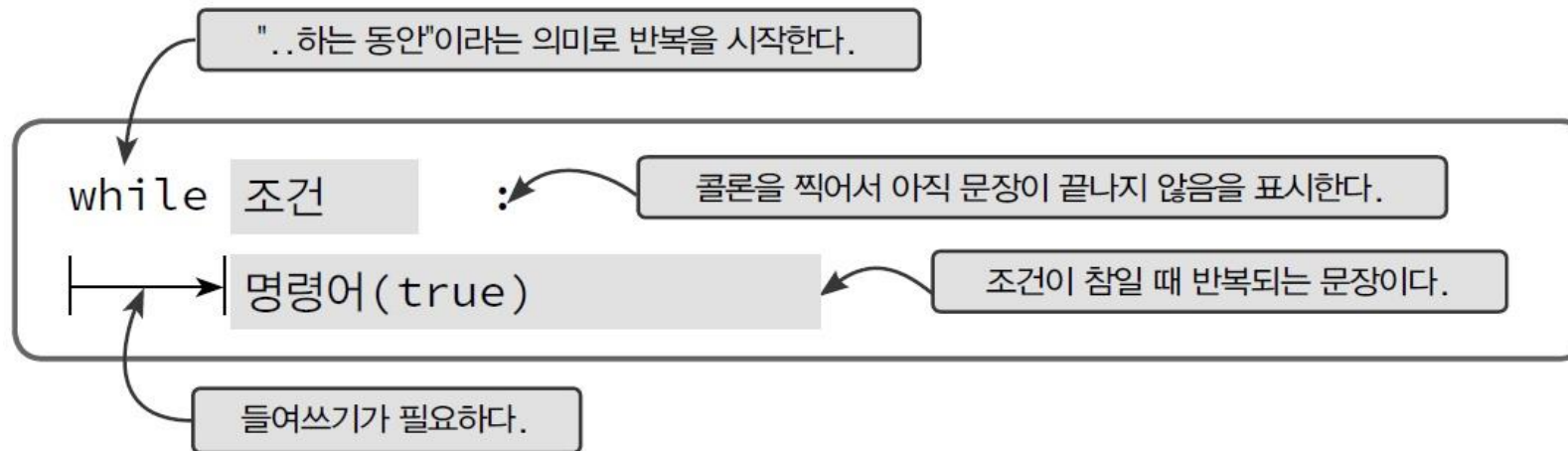
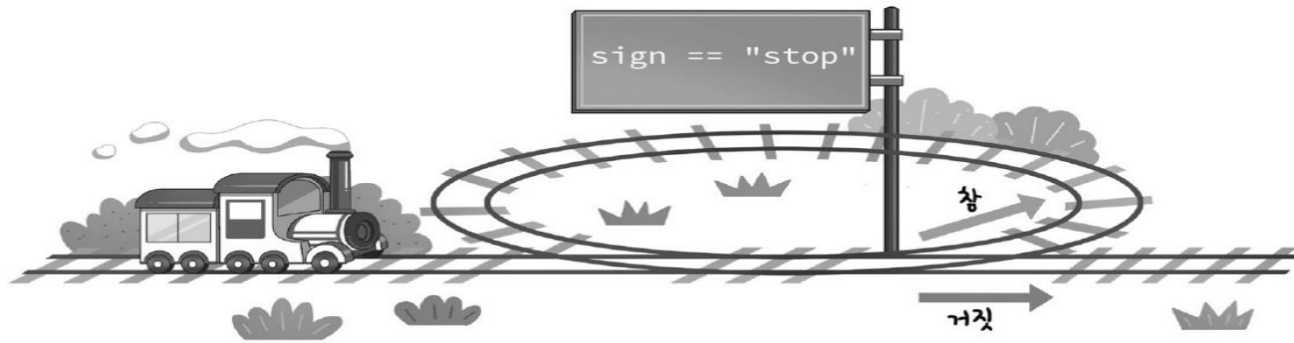


```
grade = int(input("성적을 입력하시오: "))
```

```
if grade >= 90 :  
    print("학점 A")  
elif grade >= 80 :  
    print("학점 B")  
elif grade >= 70 :  
    print("학점 C")  
elif grade >= 60 :  
    print("학점 D")  
else :  
    print("학점 F");
```

반복 (while)

❖ 조건이 참인 동안 반복



반복 (while 사용예)

```
sign = "stop"

while sign == "stop":
    sign = input("현재 신호를 입력하시오: ")    # 반복되는 부분은 들여쓰기
print("OK! 진행합니다.")
```

수행결과

```
현재 신호를 입력하시오: stop
현재 신호를 입력하시오: stop
현재 신호를 입력하시오: stop
현재 신호를 입력하시오: stop
현재 신호를 입력하시오: stop
현재 신호를 입력하시오: go
OK! 진행합니다.
```

반복(for)

for variable in sequence:

code block

❖ Sequence 를 만들 수 있는 내재(built-in) 함수 range()

`range([start,] stop [, step])` : start ~ stop -1 까지 step 단위로 증가

✓ start 의 기본값은 0, step의 기본값은 1

✓ start, stop, step 은 정수만 인식

```
>>> for i in range(5):
```

```
...     print(i)
```

```
...
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

```
>>> for i in range(3, 6):
```

```
...     print(i)
```

```
...
```

```
3
```

```
4
```

```
5
```

```
>>> for i in range(4, 10, 2):
```

```
...     print(i)
```

```
...
```

```
4
```

```
6
```

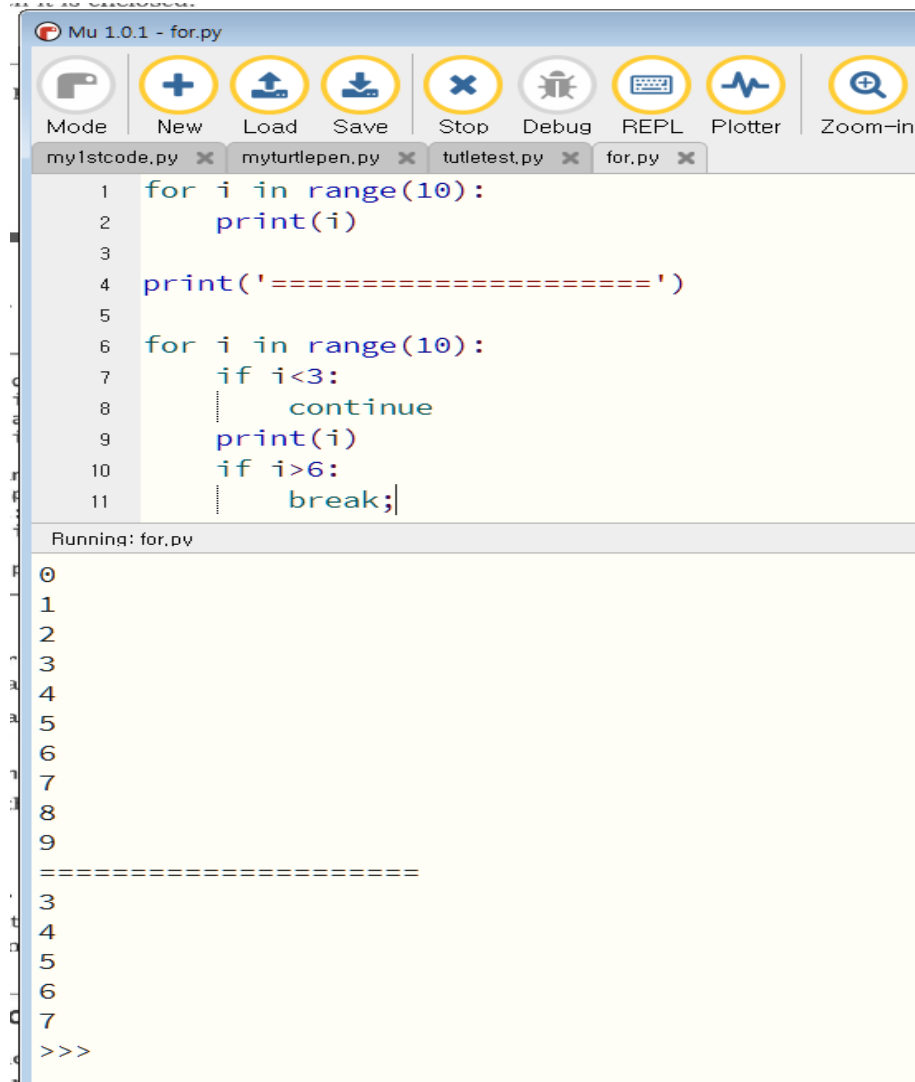
```
8
```

반복조절(break; continue)

- ❖ Sequence에는 수치만 사용되는 것이 아니라 순서를 표현할 수 있는 요소는 다 사용될 수 있다.

```
total = 0
for c in '123456789':
    total = total + int(c)
print(total)
```

- ❖ 루프 블록의 탈출
break
- ❖ 루프 블록의 나머지 명령어 skip
continue



```
Mu 1.0.1 - for.py
Mode New Load Save Stop Debug REPL Plotter Zoom-in
my1stcode.py x myturtlepen.py x tutletest.py x for.py x
1 for i in range(10):
2     print(i)
3
4 print('=====')
5
6 for i in range(10):
7     if i<3:
8         continue
9     print(i)
10    if i>6:
11        break;

Running: for.py
0
1
2
3
4
5
6
7
8
9
=====
3
4
5
6
7
>>>
```


파이썬 디버거 pdb

- ❖ <https://docs.python.org/2/library/pdb.html>
- ❖ 인터프리터 내부에서 pdb 모듈을 불러서 실행

```
>>> import pdb
```

```
>>> import mymodule
```

```
>>> pdb.run('mymodule.test()')
```

```
> <string>(0)?()
```

```
(Pdb) continue
```

```
> <string>(1)?()
```

```
(Pdb) continue
```

```
NameError: 'spam'
```

```
> <string>(1)?()
```

```
(Pdb)
```

- `python -m pdb myscript.py`

c(ontinue)

멈춰있는 program을 다시 run 하게 한다.

b(reak)

break point 를 설정한다.

n(ext)

다음 line 으로 넘어간다.

s(tep)

함수안으로 들어간다.

p expression

expression을 평가해서 값을 보여준다.

pp expression

expression 평가값을 pretty-print 로 보여준다.

l(ist) [first[, last]]

일정 구간의 source code 를 보여준다.